

For Trainer use or to accompany paid student lab manuals **in class only**



Microsoft Cloud Workshop

Media AI

Hands-on lab step-by-step

March 2018

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

The names of manufacturers, products, or URLs are provided for informational purposes only and Microsoft makes no representations and warranties, either expressed, implied, or statutory, regarding these manufacturers or the use of the products with any Microsoft technologies. The inclusion of a manufacturer or product does not imply endorsement of Microsoft of the manufacturer or product. Links may be provided to third party sites. Such sites are not under the control of Microsoft and Microsoft is not responsible for the contents of any linked site or any link contained in a linked site, or any changes or updates to such sites. Microsoft is not responsible for webcasting or any other form of transmission received from any linked site. Microsoft is providing these links to you only as a convenience, and the inclusion of any link does not imply endorsement of Microsoft of the site or the products contained therein.

© 2018 Microsoft Corporation. All rights reserved.

Microsoft and the trademarks listed at <https://www.microsoft.com/en-us/legal/intellectualproperty/Trademarks/Usage/General.aspx> are trademarks of the Microsoft group of companies. All other trademarks are property of their respective owners.

Contents

| | |
|--|----|
| Media AI hands-on lab step-by-step | 1 |
| Abstract and learning objectives..... | 1 |
| Overview..... | 1 |
| Solution architecture | 2 |
| Requirements | 2 |
| Before the hands-on lab | 3 |
| Prerequisites..... | 3 |
| Task 1: Configure a development environment..... | 3 |
| Task 2: Disable IE enhanced security | 3 |
| Task 3: Update Visual Studio Tools for Azure Functions | 4 |
| Task 4: Install Visual Studio Tools for Logic Apps..... | 5 |
| Task 5: Validate connectivity to Azure | 8 |
| Task 6: Download the exercise files | 8 |
| Exercise 1: Signup for Video Indexer API Service..... | 9 |
| Task 1: Signup for Video Indexer | 9 |
| Task 2: Copy Video Indexer API Key | 12 |
| Exercise 2: Setup video import workflow..... | 12 |
| Task 1: Create Storage Account for video files..... | 12 |
| Task 2: Create Azure Logic App to process videos..... | 16 |
| Exercise 3: Enable admin website to upload videos | 29 |
| Task 1: Provision Cosmos DB Account..... | 29 |
| Task 2: Integrate Cosmos DB into Admin Web App | 32 |
| Task 3: Integrate File Upload into Admin Web App..... | 37 |
| Task 4: Add ability to delete video | 40 |
| Task 5: Deploy Admin website to an Azure Web App | 46 |
| Task 6: Configure Application Settings..... | 50 |
| Exercise 4: Update video status when processing is complete | 56 |
| Step 1: Create Azure Function..... | 56 |
| Step 2: Update Cosmos DB Document with Video Processing State..... | 64 |
| Step 3: Update Video State when processing is complete..... | 74 |
| Exercise 5: Add Video Player to Front-End Application..... | 82 |
| Step 1: Integrate Cosmos DB into Front-End Application | 82 |
| Step 2: Display Video Thumbnail Image..... | 83 |
| Step 3: Add Video Player..... | 85 |
| Step 4: Add Video Insights | 86 |
| Step 5: Integrate Video Player and Insights together..... | 86 |
| Step 6: Deploy Public website to an Azure Web App..... | 86 |
| Step 7: Configure Application Settings | 90 |

| | |
|---|----|
| Exercise 6: Test the application..... | 93 |
| Step 1: Upload Video to Admin website..... | 93 |
| Step 2: View Video and Insights in Public website | 95 |
| After the hands-on lab..... | 99 |
| Task 1: Delete resources..... | 99 |

Media AI hands-on lab

step-by-step

Abstract and learning objectives

In this workshop, students will learn how to build, setup, and configure a Web Application that performs media streaming using Azure Services; including the Video Indexer API. Students will also learn how to implement video processing using Logic Apps, Azure Functions, and Video Indexer API to encode and transcribe videos.

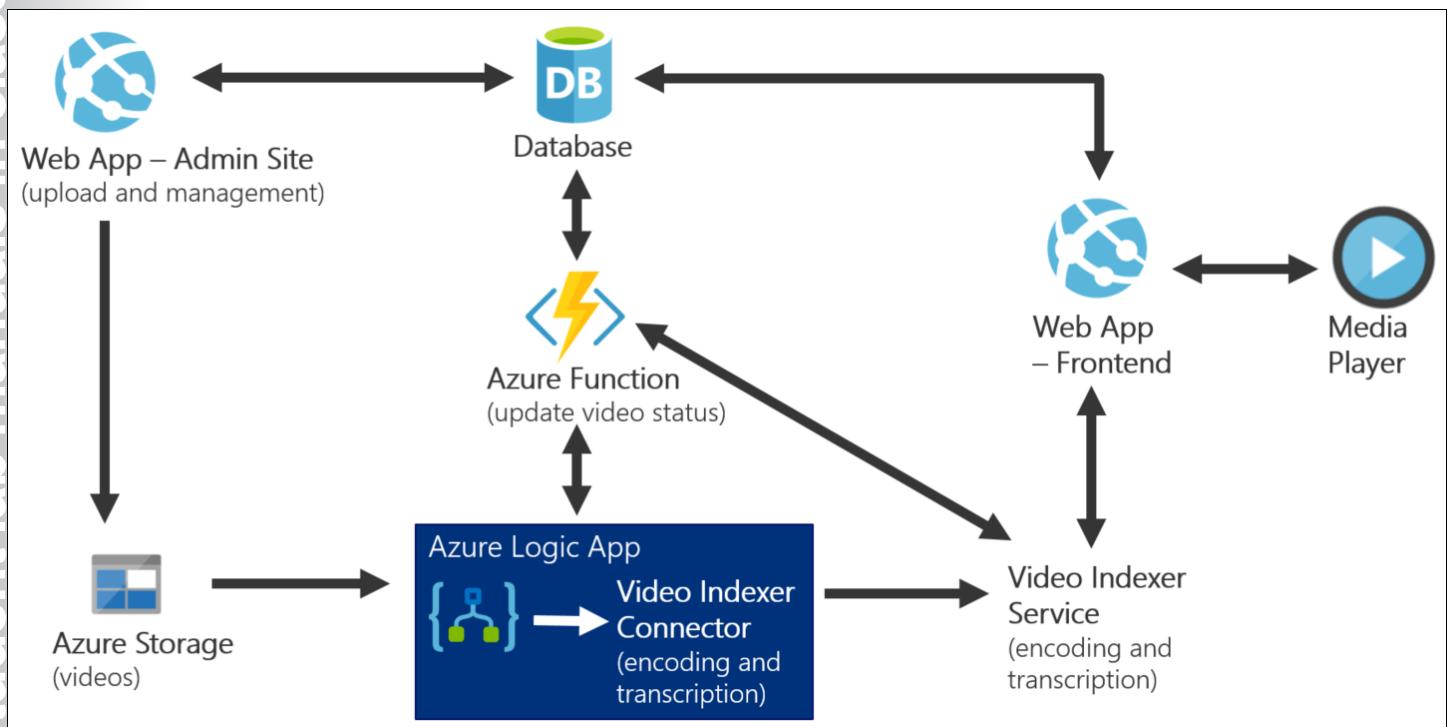
Students will be better able to build media applications including:

- Setup Video Indexer API
- Upload videos to Blob Storage to be encoded with Azure Video Indexer
- Integrate Video Indexer through Logic Apps and Azure Functions

Overview

Contoso has asked you to build a media streaming service so they can deliver their on-demand video training courses to their customers. For this solution, Contoso wants to use PaaS and Serverless services within the Microsoft Azure platform to help minimize development time and increase ease of maintenance. They would also like you to integrate automatic transcript generation and caption translation within the video encoding process so overall video production cost can be reduced, as well as improving the video player experience for their customers across the globe.

Solution architecture



Requirements

- Microsoft Azure subscription
- Local machine or Azure LABVM virtual machine configured with:
 - Visual Studio 2017 Community Edition or later

Before the hands-on lab

Duration: 30 minutes

In this lab, you will create a developer environment and download the required files for this course if you do not already have one that meets the requirements.

Prerequisites

1. Microsoft Azure subscription <http://azure.microsoft.com/en-us/pricing/free-trial/>
2. Client computer with Windows 7 or later with Visual Studio 2017

Task 1: Configure a development environment

If you do not have a machine setup with Visual Studio 2017 Community complete this task.

1. Create a virtual machine in Azure using the **Visual Studio Community 2017 (latest release) on Windows Server 2016 (x64)** image. This is important as you need to have Visual Studio 2017 version 15.4 or later to complete this lab.

| NAME | PUBLISHER | CATEGORY |
|--|-----------|----------|
| Visual Studio Community 2017 on Windows Server 2016 (x64) | Microsoft | Compute |
| Visual Studio Community 2017 (latest preview) on Windows Server 2016 (x64) | Microsoft | Compute |
| Visual Studio Community 2017 on Windows 10 Enterprise N (x64) | Microsoft | Compute |
| Visual Studio Community 2017 (latest release) on Windows Server 2016 (x64) | Microsoft | Compute |
| Visual Studio Community 2017 (latest release) on Windows 10 Enterprise N (x64) | Microsoft | Compute |

It is **highly** recommended to use a DS2 or D2 instance size for this VM.

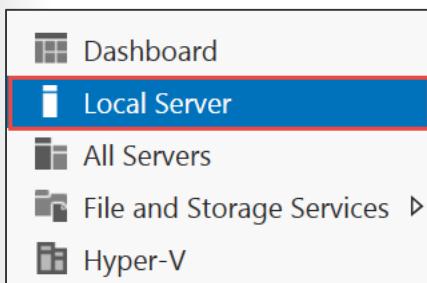
Task 2: Disable IE enhanced security

Note: Sometimes this image has IE ESC disabled, and sometimes it does not.

1. On the new VM, you just created click the Server Manager icon



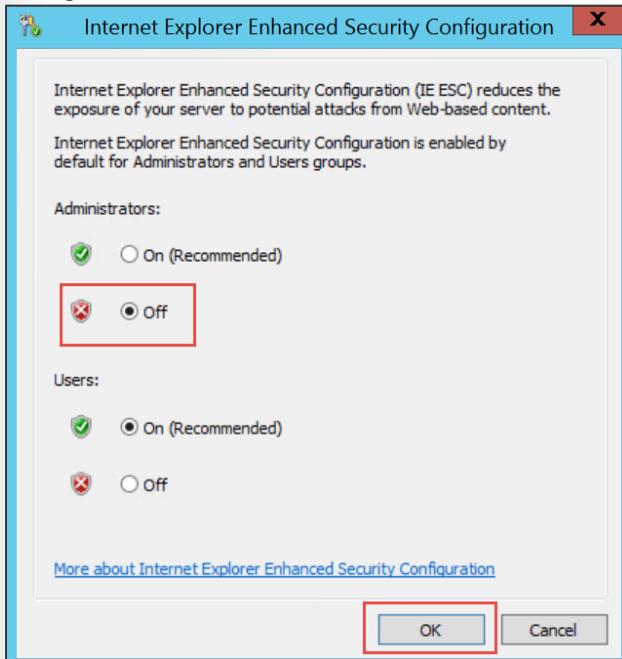
2. Click Local Server



3. On the right side of the pane, click **On** by IE Enhanced Security Configuration

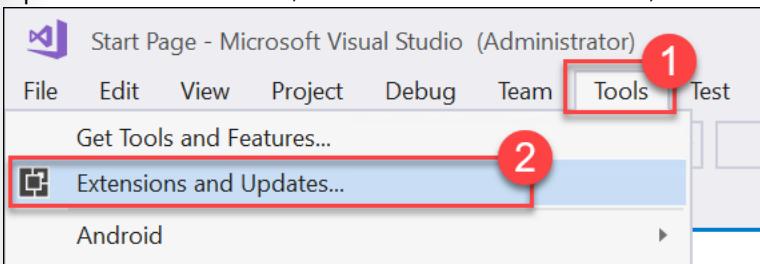


4. Change to **Off** for Administrators and click **OK**

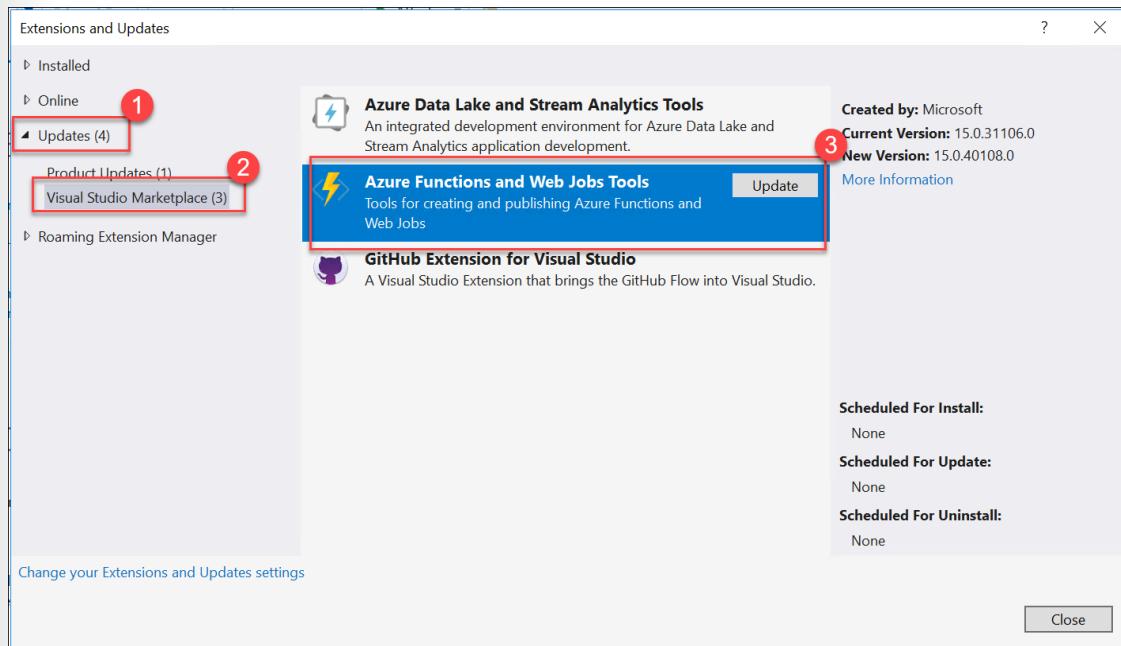


Task 3: Update Visual Studio Tools for Azure Functions

1. Open Visual Studio 2017, then click on the **Tools** menu, then click on **Extensions and Updates...**

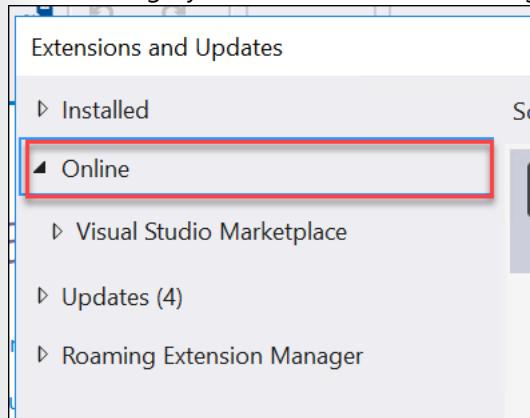


2. On the Extensions and Updates dialog, click on **Updates** then **Visual Studio Marketplace** on the left side of the dialog, then click on **Update** for the **Azure Functions and Web Jobs Tools** extension to update to the latest version.

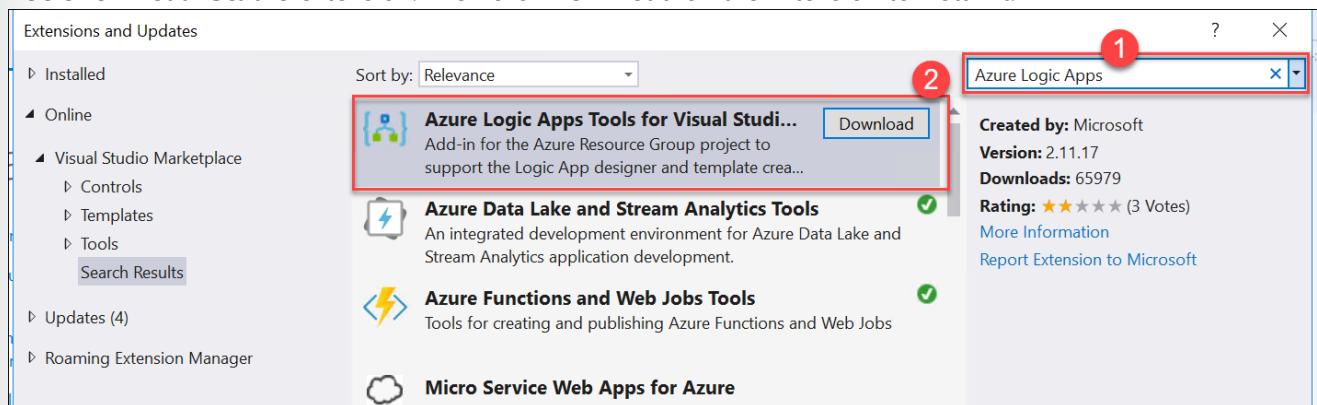


Task 4: Install Visual Studio Tools for Logic Apps

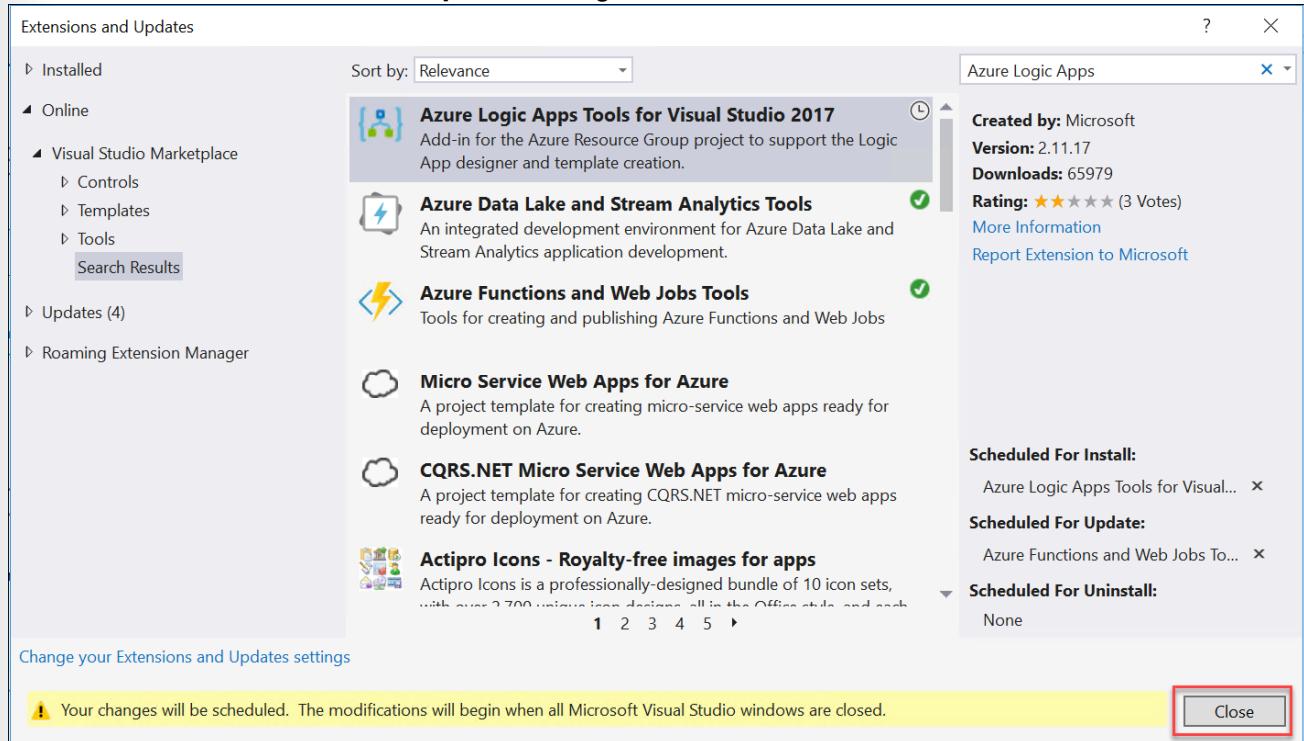
1. Within the **Extensions and Updates** dialog within Visual Studio 2017 sill open from the previous task, click on the **Online** category on the left side of the dialog



2. In the **Search** box in the upper right of the dialog, type in **Azure Logic Apps** to search for the **Azure Logic Apps Tools for Visual Studio** extension. Then click **Download** on the Extension to install it.

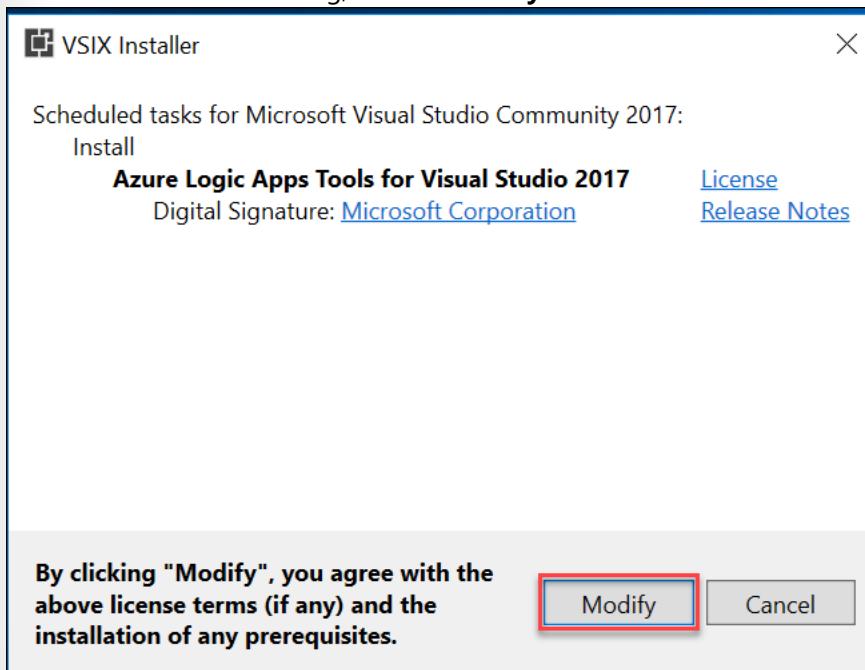


3. Click **Close** on the **Extensions and Updated** dialog

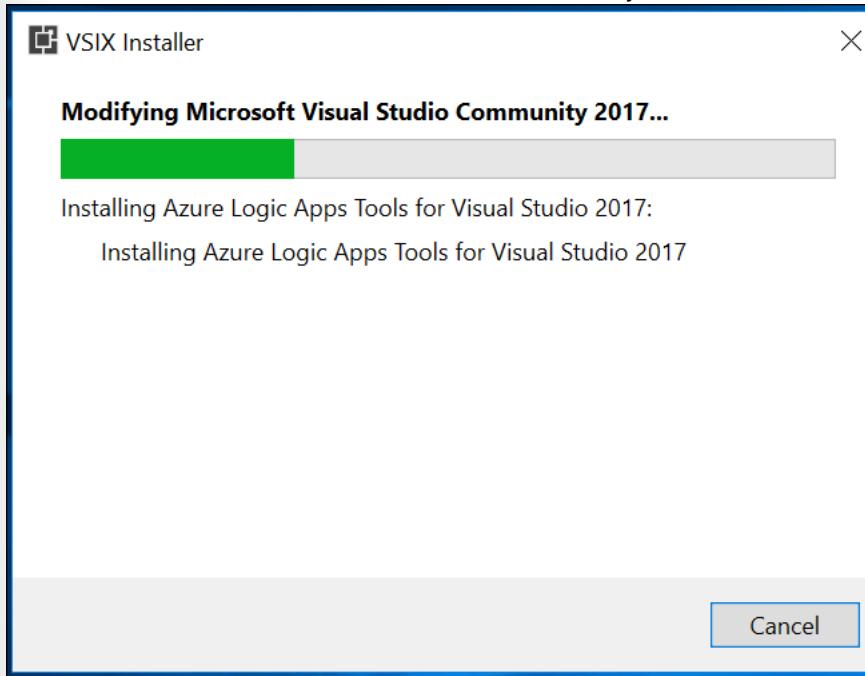


4. Close Visual Studio, then the pending installation of the Azure Logic Apps Tools for Visual Studio will automatically launch

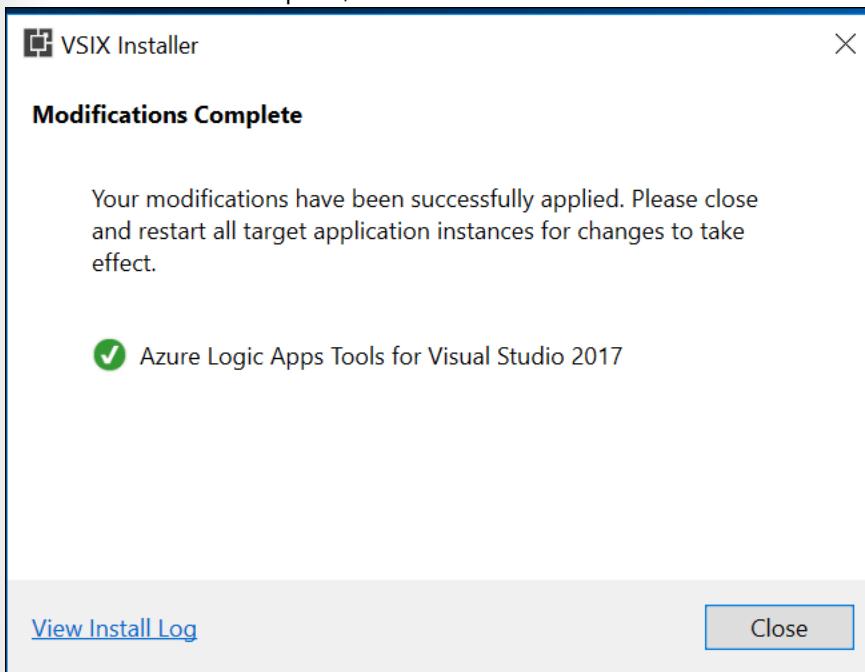
5. On the **VSIX Installer** dialog, click on **Modify**



6. Wait for the extension to be installed, this should only take about 1 minute



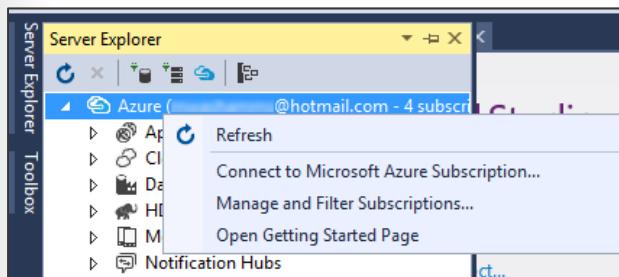
- Once installation is complete, click **Close**



- Restart Visual Studio now that the extension has been installed

Task 5: Validate connectivity to Azure

- From within the virtual machine, Launch Visual Studio 2017 and validate that you can login with your Microsoft Account when prompted
- Validate connectivity to your Azure subscription. Launch Visual Studio, open Server Explorer from the View menu, and ensure that you can connect to your Azure subscription.



Task 6: Download the exercise files

- Download the exercise files for the training (from within the virtual machine)
 - Create a new folder on your computer named **C:\Hackathon**
 - Download the support files (.zip format), <https://cloudworkshop.blob.core.windows.net/media-services-and-cdn/Media-Services-Student-Files.zip> to the new folder
 - Extract the contents to the same folder

Exercise 1: Signup for Video Indexer API Service

Duration: 15 minutes

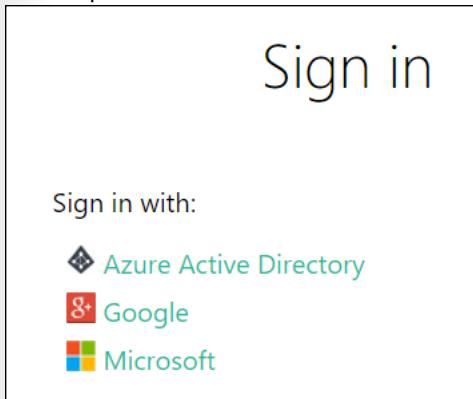
In this exercise, you will setup the Video Indexer API within Microsoft Azure.

Task 1: Signup for Video Indexer

1. Open a new browser window / tab and navigate to <https://videobreakdown.portal.azure-api.net/>
2. Click on the **SIGN IN** link in the upper-right



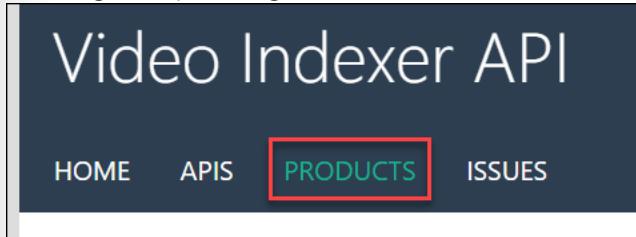
3. Click to sign in with your **Microsoft** or **Azure AD** credentials. Use the same credentials you login to your Azure Subscription with



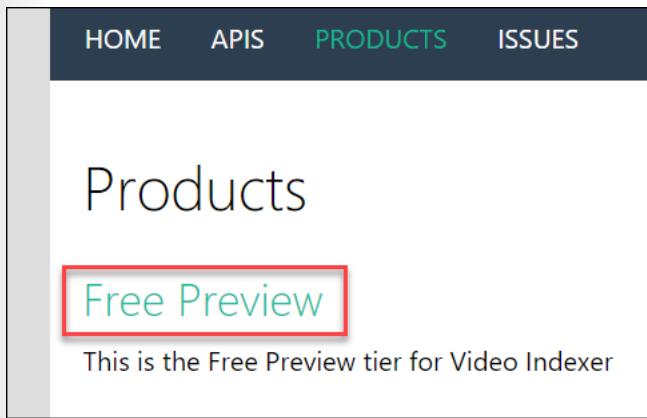
4. The first time you Sign in, you will be prompted to create a new account. Fill in the required fields, then click **Sign up**

The image shows a 'Sign up' form. At the top, it says 'Sign up' and 'Already a member? [Sign in now](#)'. Below that, it says 'Create a new API Management account' and 'To complete your registration fill out the form below.' There are four input fields: 'Email' (placeholder 'Email'), 'First name' (placeholder 'First name'), 'Last name' (placeholder 'Last name'), and a blue 'Sign up' button.

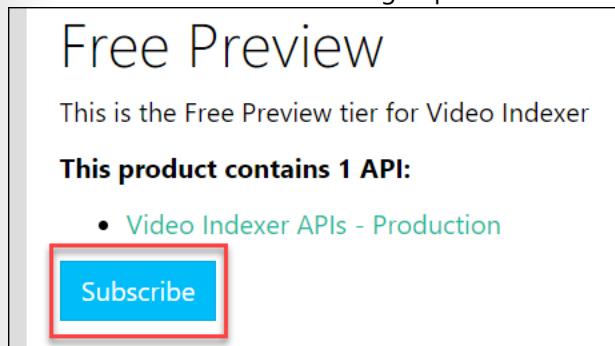
5. Once signed up and signed in, click on the **PRODUCTS** menu at the top



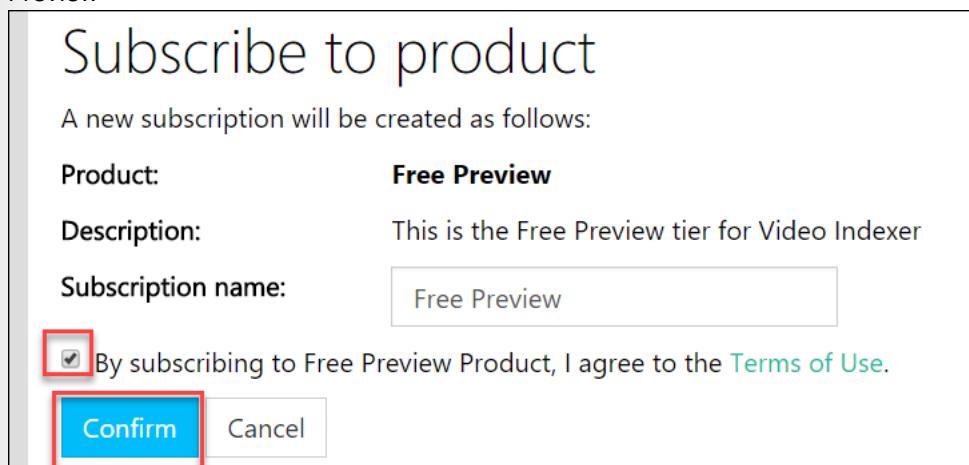
6. Click on the **Free Preview** Product link



7. Click the **Subscribe** button to sign up for the Free Preview



8. Check the "**I agree to the Terms of Use.**" Box, then click the **Confirm** button to finish signing up for the Free Preview



9. You will now see the **Free Preview** subscription has been created for your account

The screenshot shows the Video Indexer API profile page. At the top, there's a navigation bar with links for HOME, APIs, PRODUCTS, and ISSUES. Below the navigation, a message says: "You have successfully subscribed to the Free Preview product. Try the APIs in this product now." On the left, there's a "Profile" section with fields for Email, First name, and Last name. To the right, there's a "Your subscriptions" section. It shows a table with the following data:

| Subscription details | | Product | State | Action |
|----------------------|--|-------------------|--------------|--------|
| Subscription name | Free Preview | Rename | Free Preview | Active |
| Started on | 12/06/2017 | | | |
| Primary key | xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx | Show Regenerate | | |
| Secondary key | xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx | Show Regenerate | | |

Task 2: Copy Video Indexer API Key

1. While on the **Profile** page of the **Video Indexer API**, locate the **Subscription details** for the **Free Preview** subscription
2. Click the **Show** link next to the **Primary key** to reveal the API key

| Subscription details | | |
|----------------------|--------------------------------|-------------------|
| Subscription name | Free Preview | Rename |
| Started on | 12/06/2017 | |
| Primary key | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Show Regenerate |
| Secondary key | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Show Regenerate |

3. Copy the **Primary key** for the **Free Preview** subscription and save this for use later

| Subscription details | | |
|----------------------|----------------------------------|-------------------|
| Subscription name | Free Preview | Rename |
| Started on | 12/06/2017 | |
| Primary key | 41255766bb4b4f3491c3c48e7fe0d137 | Hide Regenerate |
| Secondary key | XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Show Regenerate |

Exercise 2: Setup video import workflow

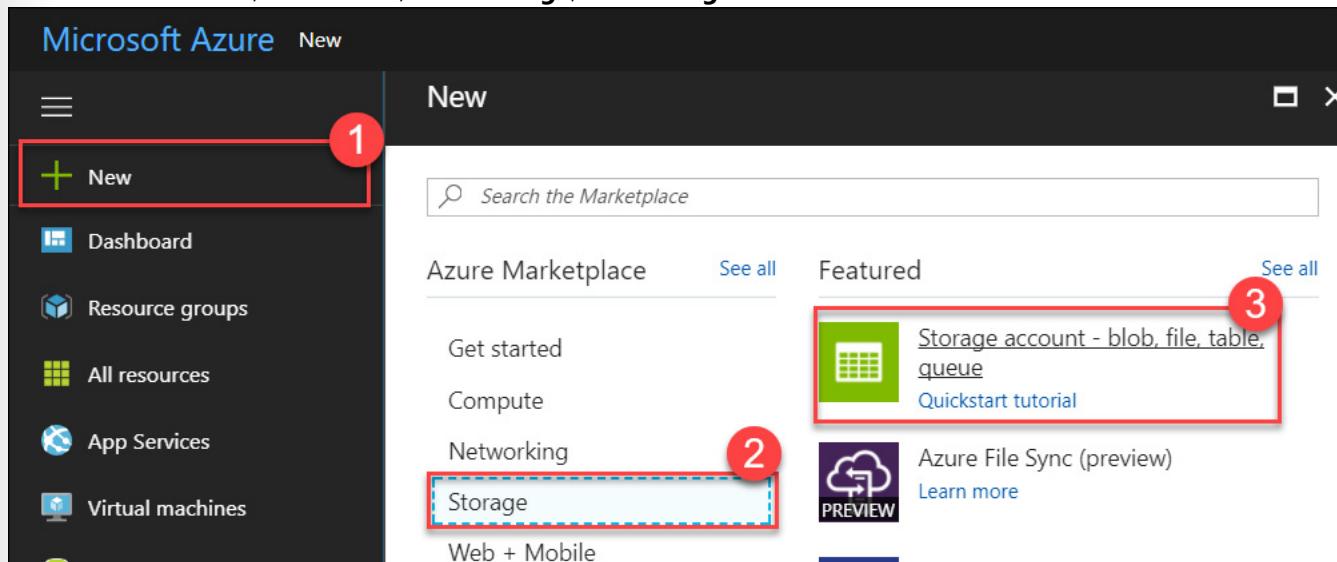
Duration: 20 minutes

In this exercise, you will set the import workflow for uploading and importing videos using the Video Indexer API.

Task 1: Create Storage Account for video files

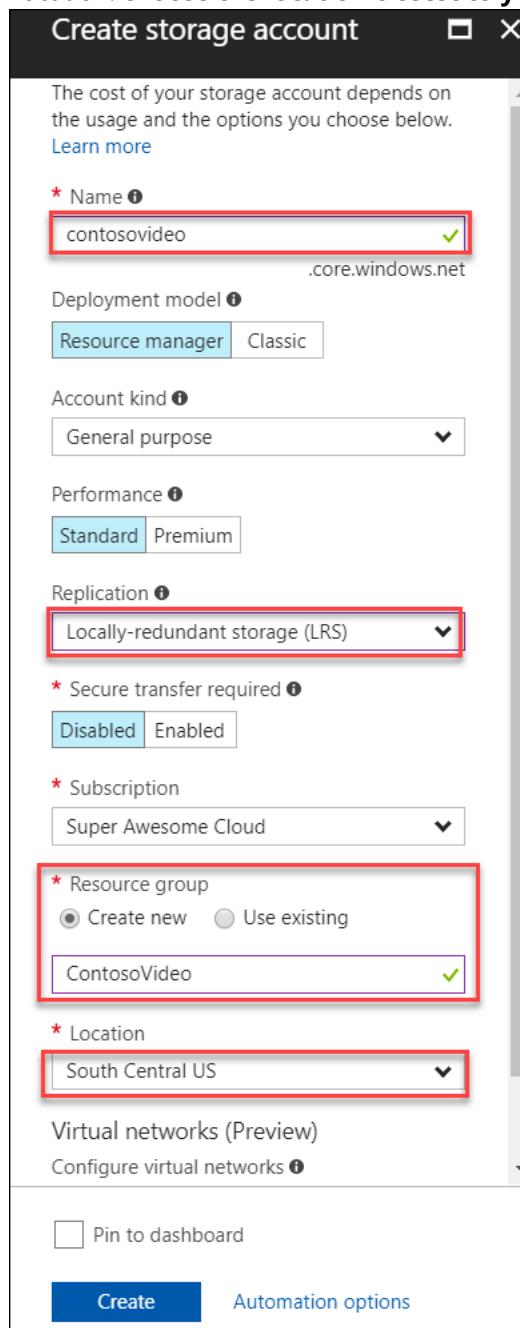
1. Open a browser window and login to the Azure Portal <http://portal.azure.com>

2. In the left-side menu, click **+New**, then **Storage**, and **Storage account**



3. On the **Create storage account** blade, enter the following values:
- Name: **enter a unique name**
 - Replication: **Locally-redundant storage (LRS)**
 - Resource Group: **ContosoVideo**

- d. Location: **choose the location closest to you**



4. Click **Create**

5. In the left-side menu, click **Resource groups**, then click on the **ContosoVideo** Resource group, then click the **Storage Account** that was just created

The screenshot shows the Microsoft Azure Resource Groups blade. On the left sidebar, 'Resource groups' is selected (marked with a red circle 1). In the main area, the 'ContosoVideo' resource group is listed (marked with a red circle 2). On the right side, under 'Subscription (change)', the 'contosovideo1' storage account is listed (marked with a red circle 3).

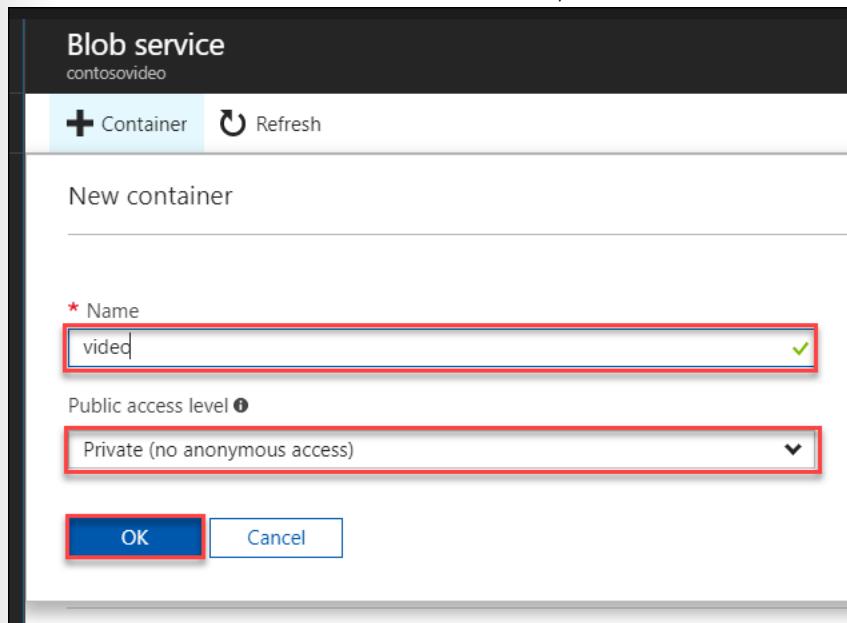
6. On the **Storage account** blade, click the **Blobs** link under Services to get started creating a new Blob Container

The screenshot shows the Storage account blade for 'contosovideo'. The left sidebar includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'SETTINGS' (with 'Access keys', 'Configuration', 'Shared access signature', 'Firewalls and virtual network...', and 'Metrics (preview)'), and 'Services'. Under 'Services', the 'Blobs' link is highlighted with a red box (marked with a red circle 1) and described as 'Object storage for understanding data'. Other services shown include 'Tables' (marked with a red circle 2).

7. Click the **+Container** button

The screenshot shows the Blob service blade for 'contosovideo'. At the top, it displays 'Blob service' and 'contosovideo'. Below that is a button labeled '+ Container' with a red box around it (marked with a red circle 1), and a 'Refresh' button. A search bar at the bottom is labeled 'Search containers by prefix'.

8. Enter **video** for the name of the new Container, leave the Public access level set to **Private**, then click **OK**



9. In the left pane of the Storage Account blade, click on **Access keys**, then copy the "primary" **key1**. Save this for reference later.

contosovideo1 - Access keys

Storage account

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Access keys

Configuration

Shared access signature

Firewalls and virtual networks

key1

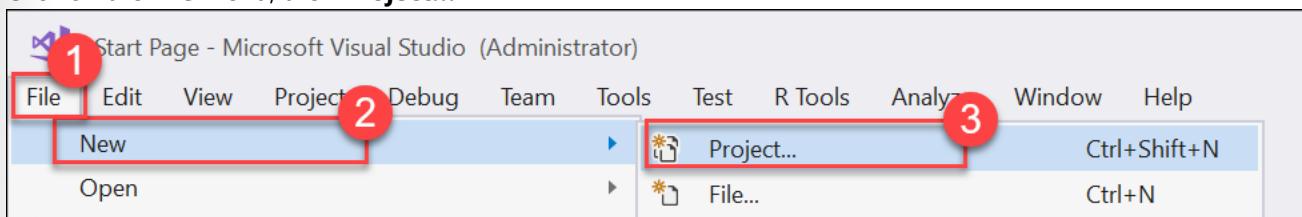
Key

Connection string

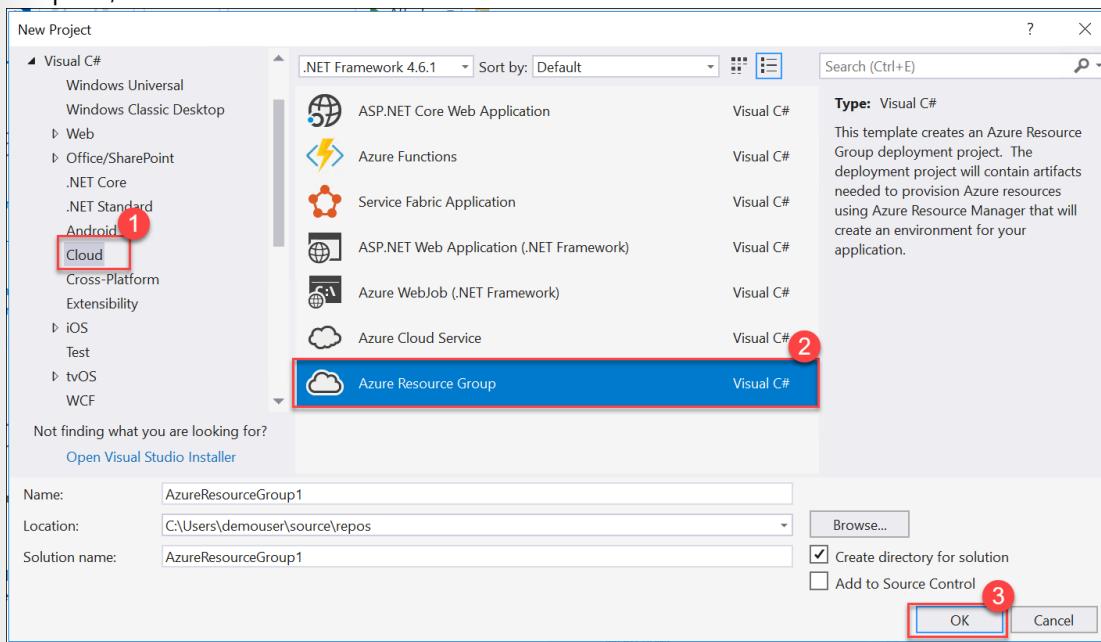
Task 2: Create Azure Logic App to process videos

1. Open **Visual Studio 2017**

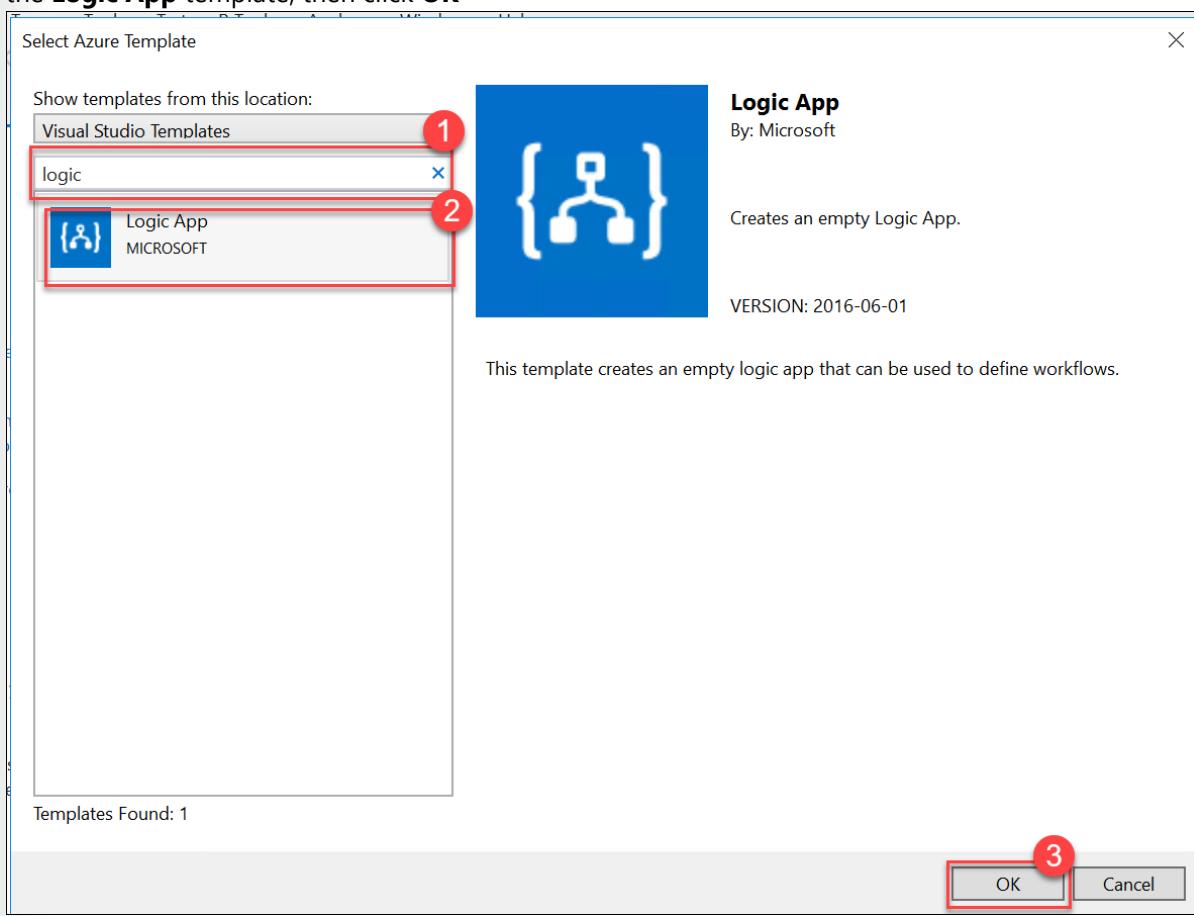
2. Click on the **File** menu, then **Project...**



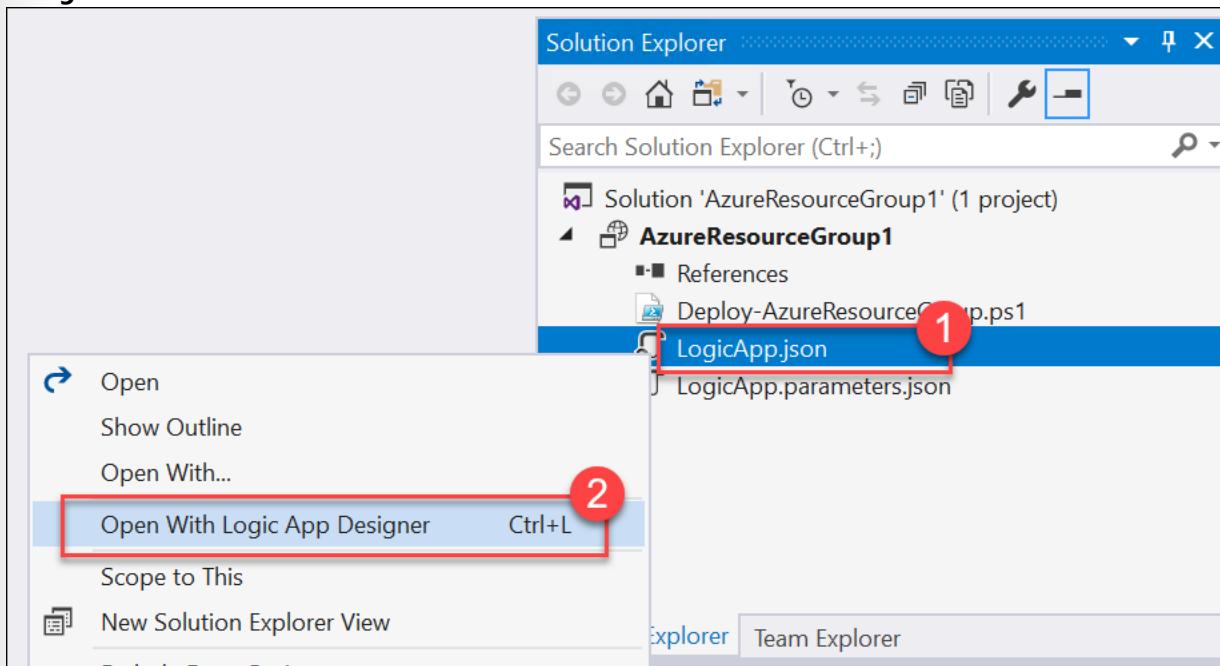
3. On the **New Project** dialog, select the **Cloud** category on the left, then select the **Azure Resource Group** project template, then click **OK**



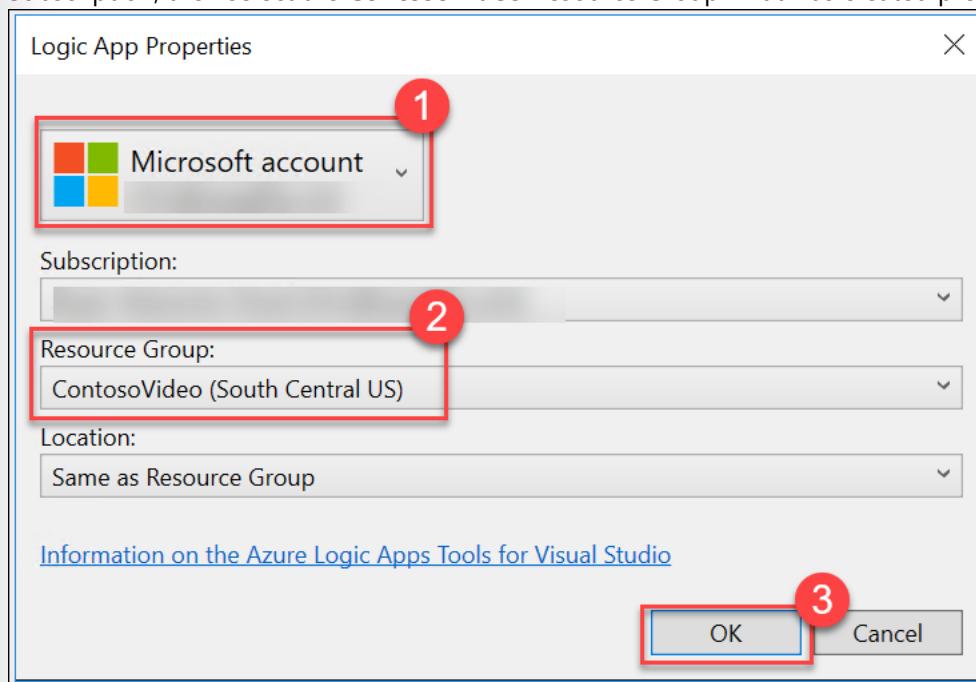
4. In the **Select Azure Template** dialog, type **Logic** into the Search box to filter the available templates, then select the **Logic App** template, then click **OK**



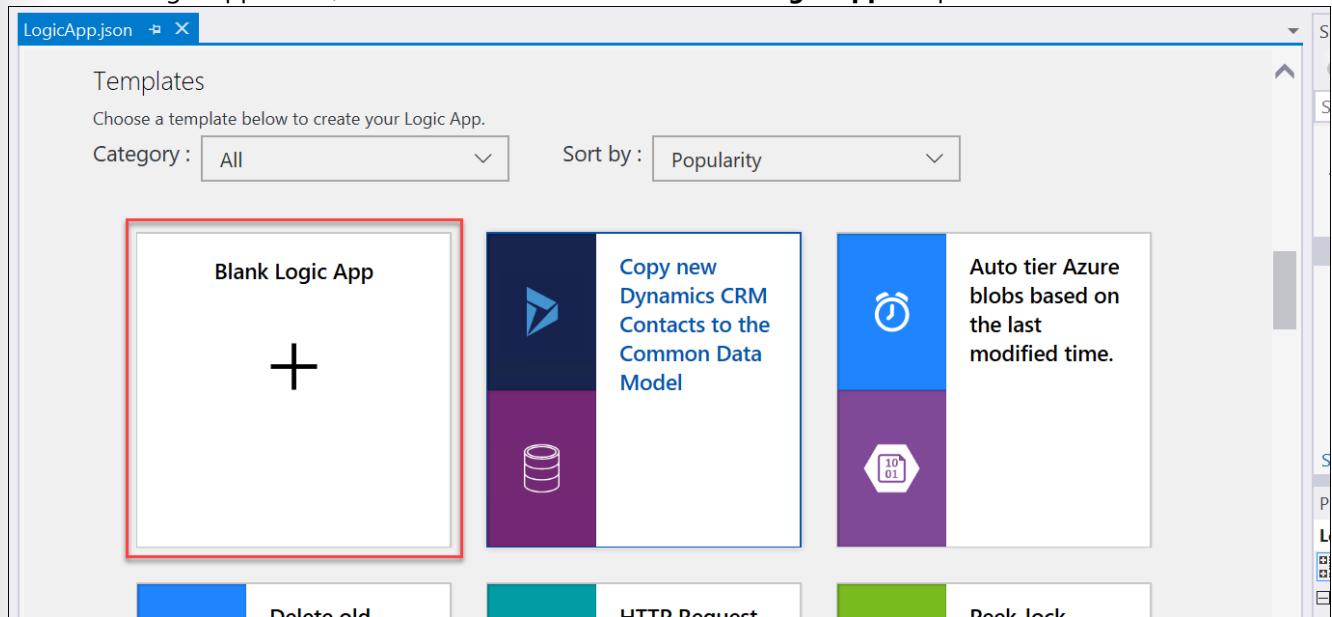
5. Within the **Solution Explorer** pane, right-click on the **LogicApp.json** file, then click on **Open With Logic App Designer**



6. On the **Logic App Properties** dialog, sign into your Microsoft or Organization Account associated with your Azure Subscription, then select the **ContosoVideo** Resource Group what was created previously, then click **OK**



7. Within the Logic App Editor, scroll down and click on the **Blank Logic App** template



8. In the **Logic App Designer**, start building out the Logic App Workflow by searching for and adding the **Azure Blob Storage – When a blob is added or modified (properties only)** Trigger

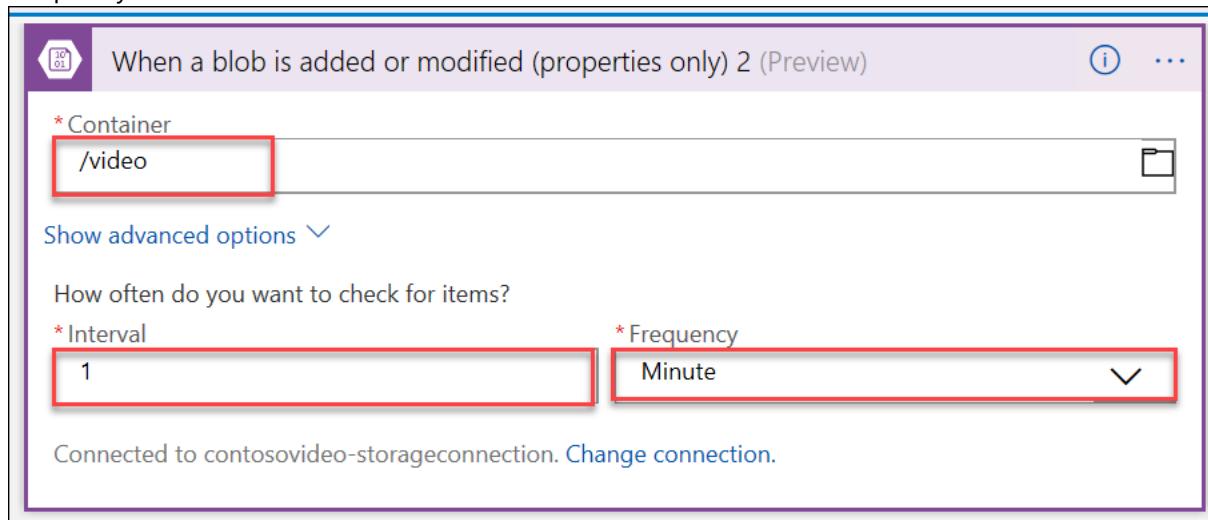
The screenshot shows the Logic App Designer interface. At the top, there is a search bar with the text "blob storage" and a magnifying glass icon. A red box with the number "1" highlights this search bar. Below the search bar, the "Connectors" section is visible, featuring the Azure Blob Storage connector icon. In the "Triggers (1)" tab, the "Azure Blob Storage When a blob is added or modified (properties only) Preview" trigger is selected, highlighted with a red box and the number "2". Below the trigger, there is a "TELL US WHAT YOU NEED" section and a "Help us decide which connectors and triggers to add next with UserVoice" button.

9. Enter a name in the **Connection Name** field, then click on the **Storage Account** that was previously created, then click **Create**

The screenshot shows the configuration page for the "When a blob is added or modified (properties only)" trigger. The "Connection Name" field is filled with "contosovideo-storageconnection" and is highlighted with a red box. Below it, the "Storage Account" section shows a table with one row, where the storage account name is "contosovideo1", the resource group is "ContosoVideo", and the location is "southcentralus". This row is also highlighted with a red box. At the bottom of the page is a blue "Create" button.

10. On the **Blob Storage Trigger**, enter the following values:

- Container: **video**
- Interval: **1**
- Frequency: **Minute**



11. Click the **+New step** button, then click **Add an action** to add another action to the workflow



12. Search for and add the **Azure Blob Storage – Create SAS URI by path** action

The screenshot shows the Microsoft Flow 'Choose an action' interface. At the top, there's a search bar with the text 'blob storage' entered. Below the search bar, under the heading 'Actions (13)', there is a list of actions categorized under 'Azure Blob Storage'. The actions listed are: 'Create blob', 'Create block blob' (with a 'Preview' link), 'Copy blob', 'Create SAS URI by path' (with a 'Preview' link), and another 'Create blob' action. The 'Create SAS URI by path' action is highlighted with a red box.

13. Click on the **Blob path** field, then choose to insert the **Path** parameter into the field from the options that display. This will use the Path of the Blob from the Trigger as the value to use when creating the SAS URI in this Action.

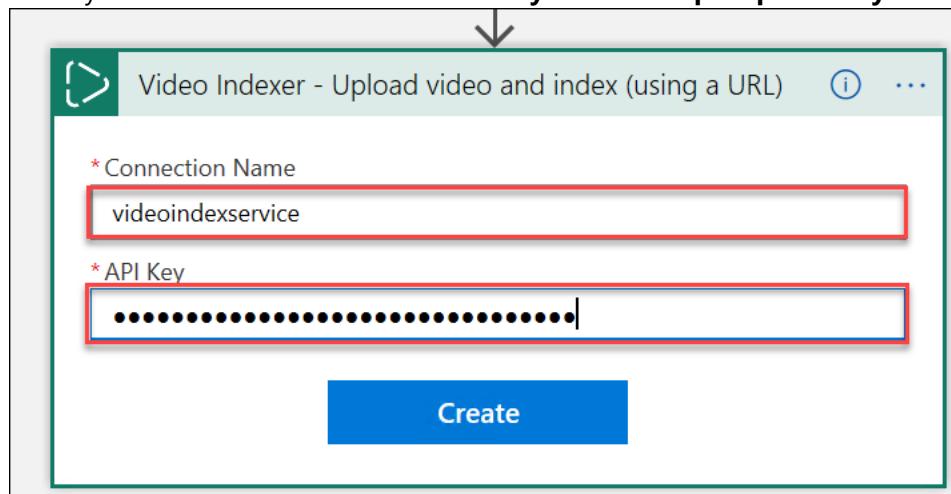
The screenshot shows the 'Create SAS URI by path (Preview)' action configuration. The 'Blob path' input field is highlighted with a red box. Below the input field, a list of parameters is shown, with 'Path' also highlighted with a red box. Other parameters listed include DisplayName, ETag, FileLocator, Id, IsFolder, LastModified, MediaType, Name, and Size.

14. Click the **+New step** button, and a new **Video Indexer – Upload video and index (using a URL)** Action

The screenshot shows the 'Choose an action' dialog. A search bar at the top contains 'upload video and index'. Below it, the 'Actions (3)' tab is selected. Three actions are listed: 'Video Indexer Delete video and its insights Preview', 'Video Indexer Upload video and index Preview', and 'Video Indexer Upload video and index (using a URL) Preview'. The third action is highlighted with a red box.

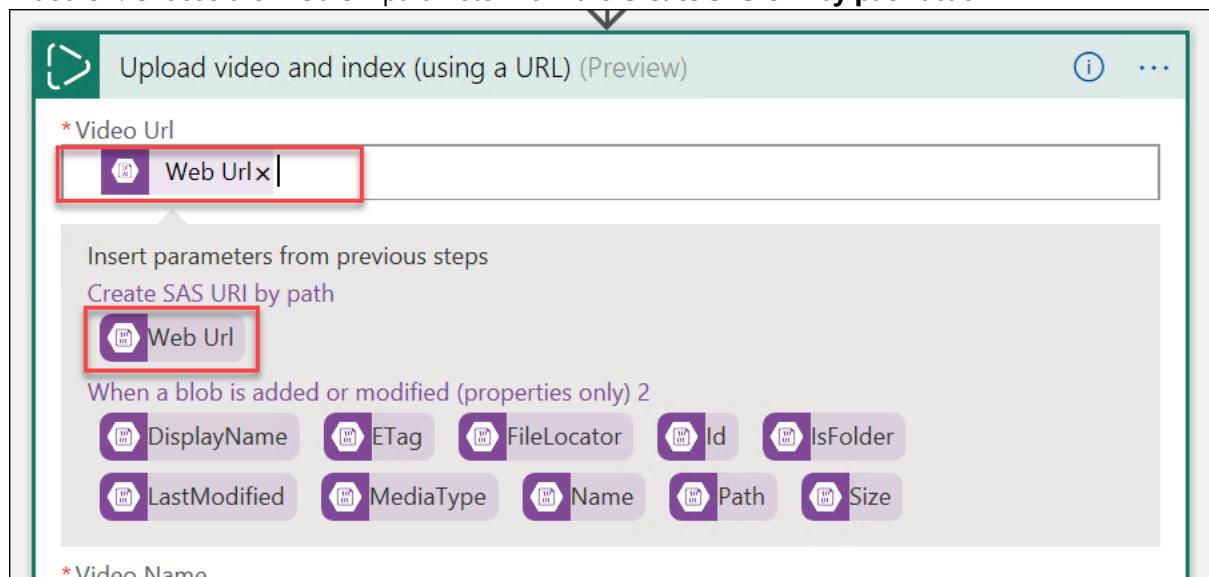
15. Enter the following values for the **Video Indexer – Upload video and index (using a URL)** connection, then click **Create**

- Connection Name: **enter a name for the connection**
- API Key: **Paste in the Video Indexer API Key that was copied previously**

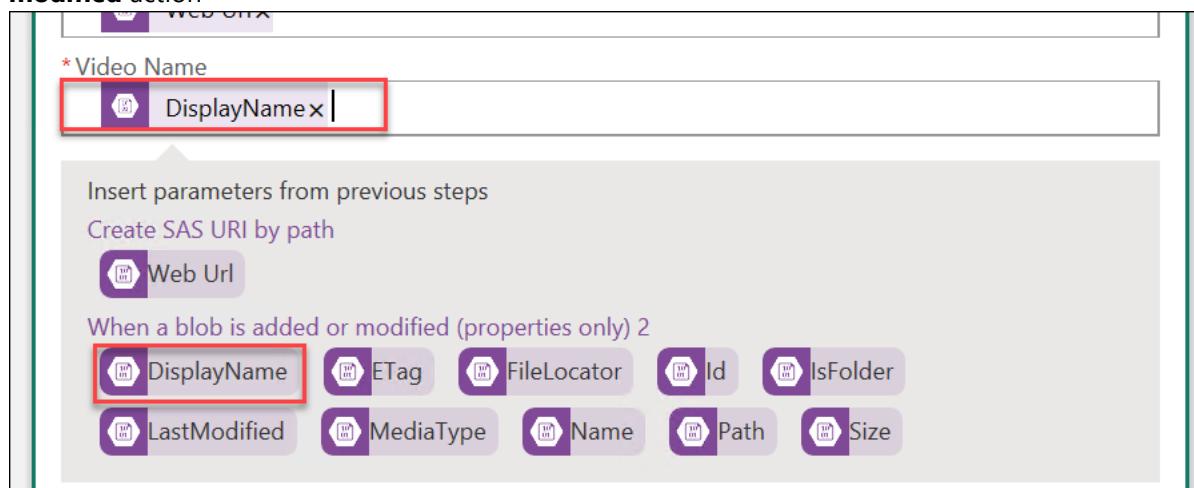


16. Enter the following values for the **Video Indexer – Upload video and index (using a URL)** action:

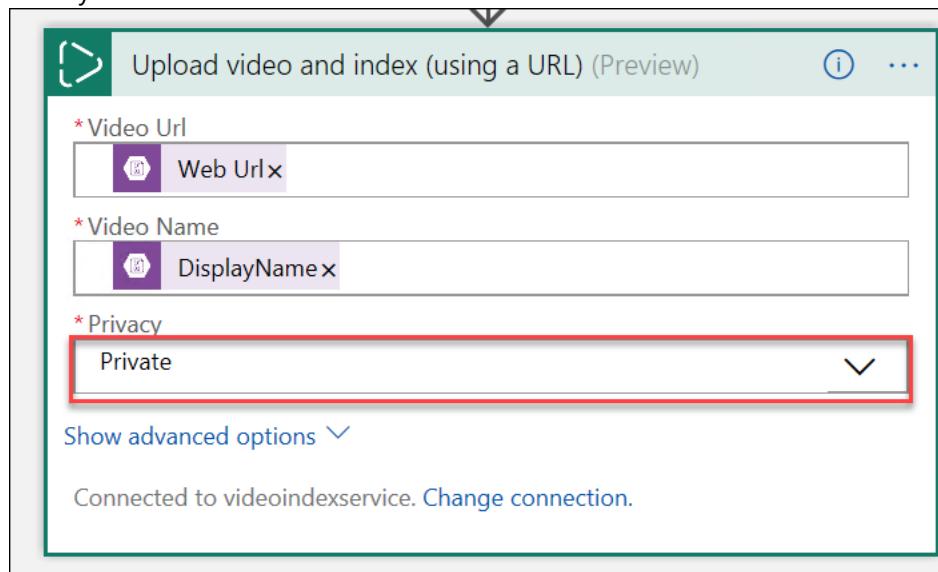
- Video Url: Choose the **Web Url** parameter from the **Create SAS URI by path** action



- b. Video Name: Choose the **DisplayName** parameter from the **When one or more blobs are added or modified** action



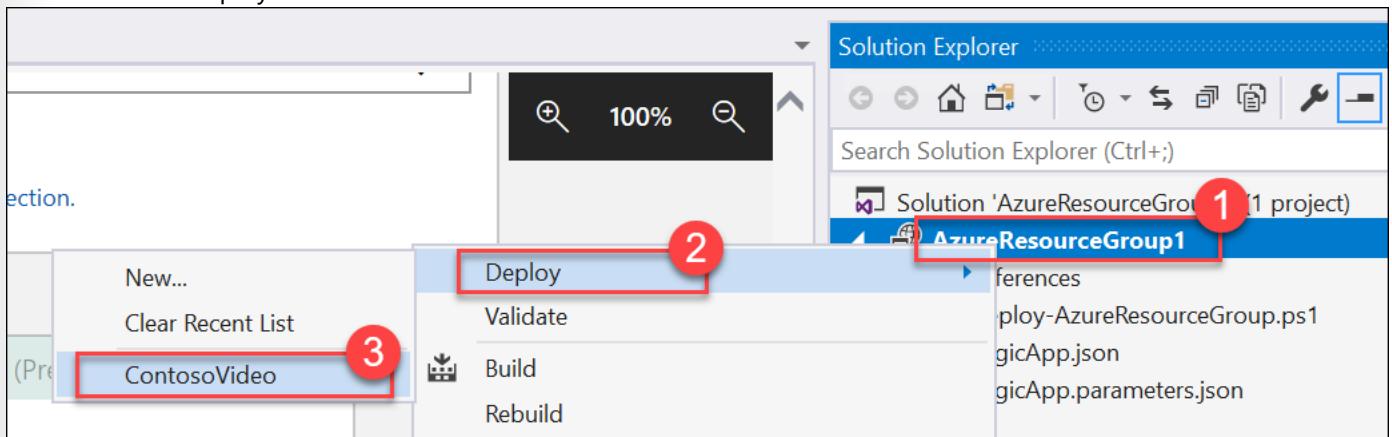
- c. Privacy: **Private**



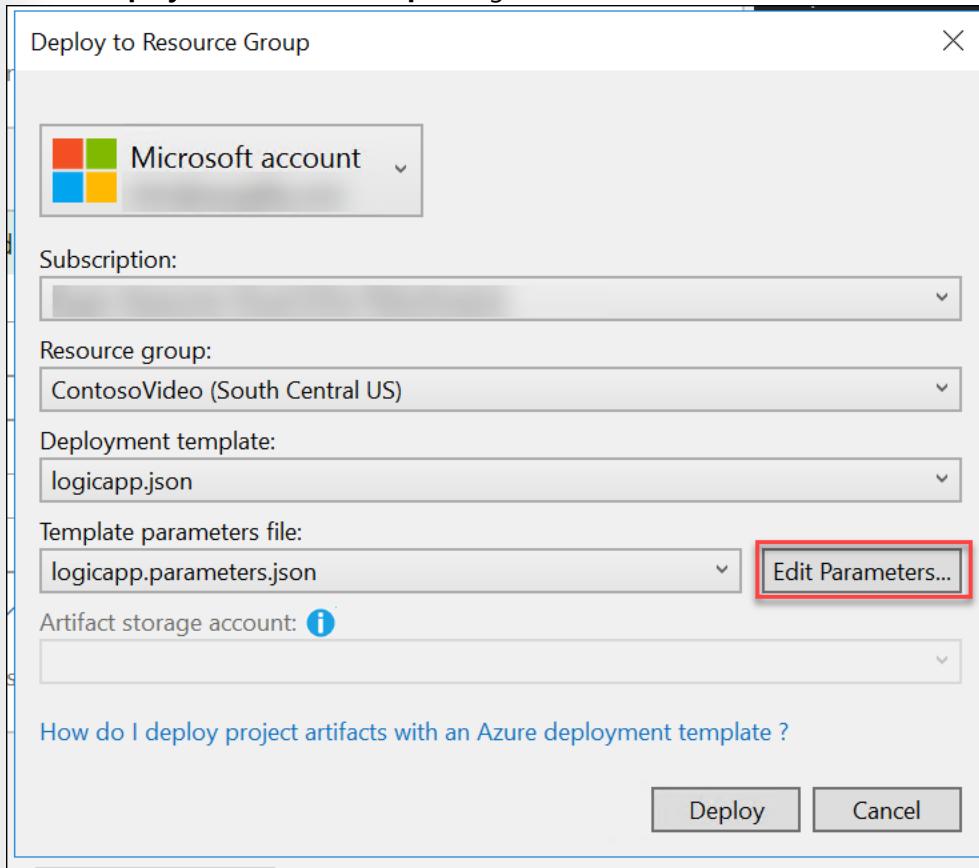
17. Save the Logic App



18. Within the **Solution Explorer** pane, right-click the **Resource Group Project**, click **Deploy**, then click on the **ContosoVideo** deployment



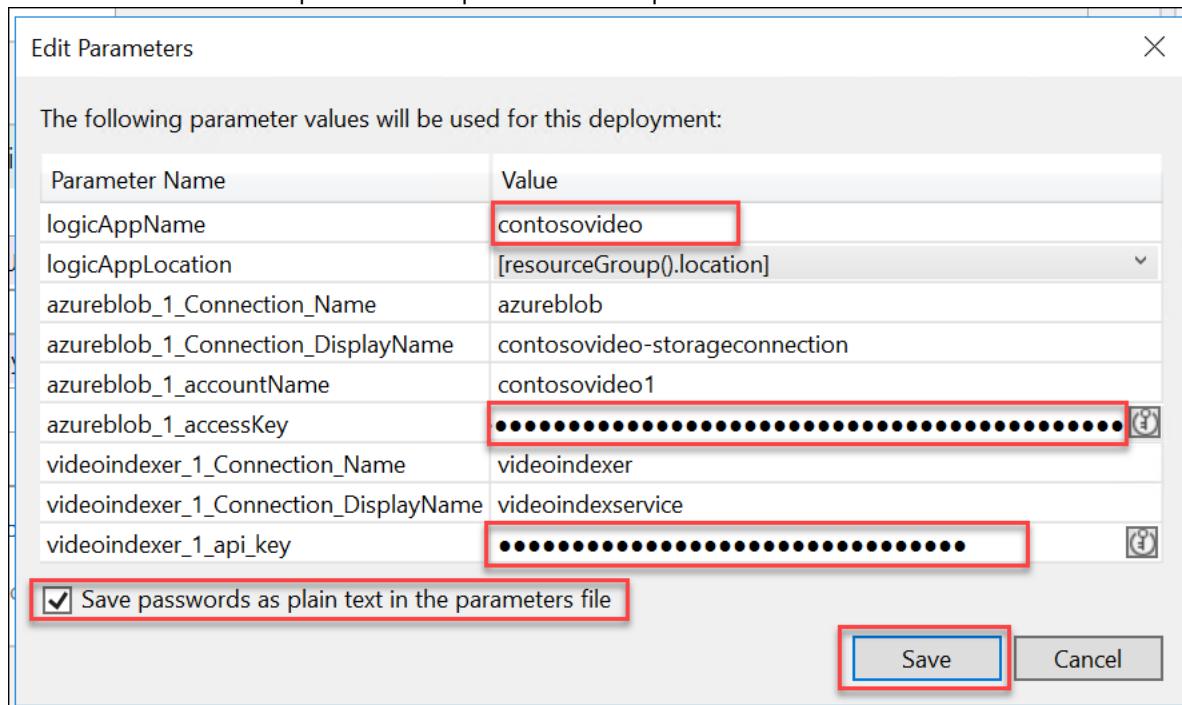
19. On the **Deploy to Resource Group** dialog, click **Edit Parameters...**



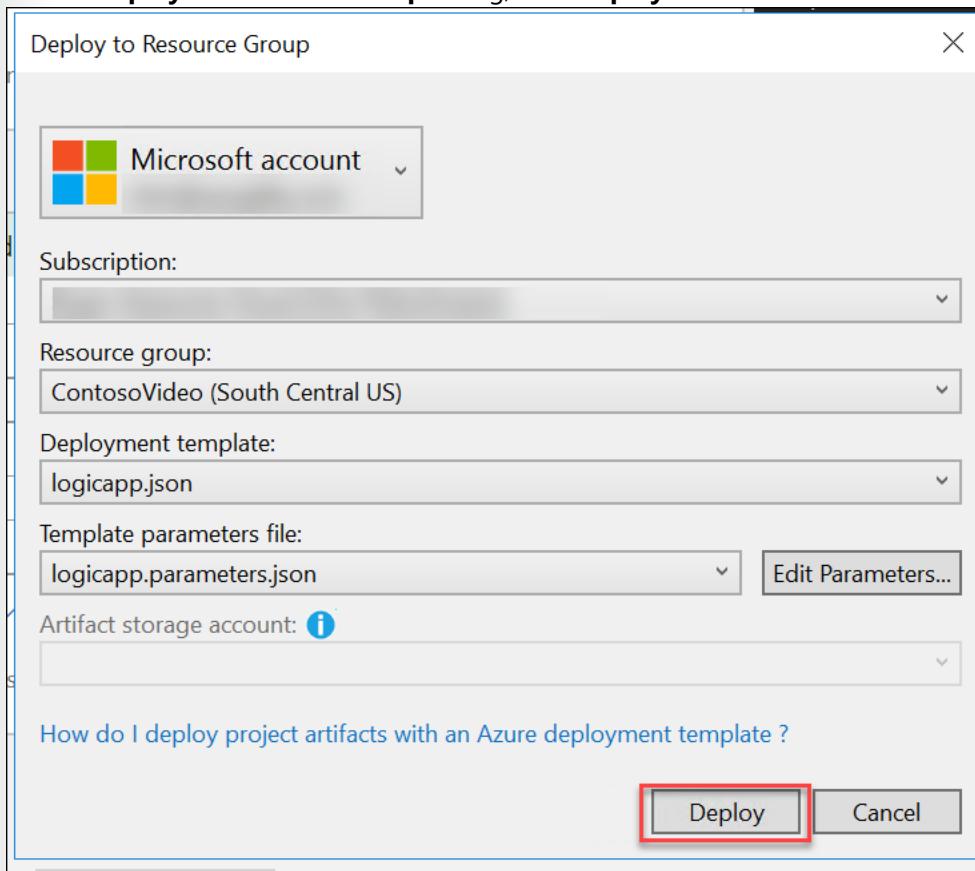
20. On the **Edit Parameters** dialog, enter the following values, then click **Save**:

- logicAppName: **enter a unique name for the Logic App**
- azureblob_1_accessKey: **paste in the Key 1 that was copied from the Storage Account previously**

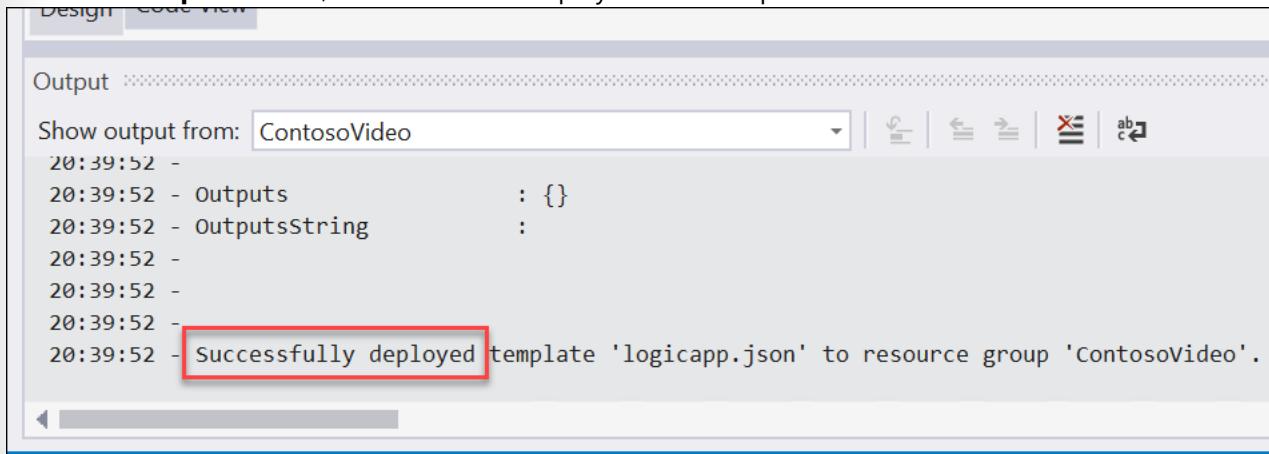
- c. videoindexer_1_api_key: **paste in the API Key for the Video Indexer service**
 - d. **Check** the box for "Save passwords as plain text in the parameters file"



21. On the **Deploy to Resource Group** dialog, click **Deploy**.



22. Watch the **Output** window, and wait for the deployment to complete



The screenshot shows the 'Output' window from a development environment. The window title is 'Output'. A dropdown menu at the top says 'Show output from: ContosoVideo'. Below the dropdown, there is a log of deployment messages. One message, 'Successfully deployed template 'logicapp.json' to resource group 'ContosoVideo''. is highlighted with a red rectangular box.

```
20:39:52 - 
20:39:52 - Outputs : {}
20:39:52 - OutputsString :
20:39:52 - 
20:39:52 - 
20:39:52 - 
20:39:52 - Successfully deployed template 'logicapp.json' to resource group 'ContosoVideo'.
```

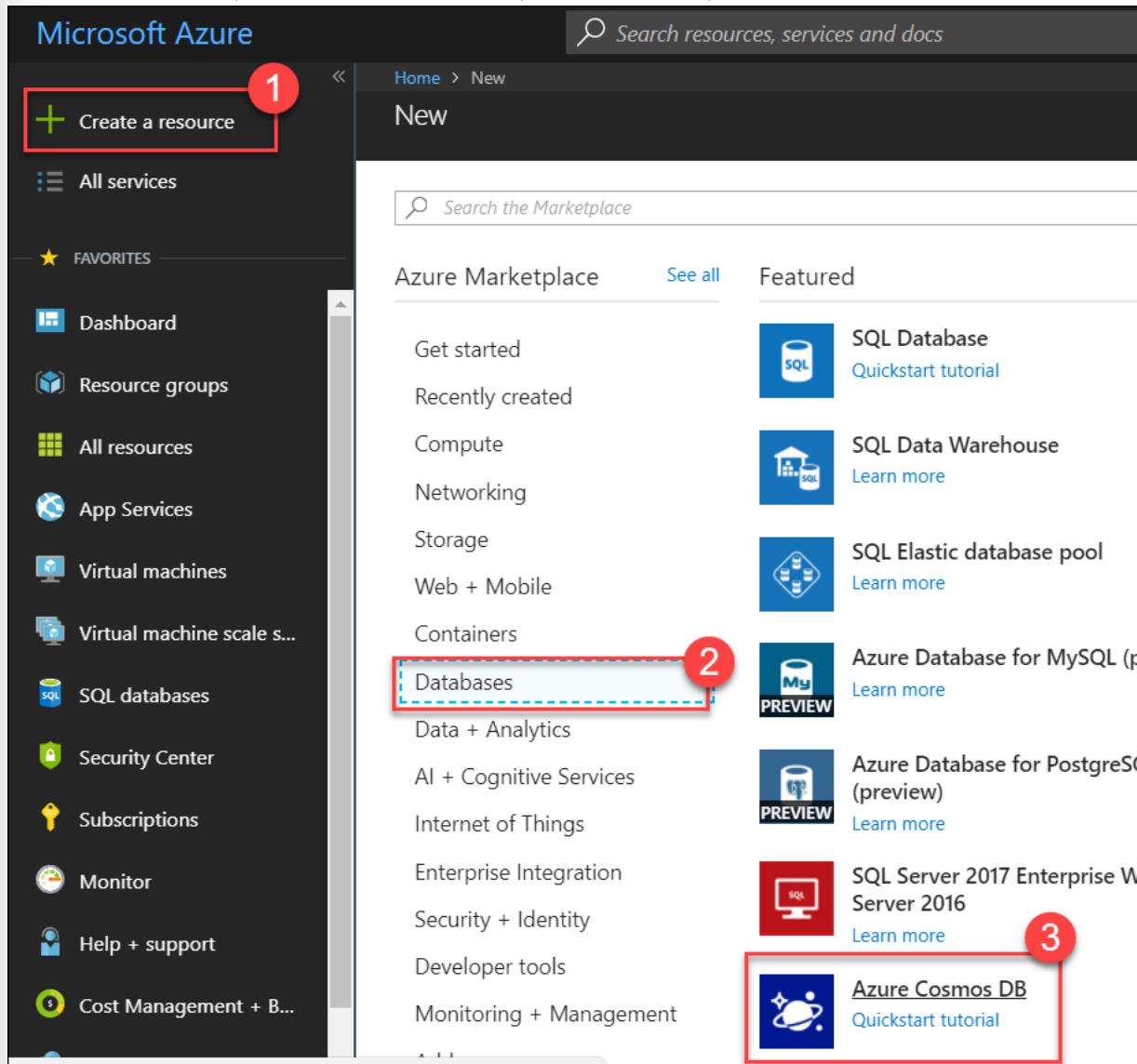
Exercise 3: Enable admin website to upload videos

Duration: 45 minutes

In this exercise, you will wire up the Admin website to enable Video Upload functionality.

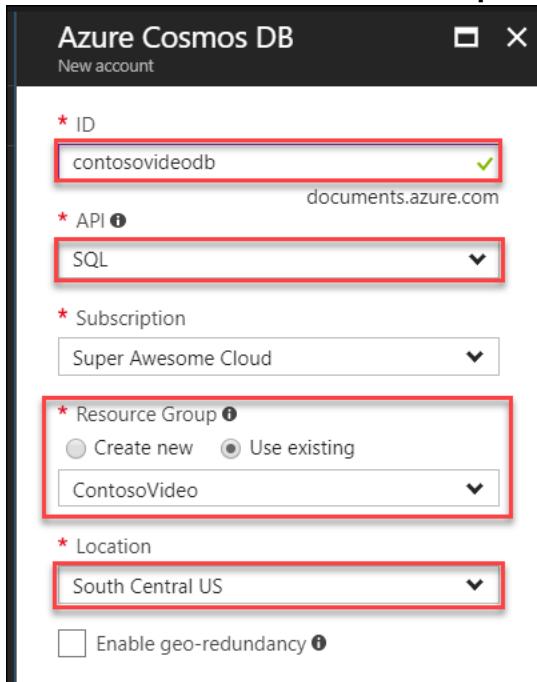
Task 1: Provision Cosmos DB Account

1. In the left-side menu, click **+Create a resource**, then **Databases**, then **Azure Cosmos DB**



2. On the **Azure Cosmos DB New account** blade, enter the following values:
 - a. ID: **enter a unique name**
 - b. API: **SQL**
 - c. Resource Group: **ContosoVideo**

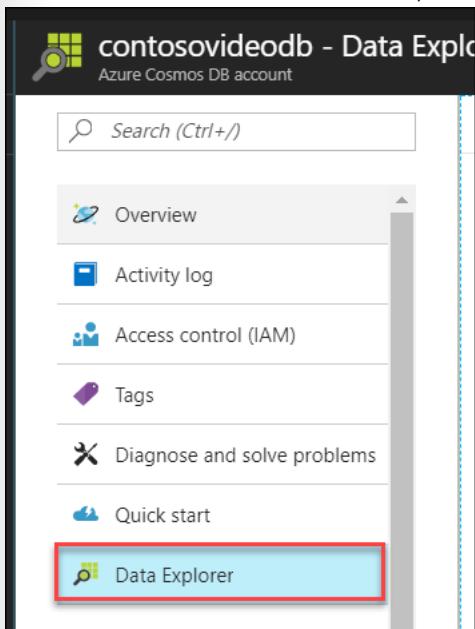
- d. Location: **choose the same location as previously**



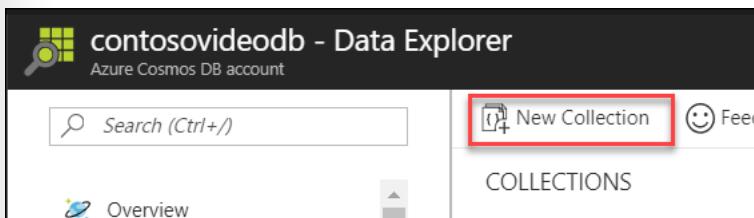
3. Click **Create**
4. In the left-side menu, click **Resource groups**, then click the **ContosoVideo** Resource group, then click on the **Cosmos DB Account** that was just created

The screenshot shows the Azure portal interface. On the left, the sidebar has a red circle labeled '1' over the 'Resource groups' link. The main area shows the 'Resource groups' blade for the 'ContosoVideo' resource group. A red circle labeled '2' is over the 'ContosoVideo' item in the list. The details pane on the right shows the 'Overview' tab selected, and a red circle labeled '3' is over the 'contosovideodb1' Cosmos DB account listed under '5 items'.

5. On the **Cosmos DB Account** blade, click on **Data Explorer**



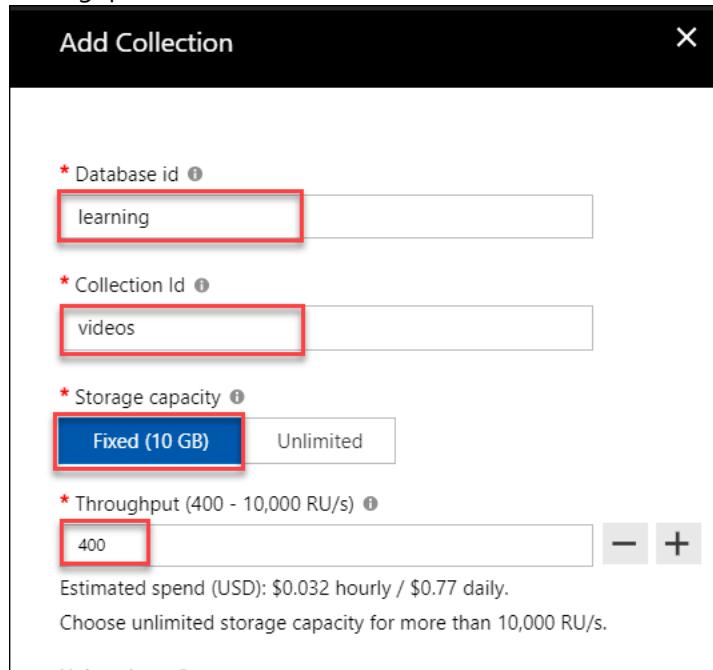
6. Click the **New Collection** button



7. On the **Add Collection** pane, enter the following values:

- Database id: **learning**
- Collection Id: **videos**
- Storage capacity: **Fixed (10GB)**

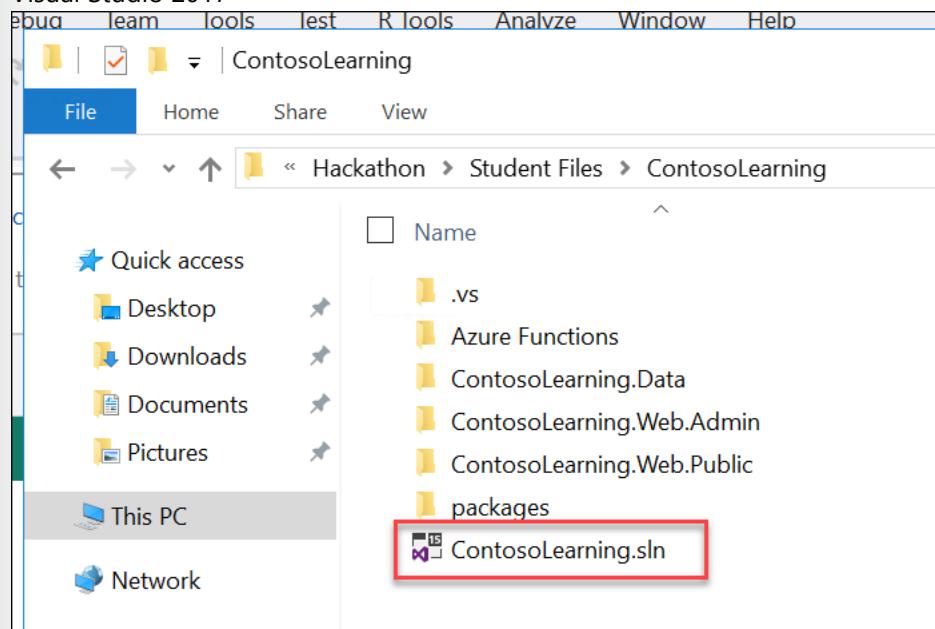
- d. Throughput: 400



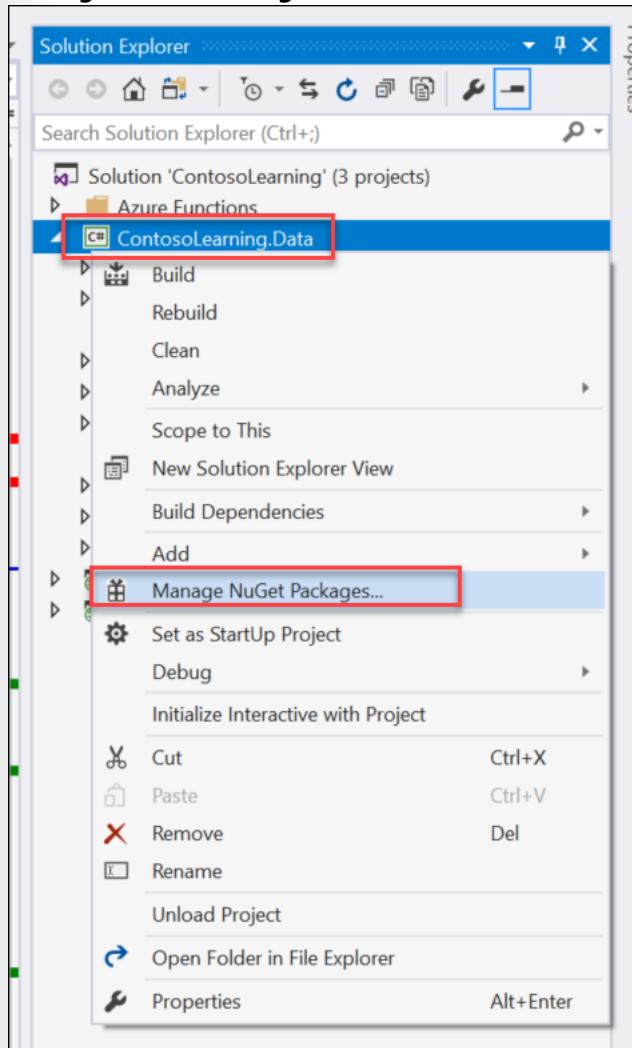
8. Click **OK**

Task 2: Integrate Cosmos DB into Admin Web App

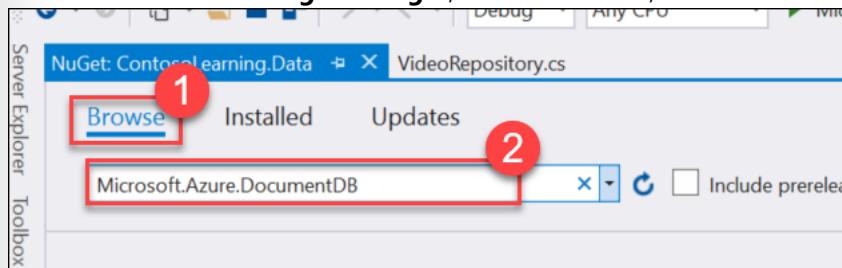
1. Within the folder where the exercise files have been extracted to, Open the **ContosoLearning.sln** solution within Visual Studio 2017



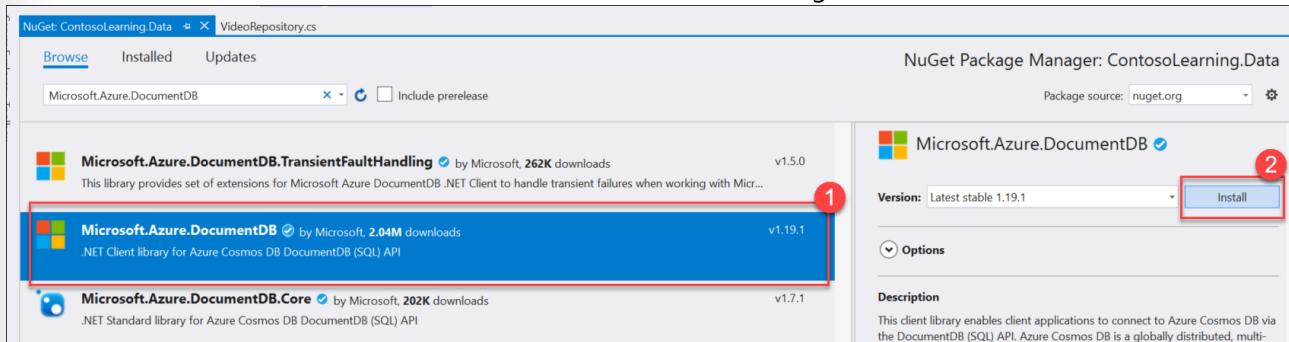
2. Within Solution Explorer, locate the **ContosoLearning.Data** project, then right-click the project and click on **Manage NuGet Packages...**



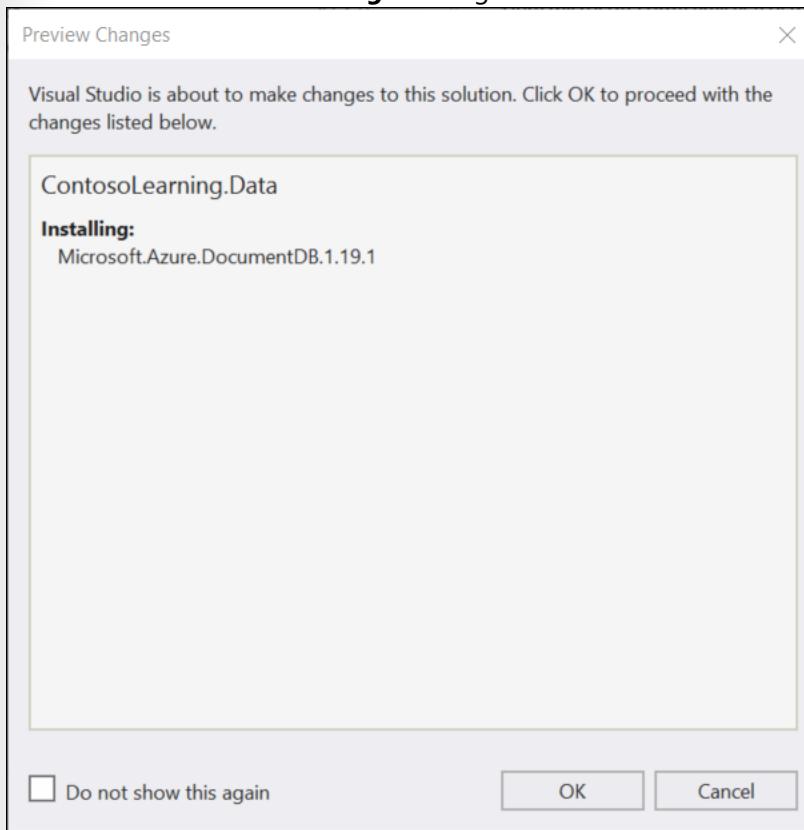
3. Within the **NuGet Package Manager**, click on **Browse**, then search for **Microsoft.Azure.DocumentDB**



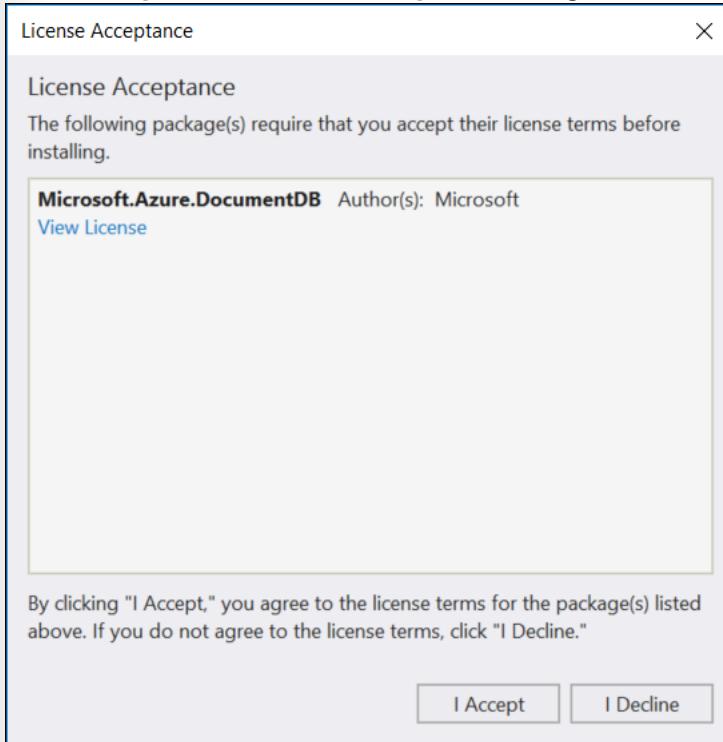
4. Click on and select the **Microsoft.Azure.DocumentDB** NuGet Package, then click **Install**



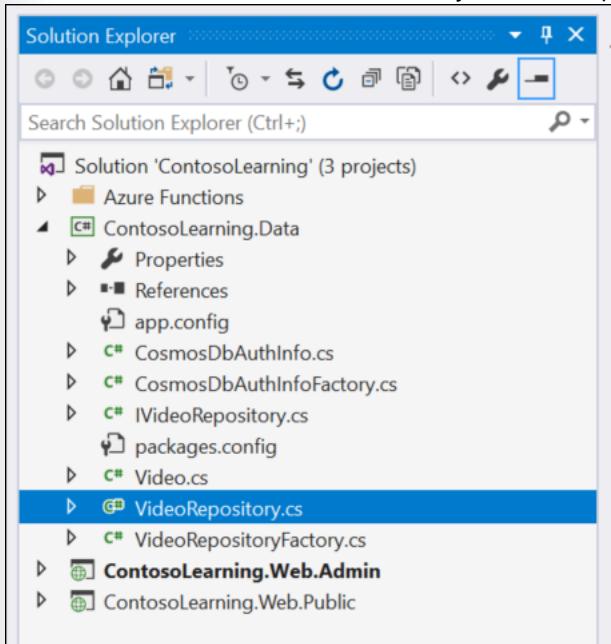
5. Click **OK** on the **Preview Changes** dialog



6. Click **I Accept** on the **License Acceptance** dialog



7. Within Solution Explorer, locate the **ContosoLearning.Data** project, and open the **VideoRepository.cs** file. This is the source code file that will contain the source code for interacting with the Cosmos DB Database / Collection within this solution. This data access layer (DAL) implementation uses a simplified Repository pattern.



8. Add the following **using** statements to the top of the **VideoRepository.cs** file

```
using Microsoft.Azure.Documents.Client;  
using Microsoft.Azure.Documents.Linq;
```

9. Add the following method named **createDocumentClient()** to the **VideoRepository** class. This is a reusable method that will be used by a couple different methods within the class to instantiate a new DocumentClient class instance for working with Cosmos DB.

```
private DocumentClient createDocumentClient()
{
    return new Microsoft.Azure.Documents.Client.DocumentClient(
        new Uri(_cosmosDbAuthInfo.Endpoint),
        _cosmosDbAuthInfo.AuthKey
    );
}
```

10. Modify the code for the **GetAll()** method to the following to load all Documents from Cosmos DB and return them as a List object

```
var list = new List<Video>();

using (var documentClient = this.createDocumentClient())
{
    var documentQuery = documentClient.CreateDocumentQuery<Video>(
        UriFactory.CreateDocumentCollectionUri(_cosmosDbAuthInfo.Database,
        _cosmosDbAuthInfo.Collection)
        .AsDocumentQuery();

    while (documentQuery.HasMoreResults)
    {
        list.AddRange(
            await documentQuery.ExecuteNextAsync<Video>()
        );
    }
}

return list;
```

11. Notice that the above code references a **_cosmosDbAuthInfo** object. This object is prepopulated with the necessary values for the Cosmos DB Account and Collection by loading from **AppSettings**. These AppSettings will be populated later.

12. Replace the body of the **Get(string id)** method with the following code that will retrieve a single Document by ID

```
using (var documentClient = this.createDocumentClient())
{
    var response = await documentClient.ReadDocumentAsync<Video>(
        UriFactory.CreateDocumentUri(_cosmosDbAuthInfo.Database, _cosmosDbAuthInfo.Collection, id)
    );

    return response.Document;
}
```

13. Replace the body of the **Delete(string videoId)** method with the following code that will delete a single Document by ID

```
using (var documentClient = this.createDocumentClient())
{
    await documentClient.DeleteDocumentAsync(
        UriFactory.CreateDocumentUri(_cosmosDbAuthInfo.Database, _cosmosDbAuthInfo.Collection,
videoId)
    );
}
```

14. Add the following code to the **Insert(Video video)** method in place of the "*// Code Here*" comment. This code will insert a new Document into the Cosmos DB Collection

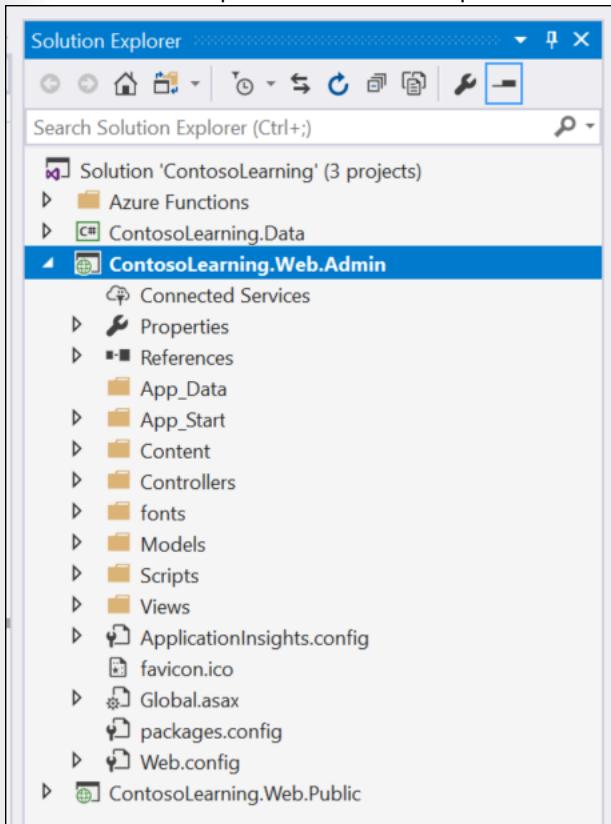
```
using (var documentClient = this.createDocumentClient())
{
    await documentClient.CreateDocumentAsync(
        UriFactory.CreateDocumentCollectionUri(_cosmosDbAuthInfo.Database,
_cosmosDbAuthInfo.Collection),
        Video
    );
}
```

15. Save the file

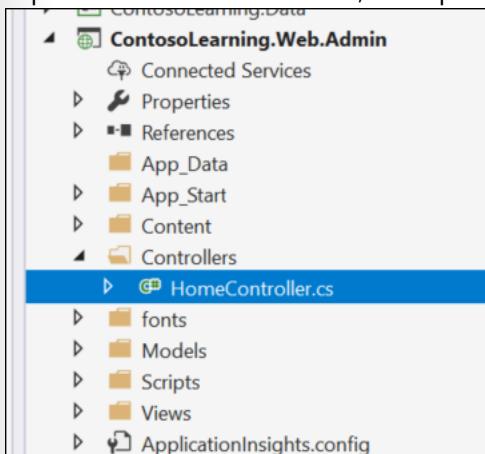
Task 3: Integrate File Upload into Admin Web App

1. Open the **ContosoLearning.sln** solution within Visual Studio 2017

2. Within Solution Explorer, locate and expand the **ContosoLearning.Web.Admin** project



3. Expand the **Controllers** folder, and open the **HomeController.cs** file



4. Add the following using statements to the top of the **HomeController.cs** file

```
using Microsoft.WindowsAzure.Storage;
using System.Configuration;
```

5. Within the **HomeController**, locate the **Upload** method with the **HttpPost** attribute

```
[HttpPost]
public async Task<ActionResult> Upload(HttpPostedFileBase file, HomeUploadModel model)
{
    if (!ModelState.IsValid)
    {
        return View(model);
    }

    // Generate a unique id for the uploaded video
    var videoId = Guid.NewGuid().ToString();

    // =====
    // Upload Video File to Blob Storage
    // =====

    // Code Here

    // =====
    // Save Video info to Cosmos DB Database / Collection
    // =====

    // Code Here

    return RedirectToAction("Index");
}
```

6. Paste in the following code where the comment states "*Upload Video File to Blob Storage*". This code will take the file uploaded to the Action Method via an HTTP Post, and upload that file to the Blob Storage Account.

```
// Load Connection String to Azure Storage Account
var videoConnString = ConfigurationManager.ConnectionStrings["videostorage"].ConnectionString;
if (string.IsNullOrWhiteSpace(videoConnString))
{
    throw new Exception("The 'videostorage' Connection String is NOT set");
}

// Get reference to the Blob Container to upload to
var storageAccount = CloudStorageAccount.Parse(videoConnString);
var blobClient = storageAccount.CreateCloudBlobClient();
var container = blobClient.GetContainerReference("video");
await container.CreateIfNotExistsAsync();

// Upload Video file to Blob Storage
var blob = container.GetBlockBlobReference(videoId);
await blob.UploadFromStreamAsync(file.InputStream);
```

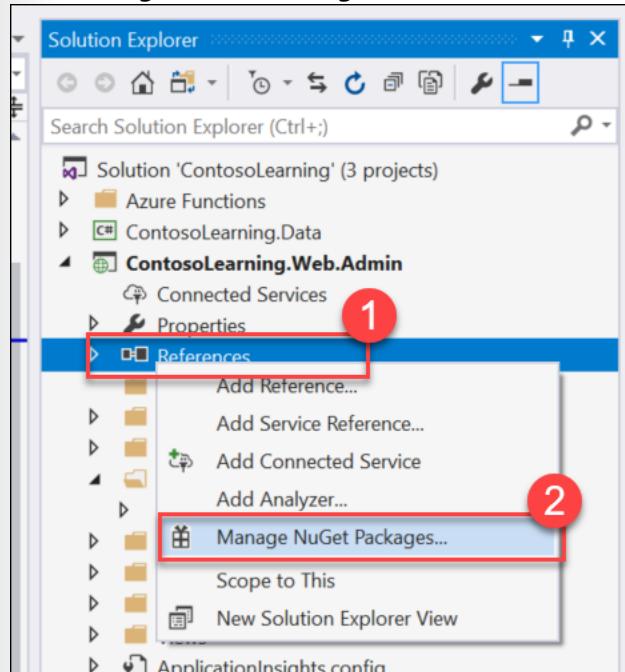
7. Paste in the following code where the comment states "Save Video info to Cosmos DB...". This code will use the VideoRepository to save a new Document to Cosmos DB for the newly uploaded video.

```
// Save new Document to Cosmos DB for this Video
var videoRepo = VideoRepositoryFactory.Create();
var video = new Video();
video.Id = videoId;
video.Title = model.Title;
video.Description = model.Description;
video.ProcessingState = "Processing";
video.ProcessingProgress = "0%";
await videoRepo.Insert(video);
```

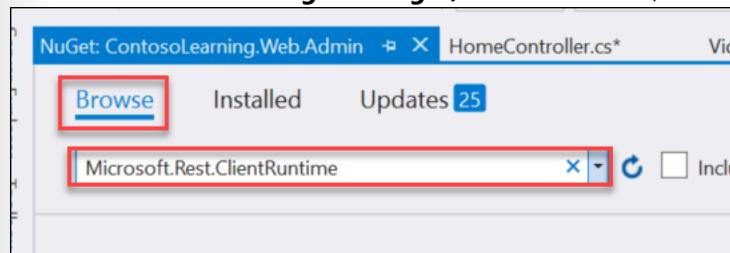
8. Save the file

Task 4: Add ability to delete video

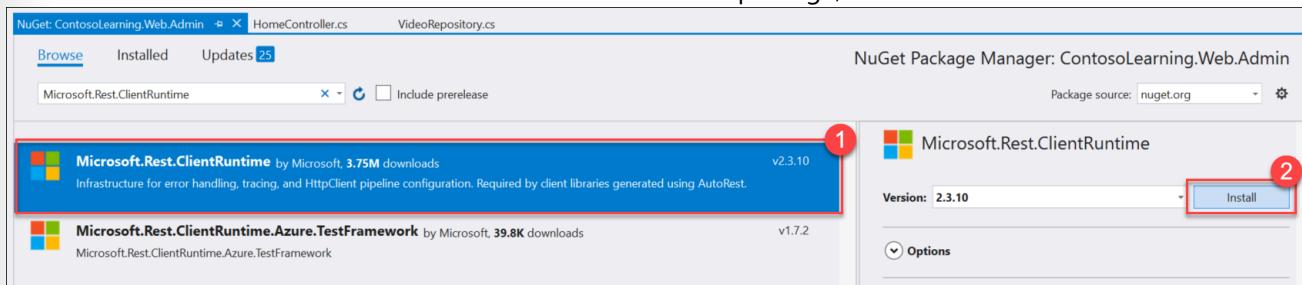
1. Within Solution Explorer, expand the **ContosoLearning.Web.Admin** project, then right-click on **References**, then click **Manage NuGet Packages...**



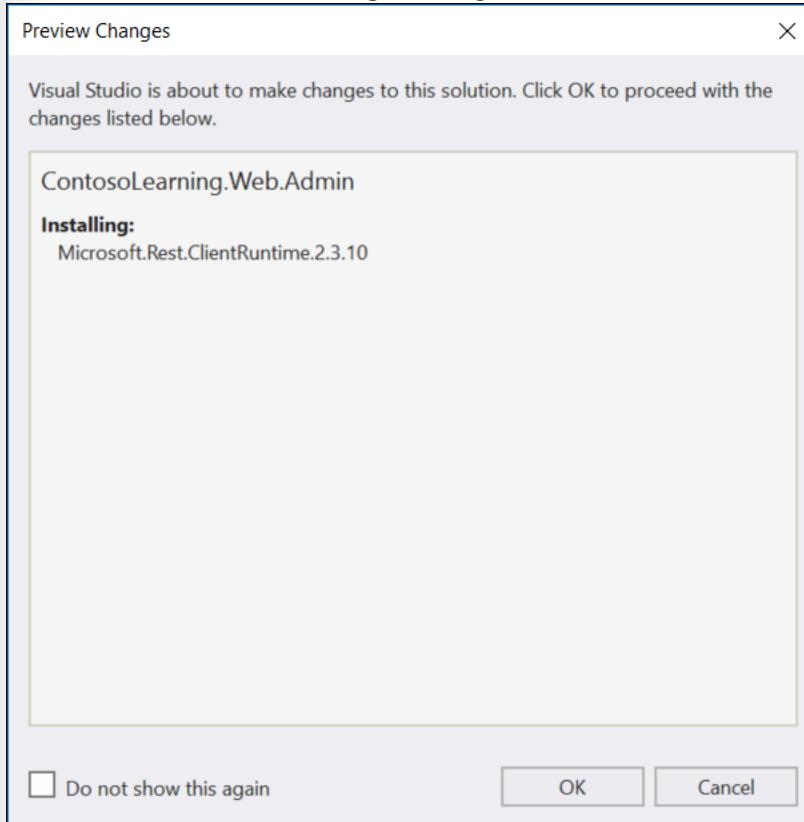
2. Within the **NuGet Package Manager**, click on **Browse**, then search for **Microsoft.Rest.ClientRuntime**



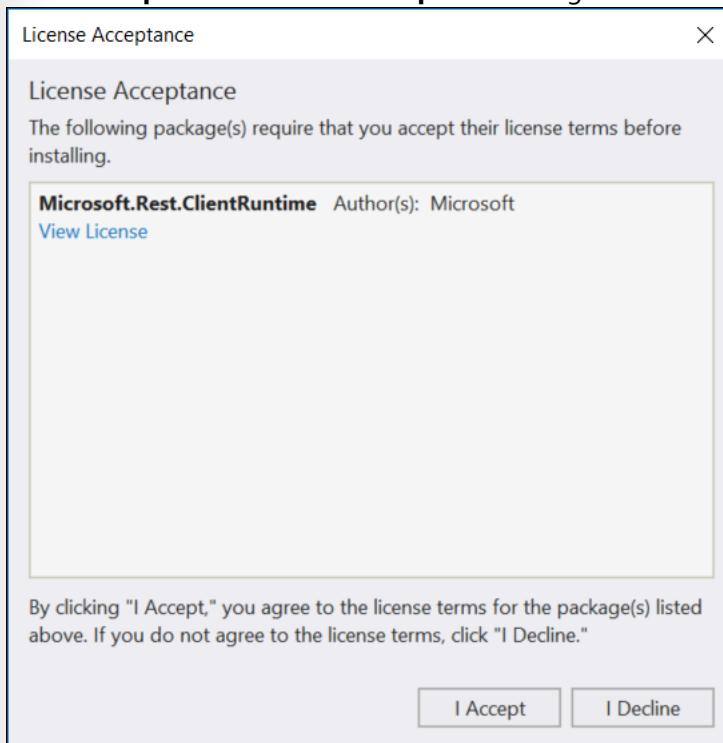
3. Click on and select the **Microsoft.Rest.ClientRuntime** NuGet package, then click **Install**



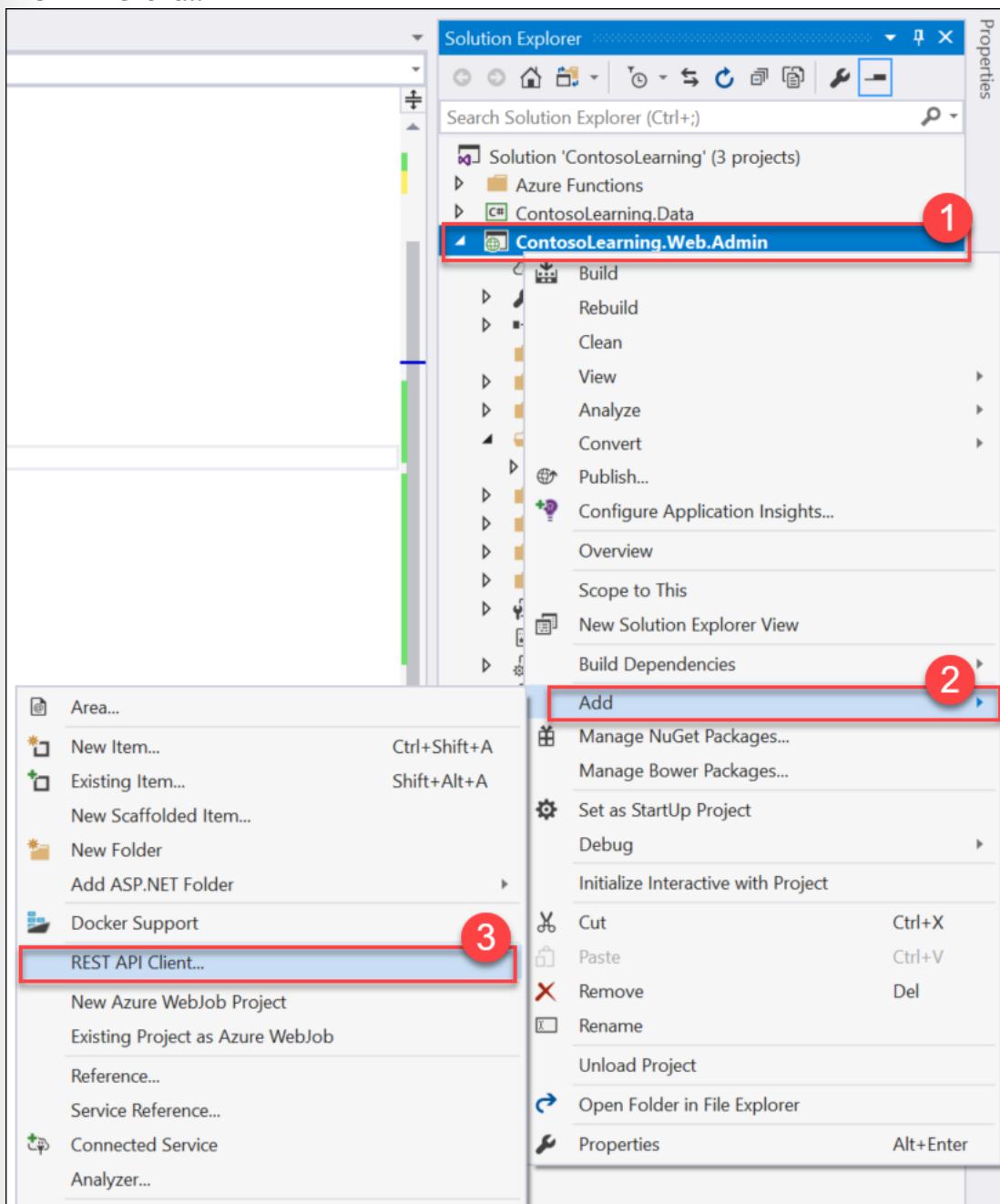
4. Click **OK** on the **Preview Changes** dialog



5. Click **I Accept** on the **License Acceptance** dialog

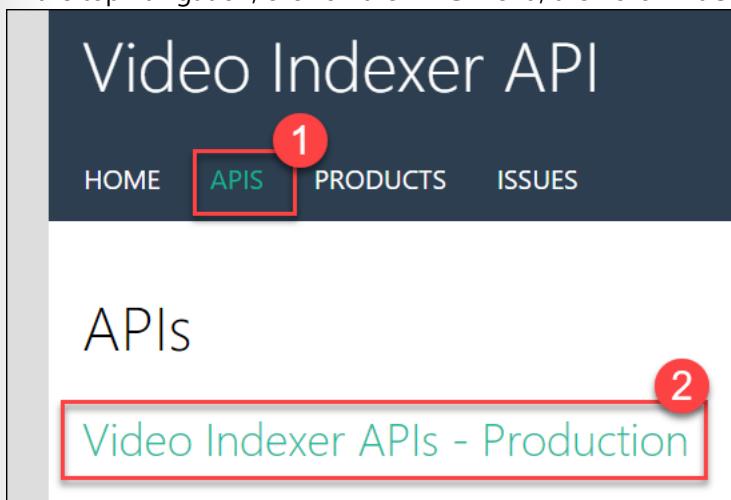


6. Within the Solution Explorer window, right-click the **ContosoLearning.Web.Admin** project, then click **Add**, then **REST API Client...**

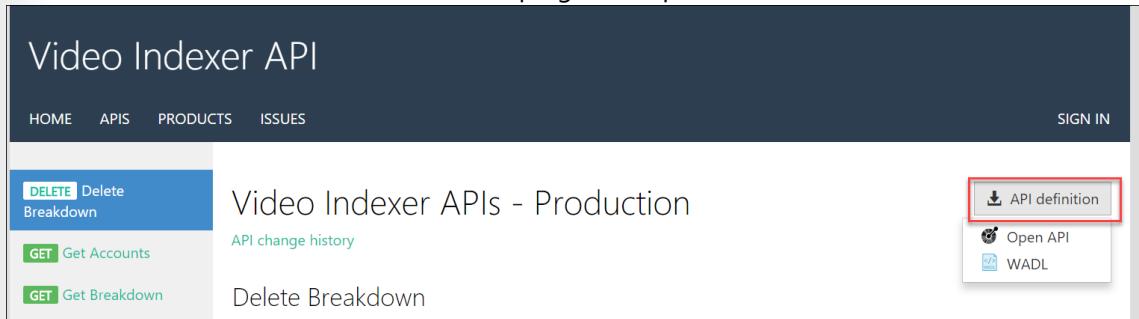


7. Open a new browser window and navigate to the **Video Indexer API** Developer Portal at:
<https://videobreakdown.portal.azure-api.net/>

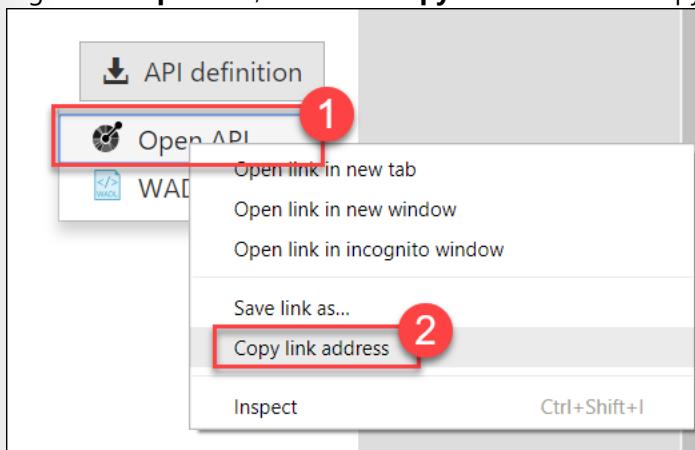
8. In the top navigation, click on the **APIS** menu, then click **Video Indexer APIs – Production**



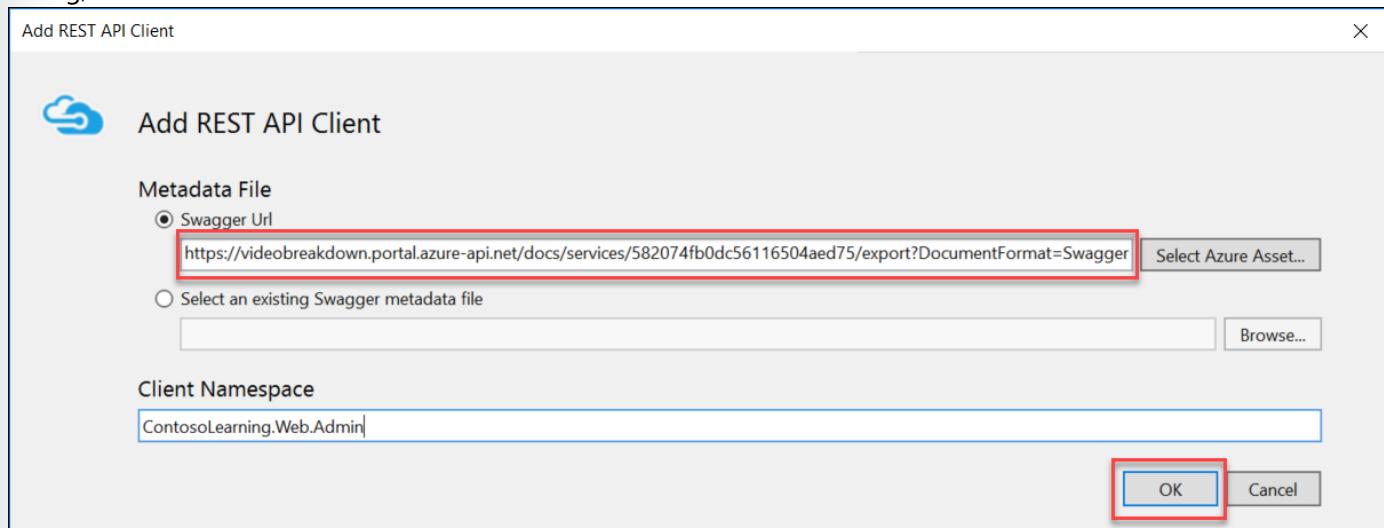
9. Click on the **API definition** button in the top right to expand its menu



10. Right-click **Open API**, then click **Copy link address** to copy the URL for the link to the clipboard



11. Go back to Visual Studio, and paste in the copied **URL** into the **Swagger Url** field of the **Add REST API Client** dialog, then click **OK**



12. Within the **HomeController.cs** file within the **ContosoLearning.Web.Admin** project, locate the **Delete(string id)** Action Method, and replace its contents with the following source code that will delete the Video from both Blob Storage and Cosmos DB

```
// =====
// Delete document from Cosmos DB
// =====

var videoRepo = VideoRepositoryFactory.Create();

var video = await videoRepo.Get(id);

await videoRepo.Delete(id);

// =====
// Delete files from Blob Storage
// =====

// Load Connection String to Azure Storage Account
var videoConnString = ConfigurationManager.ConnectionStrings["videostorage"].ConnectionString;
if (string.IsNullOrWhiteSpace(videoConnString))
{
    throw new Exception("The 'videostorage' Connection String is NOT set");
}

// Get reference to the Blob Container to upload to
var storageAccount = CloudStorageAccount.Parse(videoConnString);
var blobClient = storageAccount.CreateCloudBlobClient();
```

```
// Get reference to 'video' container
var videoContainer = blobClient.GetContainerReference("video");
await videoContainer.CreateIfNotExistsAsync();

// Delete Video file from Blob Storage
var videoBlob = videoContainer.GetBlockBlobReference(id);
await videoBlob.DeleteAsync();
```

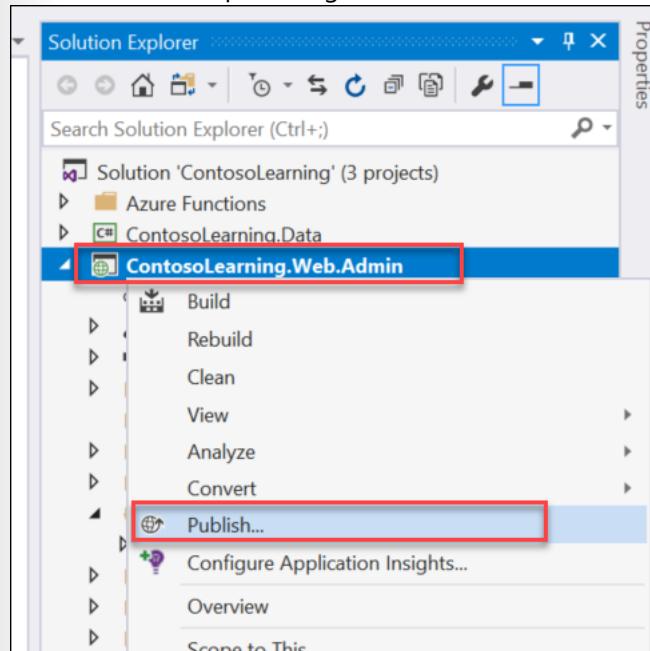
13. After the above code, within the **Delete** method, paste in the following code that will also delete the video from **Video Indexer** when it's deleted from the application.

```
var videoIndexerClient = new AdminClient(
    new Microsoft.Rest.TokenCredentials(
        ConfigurationManager.AppSettings["VideoIndexerAPI_Key"]
    )
);
var response = await videoIndexerClient.DeleteBreakdownWithHttpMessagesAsync(video.VideoId, true);
```

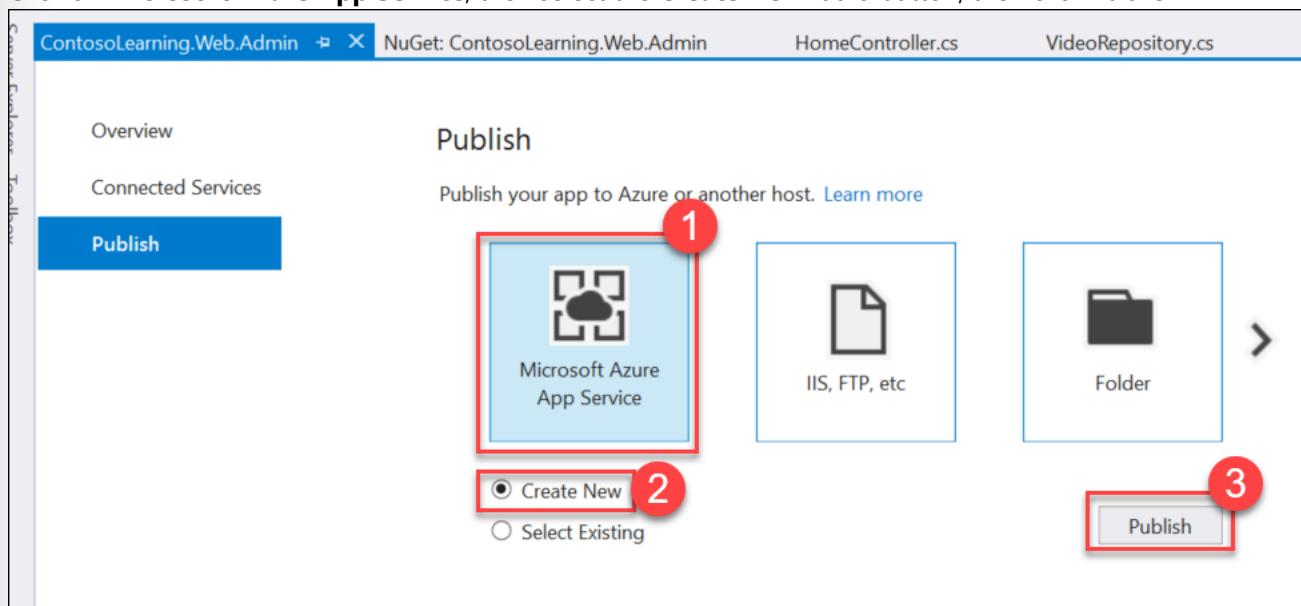
14. Save the file

Task 5: Deploy Admin website to an Azure Web App

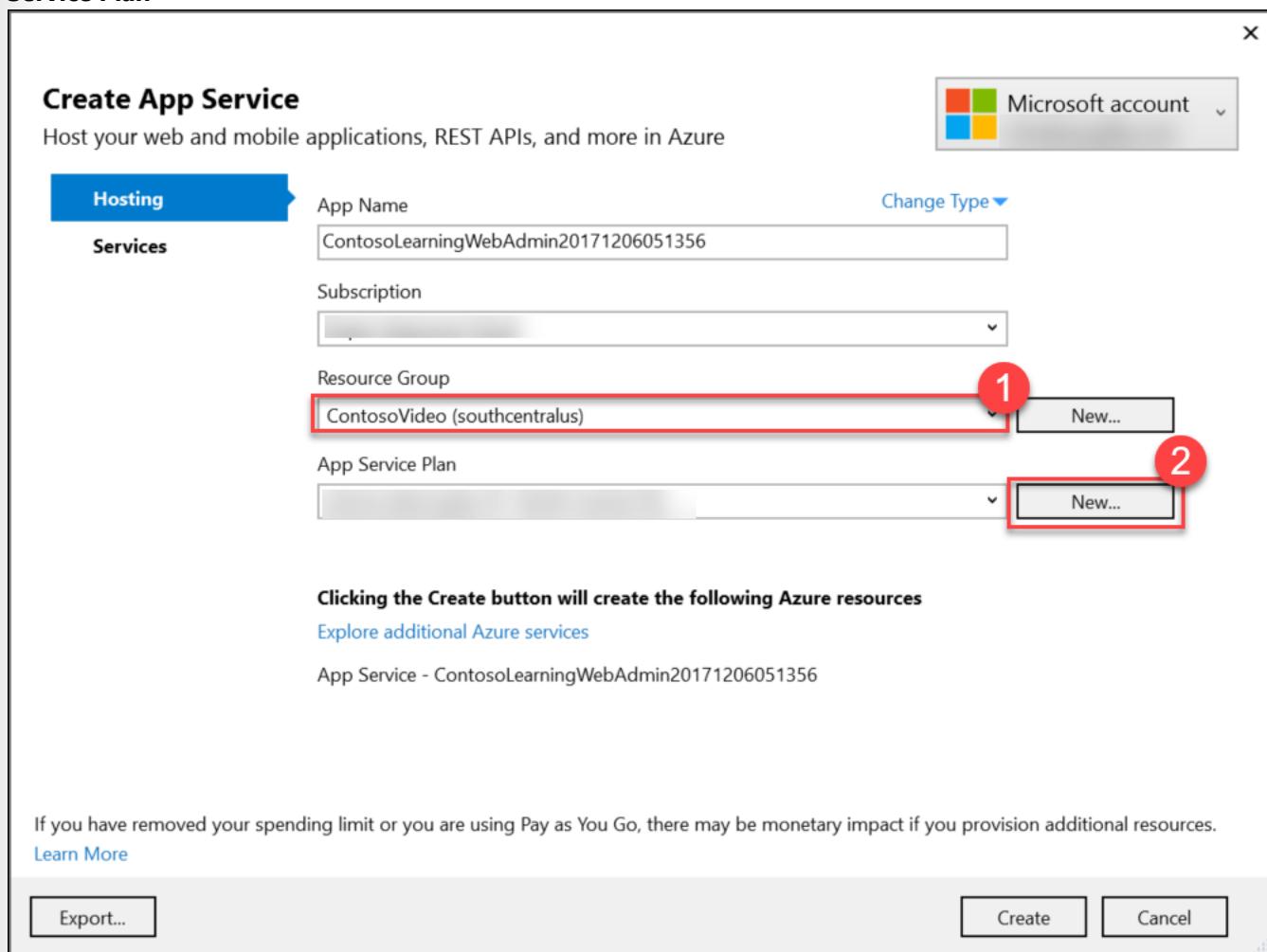
1. Within Solution Explorer, right-click the **ContosoLearning.Web.Admin** project, then click **Publish...**



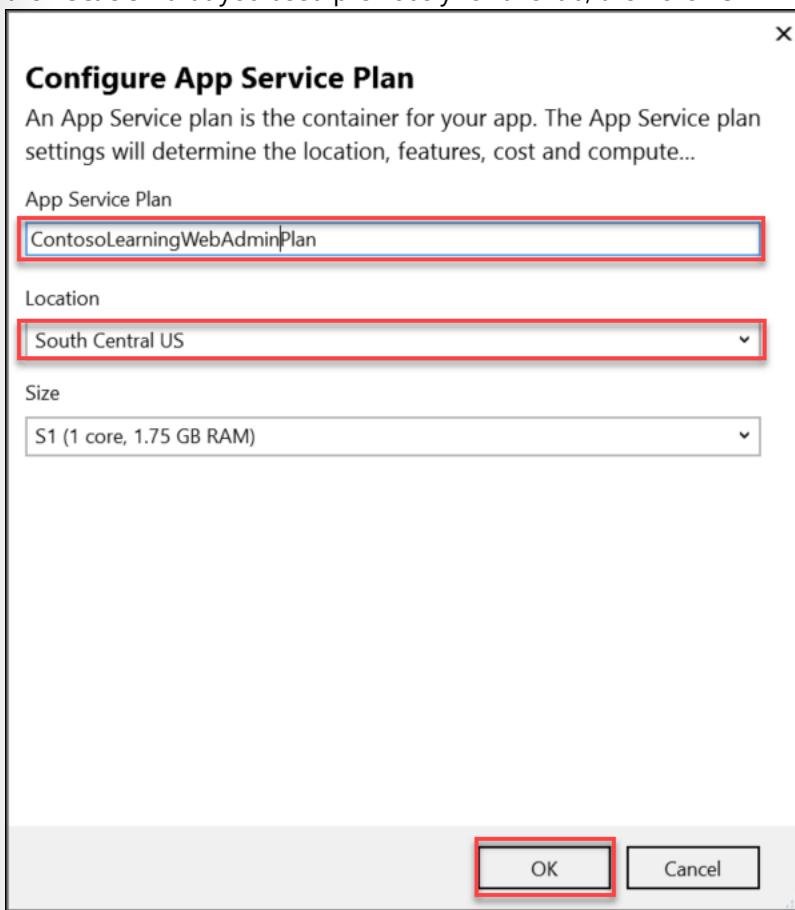
2. Click on **Microsoft Azure App Service**, then select the **Create New** radio button, then click **Publish**



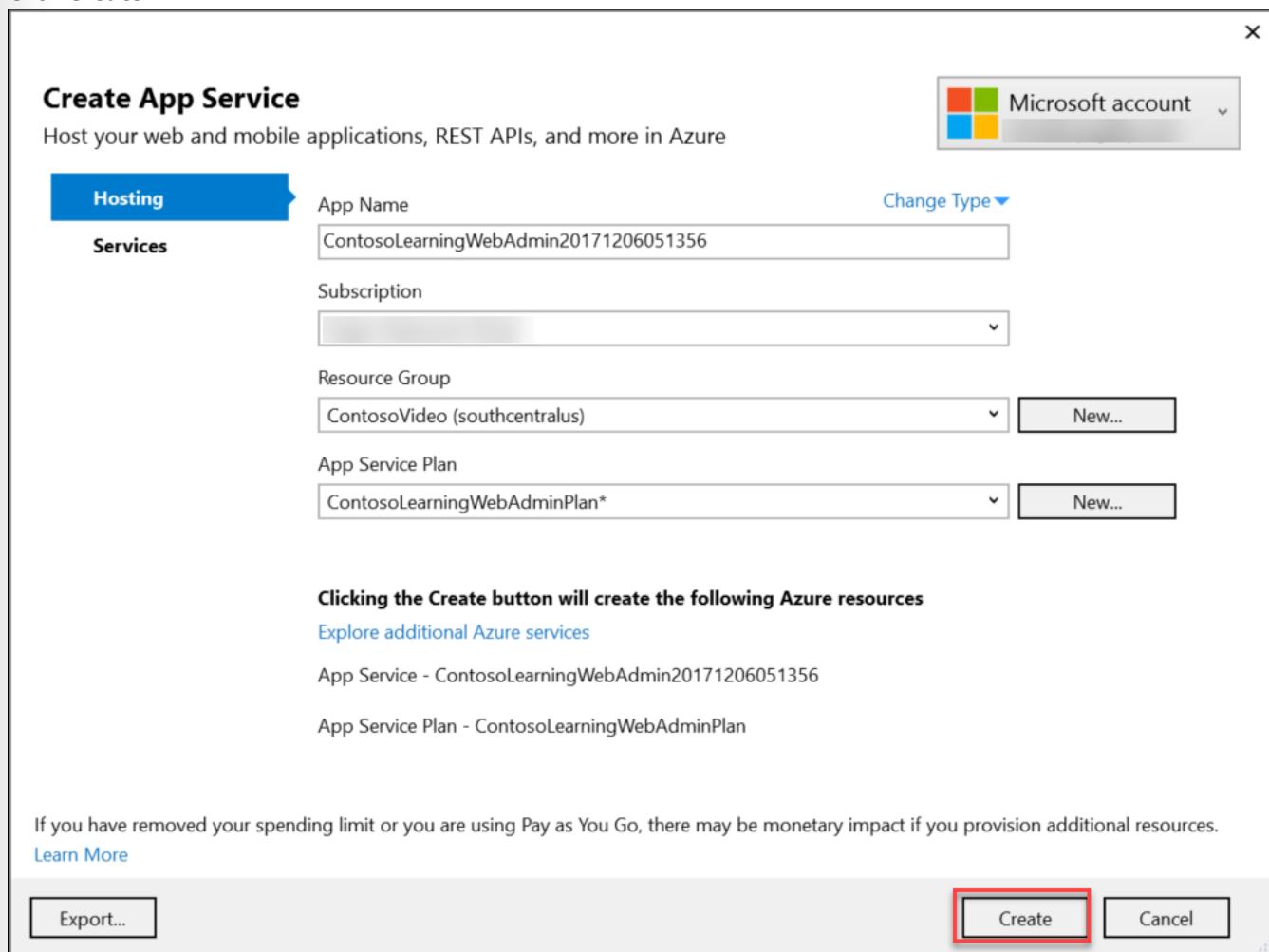
3. On the **Create App Service** dialog, select the **ContosoVideo** Resource Group, then click **New** next to **App Service Plan**



- On the **Configure App Service Plan** dialog, enter a valid name into the **App Service Plan** name field, then select the **Location** that you used previously for this lab, then click **OK**



5. Click **Create**



6. Once the application has finished deploying to the Azure Web App, a new browser window will be opened and navigated to the web app running in Azure

7. You should now see an Error screen for the app. This is normal since the AppSettings haven't been configured yet.

Server Error in '/' Application.

Invalid URI: The URI is empty.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information originated in the code.

Exception Details: System.UriFormatException: Invalid URI: The URI is empty.

Source Error:

An unhandled exception was generated during the execution of the current web request. Information regarding the exception can be identified using the exception stack trace below.

Stack Trace:

```
[UriFormatException: Invalid URI: The URI is empty.]  
System.Uri.CreateThis(String uri, Boolean dontEscape, UriKind uriKind) +6643964  
System.Uri.Create(String urlString) +21
```

Task 6: Configure Application Settings

1. Open the Azure Portal, click on **Resource groups** in the left-side menu, then click the **ContosoVideo** Resource Group, then click on the **Azure Cosmos DB Account** that was created previously

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is visible with items like 'Dashboard', 'Resource groups' (marked with a red circle and number 1), 'All resources', 'App Services', 'Virtual machines', 'Virtual machine scale sets', 'SQL databases', and 'Security Center'. The main area shows the 'Resource groups' blade for 'Super Awesome Cloud'. It lists one resource group named 'ContosoVideo' (marked with a red circle and number 2). On the right, the 'ContosoVideo' Resource Group blade is open, showing details like 'Subscription (change) Super Awesome Cloud', 'Subscription ID 41850220-19f0-49c6-87ea-766c9b0da1f', and a list of resources including 'azureblob', 'contosovideo' (marked with a red circle and number 3), 'contosovideodb1', and 'videoindexer'. The 'Keys' tab is selected in the bottom navigation bar of the Resource Group blade.

2. On the **Azure Cosmos DB Account** blade, click on **Keys**

The screenshot shows the 'Keys' tab selected in the bottom navigation bar of the 'ContosoVideo' Resource Group blade. Other tabs include 'Firewall', 'Keys' (selected), and 'Add Azure Search'.

3. Copy the **URI** and **Primary Key** for the Cosmos DB Account for use later

The screenshot shows the 'Keys' blade for the 'contosovideodb' Cosmos DB account. On the left is a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Quick start, and Data Explorer. The main area has tabs for 'Read-write Keys' and 'Read-only Keys'. Under 'Read-write Keys', the 'URI' is listed as 'https://contosovideodb.documents.azure.com:443/' and the 'PRIMARY KEY' is listed as 'ZkyTwtfip2WeziHq9UUJxn9t0bAyrvV26mXEzelZoEcnRFbvo54vTNVZwe7qELkvKstmscuQVcJzcD4C1mg=='. Both fields are highlighted with red boxes.

4. Click on **Resource groups** in the left-side menu, then click the **ContosoVideo** Resource Group, then click on the **Storage Account** created in Exercise 2

The screenshot shows the 'Resource groups' blade in the Azure portal. The left sidebar has a 'Resource groups' item with a red circle containing the number 1. The main area lists four resource groups: 'ContosoVideo' (highlighted with a red box and red circle 2), 'key1', 'key2', and 'key3'. To the right, the 'ContosoVideo' resource group blade is open, showing its overview and settings. In the bottom right corner of this blade, there is a red circle containing the number 3, pointing to the 'Storage accounts' section where the 'contosovideo' storage account is listed.

5. On the **Storage account** blade, click on **Access keys**

The screenshot shows the 'Access keys' blade for a storage account. The left sidebar has a 'SETTINGS' section with an 'Access keys' item highlighted with a red box and red circle 1. The main area shows two access keys: 'key1' and 'key2'. 'key1' is highlighted with a red box and red circle 2. The 'Key' field contains the value 'rqAVBL+7O4JYPnRS8ojt8Utqu5pu50YuYC...'. Below it, the 'Connection string' field contains the value 'DefaultEndpointsProtocol=https;AccountName=contosovideo1;AccountKey=rqAVBL+7O4JYPnRS8ojt8Utqu...'.

6. Copy the **Connection String** for Key 1 for use later

The screenshot shows the 'Access keys' blade for a storage account. The left sidebar has a 'SETTINGS' section with an 'Access keys' item highlighted with a red box and red circle 1. The main area shows two access keys: 'key1' and 'key2'. 'key1' is highlighted with a red box and red circle 2. The 'Key' field contains the value 'rqAVBL+7O4JYPnRS8ojt8Utqu5pu50YuYC...'. Below it, the 'Connection string' field contains the value 'DefaultEndpointsProtocol=https;AccountName=contosovideo1;AccountKey=rqAVBL+7O4JYPnRS8ojt8Utqu...'. A red box highlights the 'Connection string' field.

7. Click on **Resource groups** in the left-side menu, then click the **ContosoVideo** Resource Group, then click on the **Azure Web App**

The screenshot shows the Microsoft Azure portal interface. On the left, the sidebar includes links for 'Create a resource', 'All services', 'FAVORITES' (Dashboard, All resources, App Services, Virtual machines, Virtual machine scale sets, SQL databases, Security Center, Subscriptions, Monitor), and other options like 'Continuous Delivery (Preview)', 'Application settings', and 'Authentication / Authorization'. The main area shows 'Resource groups' under 'Super Awesome Cloud' with one item: 'ContosoVideo'. A red circle with the number '1' is over the 'Resource groups' link in the sidebar. A red circle with the number '2' is over the 'ContosoVideo' resource group in the list. The 'ContosoVideo' blade on the right shows 'Overview', 'Activity log', 'Access control (IAM)', and 'Tags' sections, along with 'SETTINGS' like 'Quickstart', 'Resource costs', 'Deployments', 'Policies', 'Properties', 'Locks', and 'Automation script'. A red circle with the number '3' is over the 'ContosoLearningWebAdmin' Azure Web App in the list of resources.

8. On the **App Service** blade, click on **Application settings**

The screenshot shows the 'App Service' blade for 'ContosoLearningWebAdmin'. It has sections for 'Continuous Delivery (Preview)', 'SETTINGS', and 'Authentication / Authorization'. A red box highlights the 'Application settings' link under 'SETTINGS'.

9. On the **Application settings** pane, scroll down to the **App settings** section

The screenshot shows the 'Application settings' pane. It displays environment variables: 'WEBSITE_NODE_DE...' with value '6.9.1' and a 'Slot Setting' checkbox. Below is a button '+ Add new setting'. The 'App settings' section is visible at the bottom of the pane.

10. Add a new App Setting with the Key of **CosmosDB_Endpoint** with the Value set to the **Cosmos DB Account URI** that was copied

Application settings

| |
|---|
| WEBSITE_NODE_DE... 6.9.1 |
| CosmosDB_Endpoint /contosovideodb1.documents.azure.com:443/ |
| + Add new setting |

11. Add a new App Setting with the Key of **CosmosDB_AuthKey** with the Value set to the **Cosmos DB Account Primary Key** that was copied

Application settings

| |
|--|
| WEBSITE_NODE_DE... 6.9.1 |
| CosmosDB_Endpoint https://contosovideodb1.documents.azure.c... |
| CosmosDB_AuthKey r3gpsYpGUJrrwz2KrlxFaAqWiMOAIYTxrJ0g= |
| + Add new setting |

12. Add a new App Setting with the Key of **CosmosDB_Database** with the Value set to **learning**

Application settings

| |
|--|
| WEBSITE_NODE_DE... 6.9.1 |
| CosmosDB_Endpoint https://contosovideodb1.documents.azure.c... |
| CosmosDB_AuthKey qHDBJfE4VdzdQP6MqPV7RHfxclnoec4YjB7u... |
| CosmosDB_Database learning |
| + Add new setting |

13. Add a new App Setting with the Key of **CosmosDB_Collection** with the Value set to **videos**

Application settings

| |
|--|
| WEBSITE_NODE_DE... 6.9.1 |
| CosmosDB_Endpoint https://contosovideodb1.documents.azure.c... |
| CosmosDB_AuthKey qHDBJfE4VdzdQP6MqPV7RHfxclnoec4YjB7u... |
| CosmosDB_Database learning |
| CosmosDB_Collection videos |
| + Add new setting |

14. Add a new App Setting with the Key of **VideoIndexerAPI_Key** with the Value set to the **Video Indexer API Key** that was copied previously

The screenshot shows a table with two columns: 'Key' and 'Value'. A red box highlights the 'Key' column for 'VideoIndexerAPI_Key' and the 'Value' column for 'bafa7860d1b44a239ee2baad6f0f0bed'. Below the table is a button labeled '+ Add new setting'.

15. Locate the **Connection strings** section, and add a new Connection String with the following values:

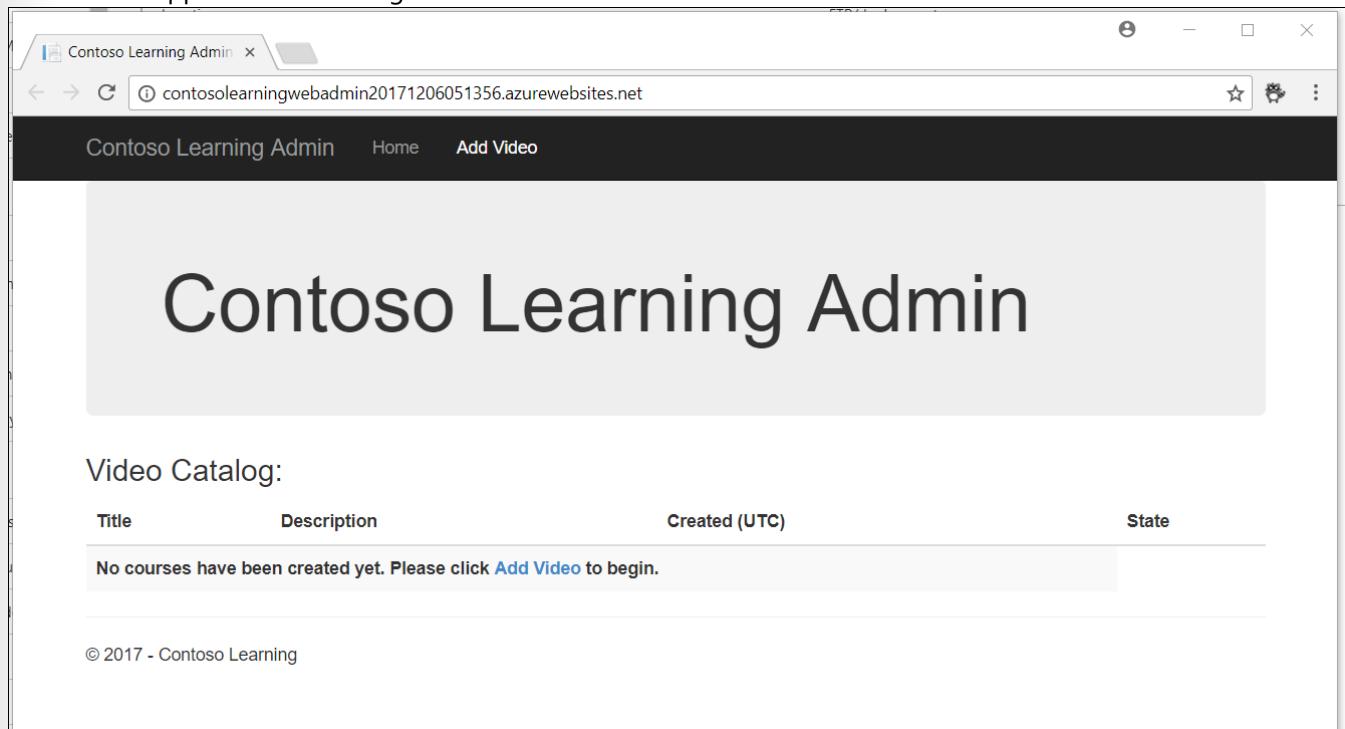
- Name: **videostorage**
- Value: **paste in the Storage Account connection string copied previously**
- Type: **Custom**

The screenshot shows a table with three columns: 'Name', 'Value', and 'Type'. A red box highlights the 'Value' column for 'videostc' which contains 'x=core.windows.net'. Below the table is a button labeled '+ Add new connection string'.

16. Click **Save**

The screenshot shows a dashboard for 'ContosoLearningWebAdmin20171206051356 - Application settings'. A red box highlights the 'Save' button. Other visible elements include a search bar, a 'Discard' button, an 'ARR Affinity' switch set to 'On', and a 'Overview' link.

17. Refresh the browser with the Admin Web App running in it. If you closed it, then open it up again. You will now see that the application is loading without error.



18. **Don't upload a video through the Admin app yet.** We still need to finish setting up the back-end of the application.

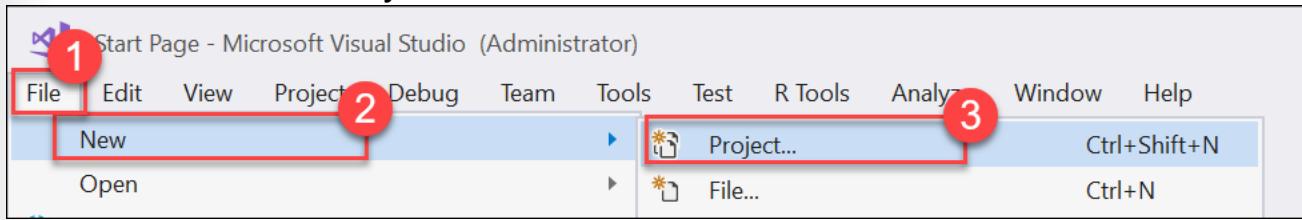
Exercise 4: Update video status when processing is complete

Duration: 20 minutes

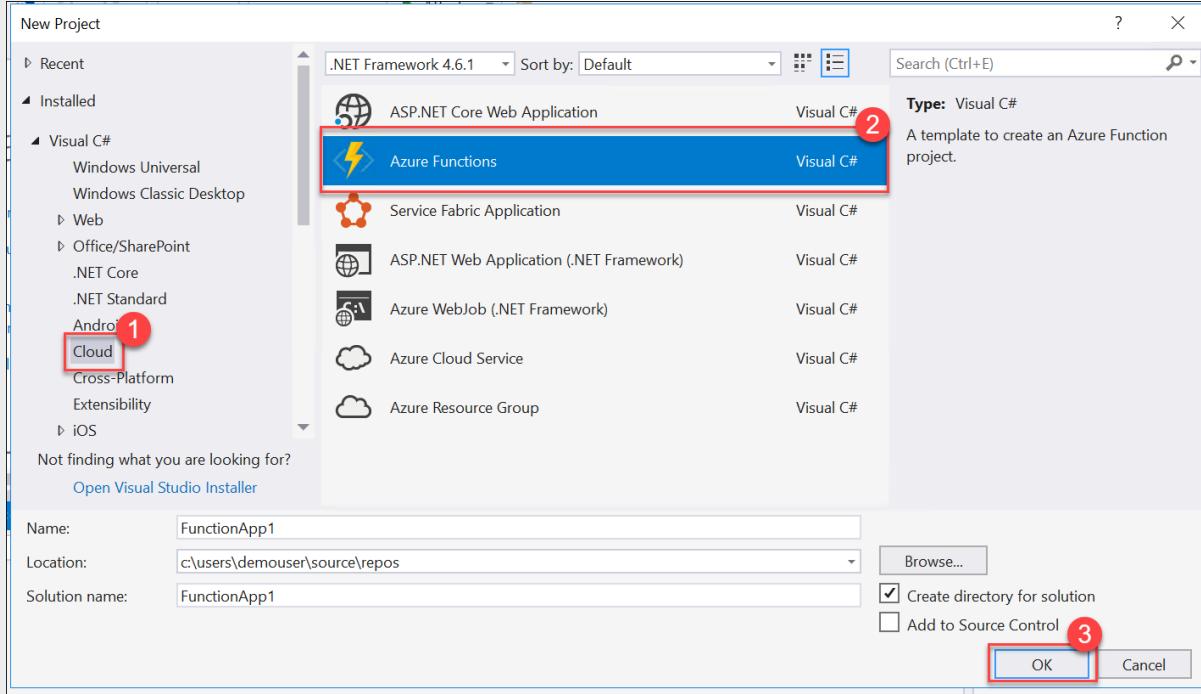
In this exercise, you will integrate an Azure Function with the Logic App Workflow so that the Azure Cosmos DB database is updated when a video is finished being processed within Video Indexer.

Step 1: Create Azure Function

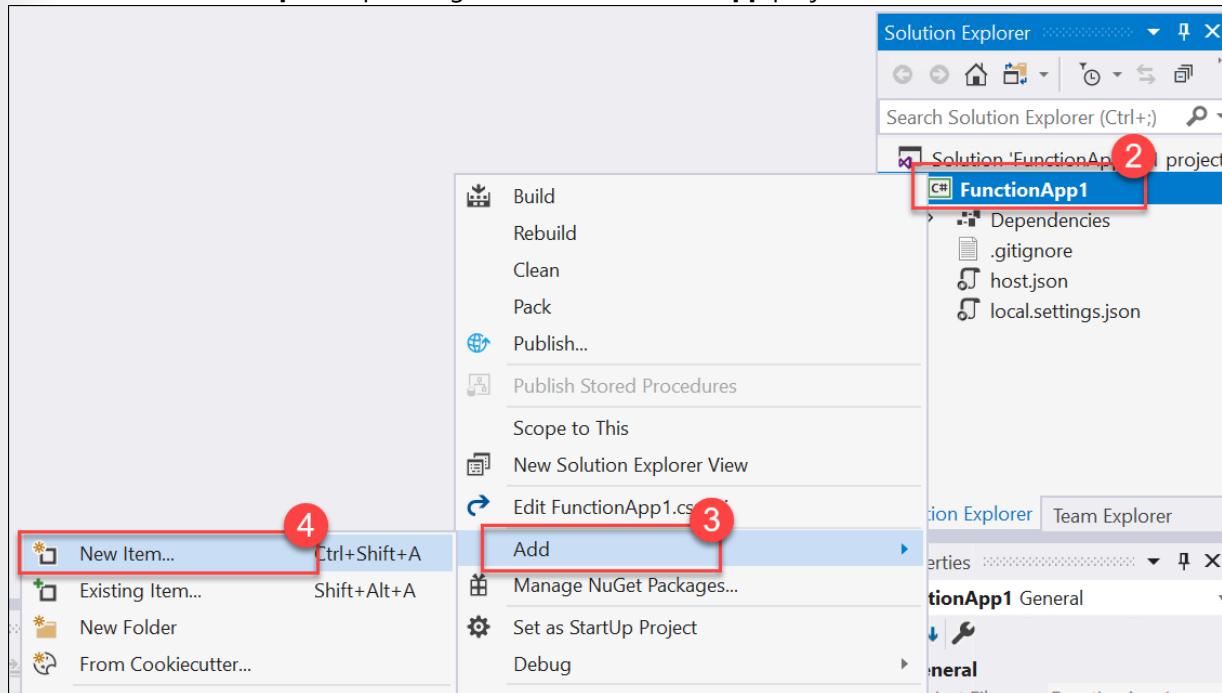
1. Open **Visual Studio 2017**
2. Click on the **File** menu, then **Project...**



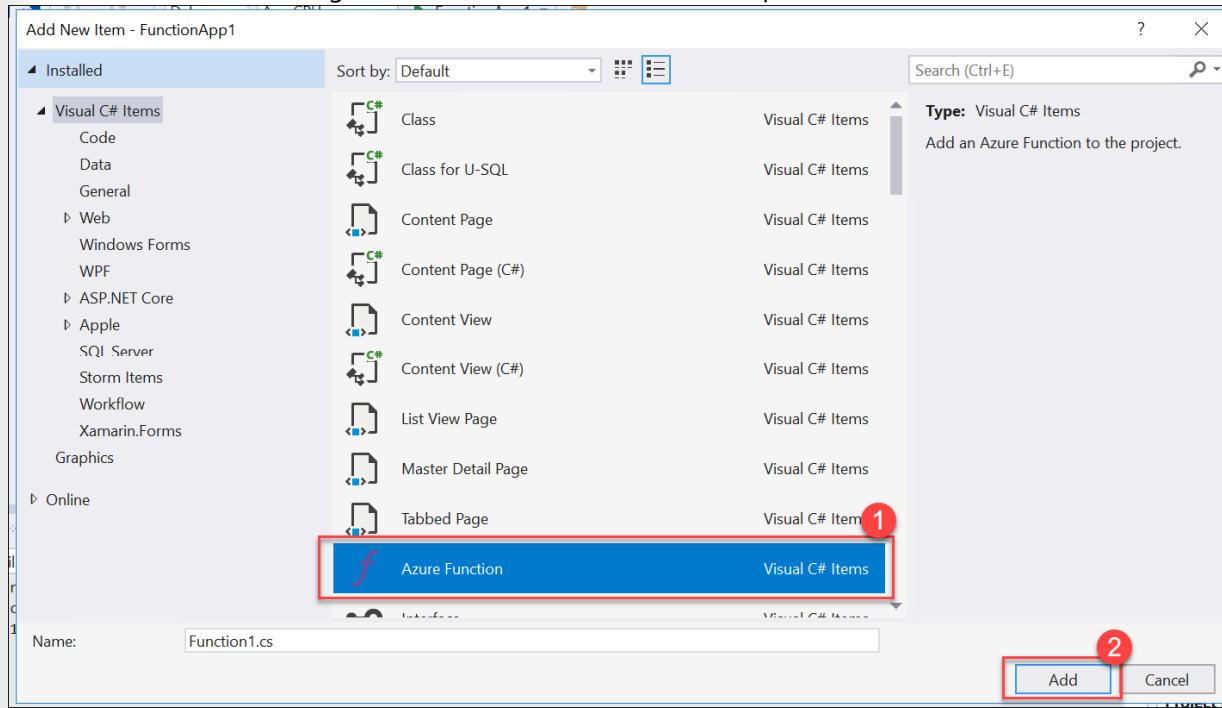
3. On the **New Project** dialog, select the **Cloud** category underneath **Visual C#** on the left, then select the **Azure Function** project template, then click **OK**



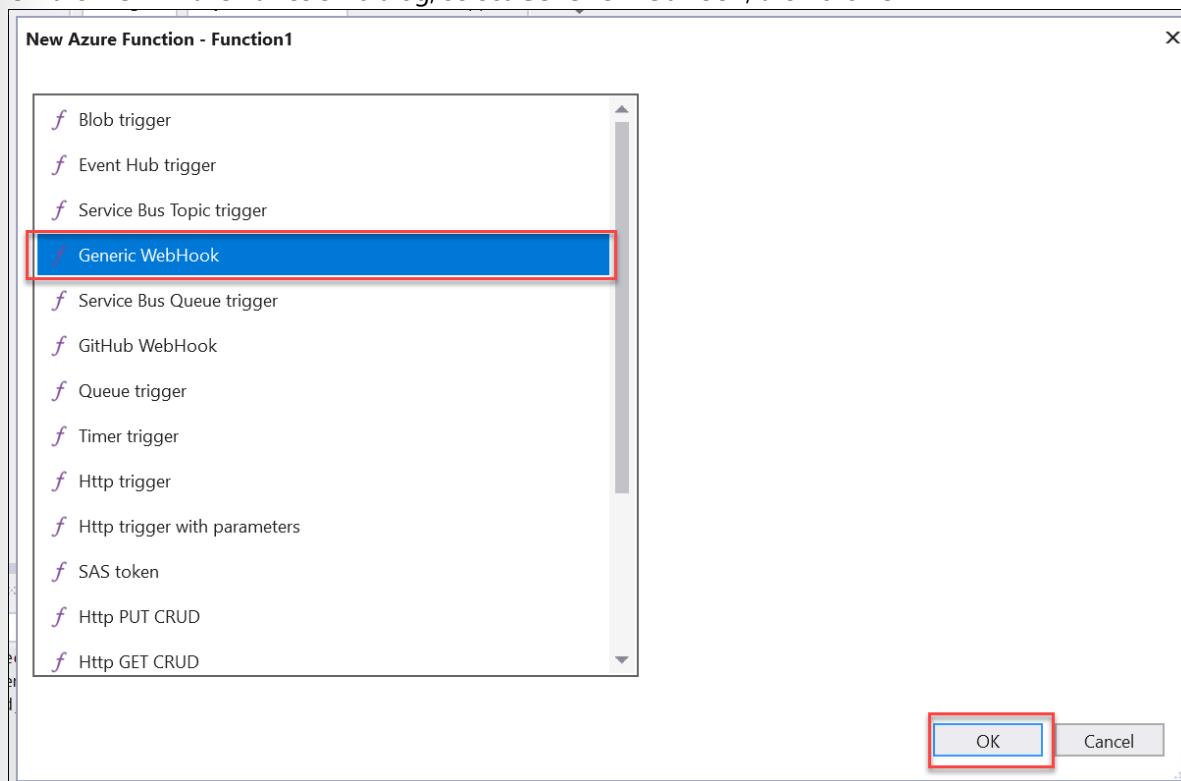
4. Within the **Solution Explorer** pane, right-click the **FunctionApp** project, then select **Add**, then **New Item...**



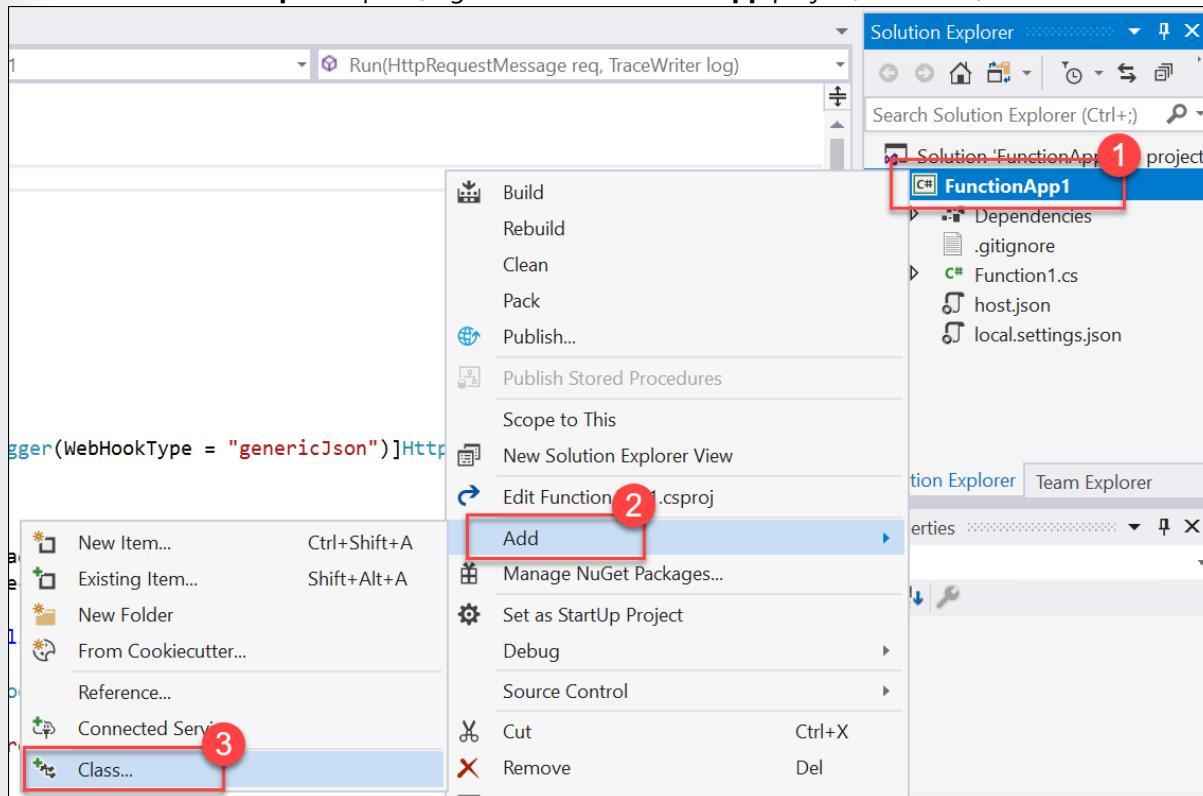
5. In the **Add New Item** dialog, select the **Azure Function** file template, then click **Add**



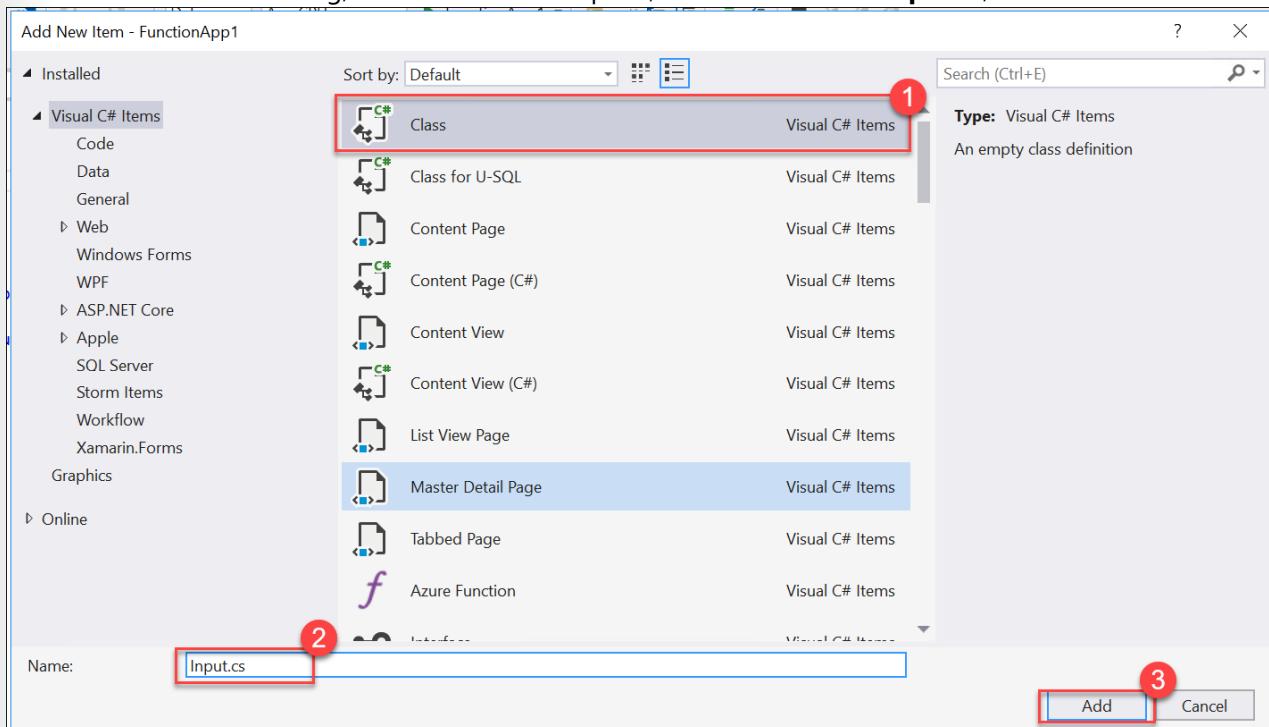
6. On the **New Azure Function** dialog, select **Generic WebHook**, then click **OK**



7. Within the **Solution Explorer** pane, right-click the **FunctionApp** project, click **Add**, then **Class...**



8. In the **Add New Item** dialog, select the **Class** template, then name the file **Input.cs**, then click **Add**



9. In the code for the **Input.cs** class, update it to match the below **Input** class with the following two properties

```
public class Input
{
    public string documentId { get; set; }
    public string videoId { get; set; }
}
```

10. Save the file

11. Open the **Function1.cs** file for the new Function that was previously created

12. Update signature of the **Run** method to contain an **input** parameter of type **Input**. This will allow the Azure Function to automatically pull in the parameter values (documentId, videoid) that are passed in through the Request Body and parse them into this Input parameter value.

```
public static async Task<object> Run(
    [HttpTrigger]Input input,
    HttpRequestMessage req,
    TraceWriter log)
```

13. Replace the body of the **Run** method with the following code that checks the necessary parameters are passed in to the method

```
log.Info("Function triggered...");

// Log the parameters passed in through the Request Body
log.Info($"DocumentId: {input.documentId}");
log.Info($"VideoId: {input.videoId}");

if (string.IsNullOrEmpty(input.videoId) || string.IsNullOrEmpty(input.documentId))
{
    log.Error("DocumentId and/or VideoId parameter missing!");
    return req.CreateResponse(HttpStatusCode.BadRequest, $"Please pass a 'videoId' and 'documentId' in the Http request body");
}

return req.CreateResponse(HttpStatusCode.OK, "Success");
```

14. The resulting **Run** method should look as follows:

```
[FunctionName("Function1")]
public static async Task<object> Run(
    [HttpTrigger]Input input,
    HttpRequestMessage req,
    TraceWriter log)
{
    log.Info("Function triggered...");

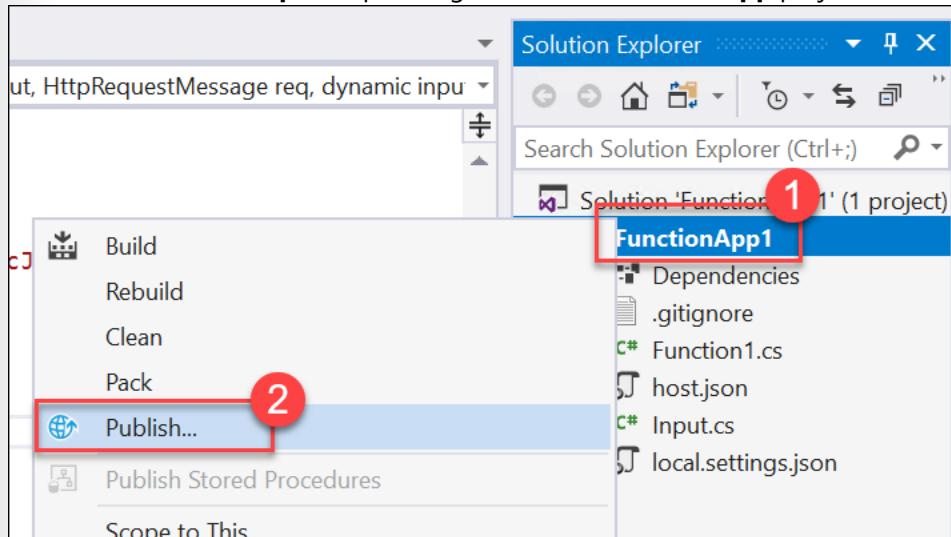
    // Log the parameters passed in through the Request Body
    log.Info($"DocumentId: {input.documentId}");
    log.Info($"VideoId: {input.videoId}");

    if (string.IsNullOrEmpty(input.videoId) || string.IsNullOrEmpty(input.documentId))
    {
        log.Error("DocumentId and/or VideoId parameter missing!");
        return req.CreateResponse(HttpStatusCode.BadRequest, $"Please pass a 'videoId' and 'documentId' in the Http request body");
    }

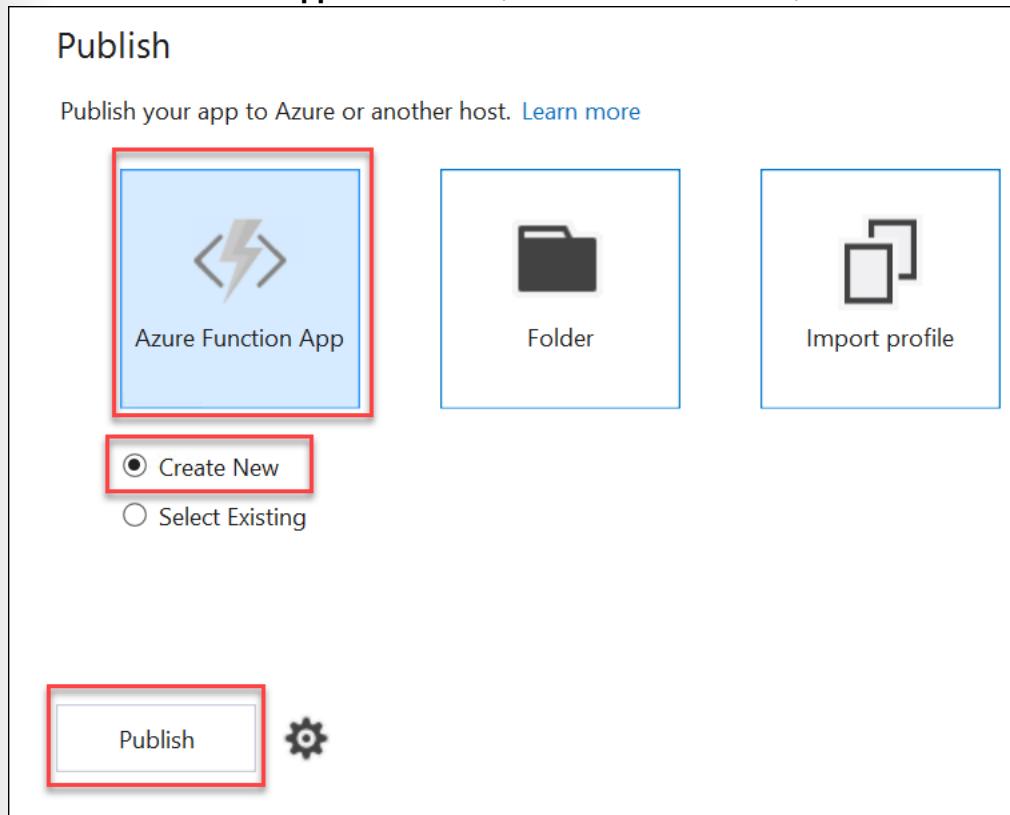
    return req.CreateResponse(HttpStatusCode.OK, "Success");
}
```

15. Save the source code files

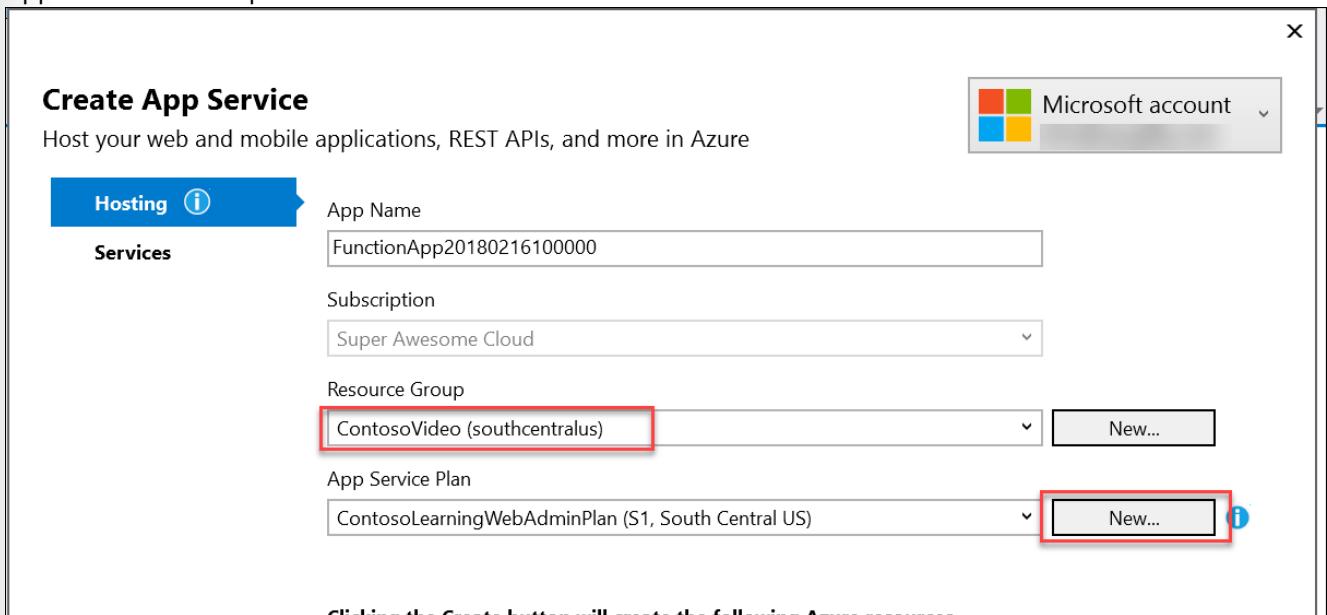
16. Within the **Solution Explorer** pane, right-click the **Function App** project, then click **Publish**



17. Select **Azure Function App** under Publish, then select **Create New**, then click **Publish**

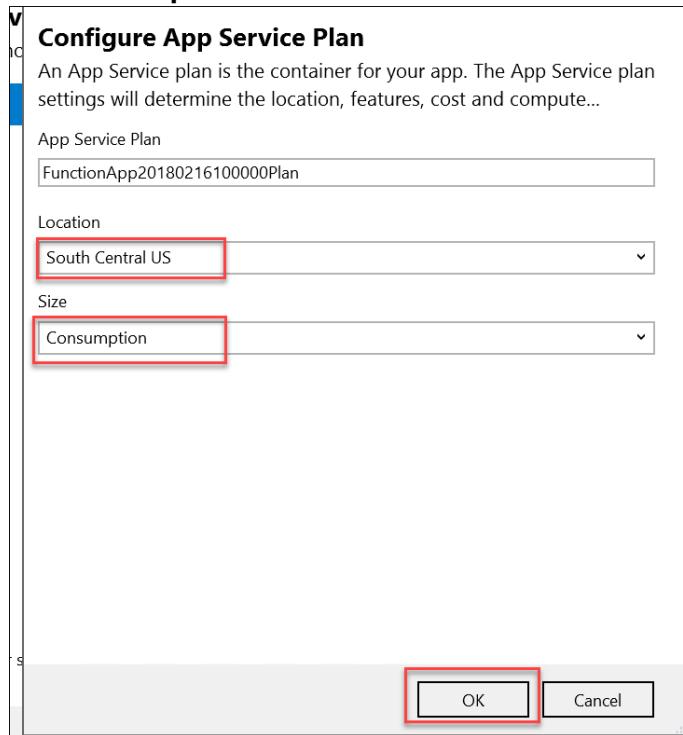


18. On the **Create App Service** dialog, select the **ContosoVideo** Resource Group, then click on **New...** next to the App Service Plan dropdown

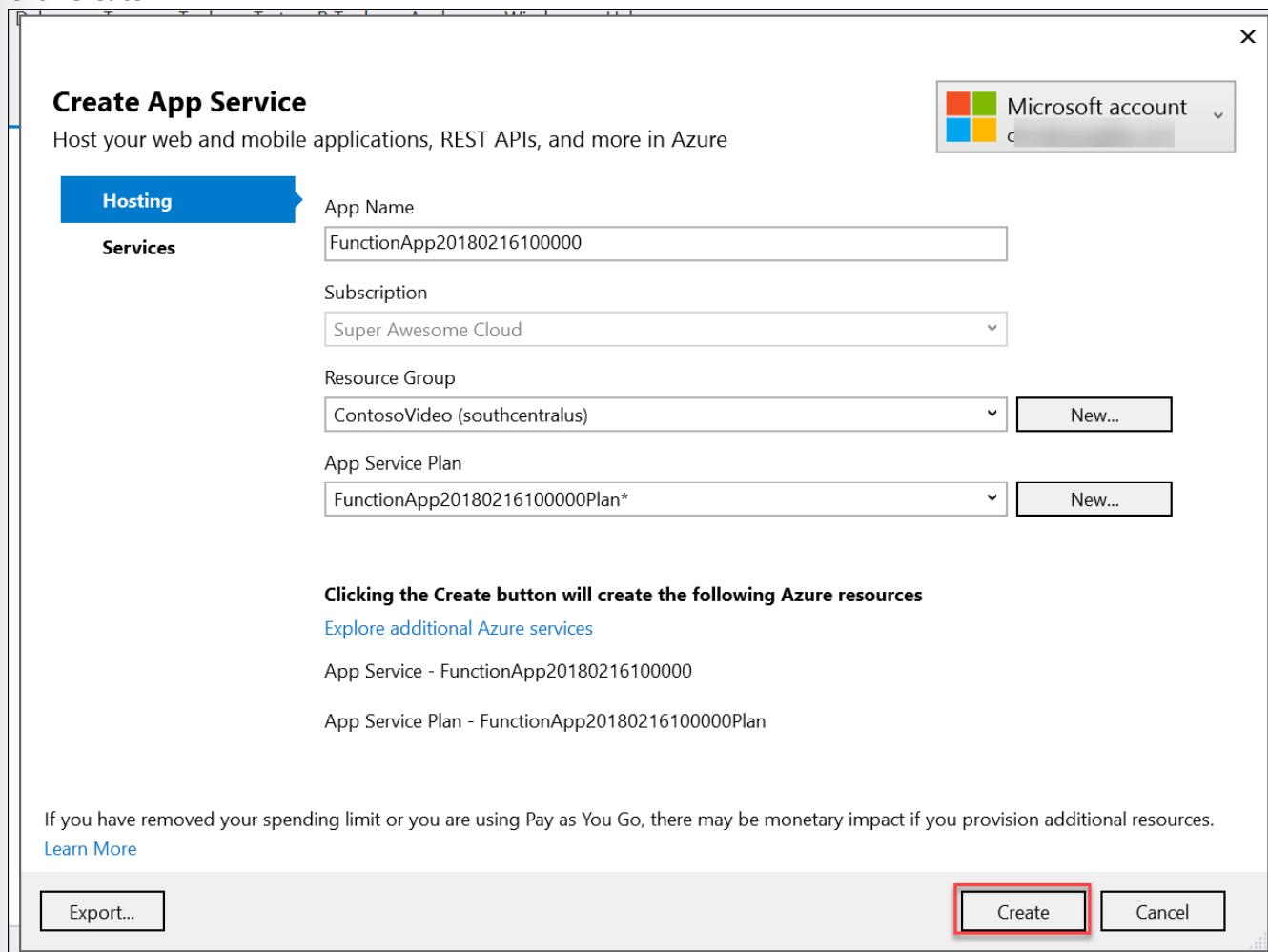


19. On the Configure App Service Plan dialog, enter the following values, then click **OK**:

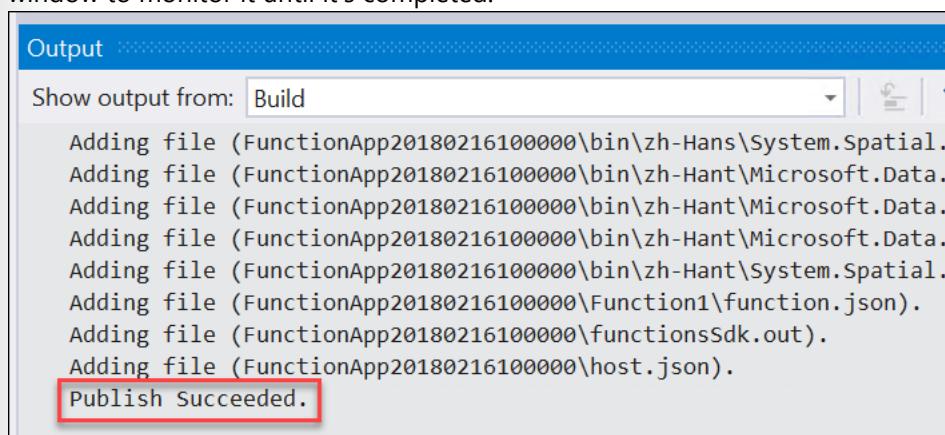
- Location: choose the same location you used for the rest of the lab
- Size: Consumption



20. Click **Create**

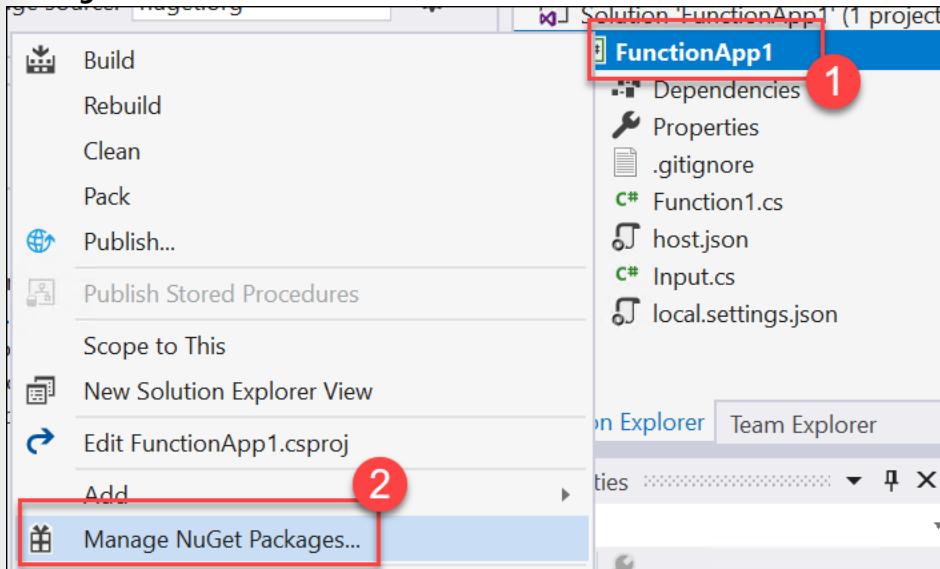


21. Wait for the publish to complete. This should take about 1 minute to complete. You can watch the **Output** window to monitor it until it's completed.

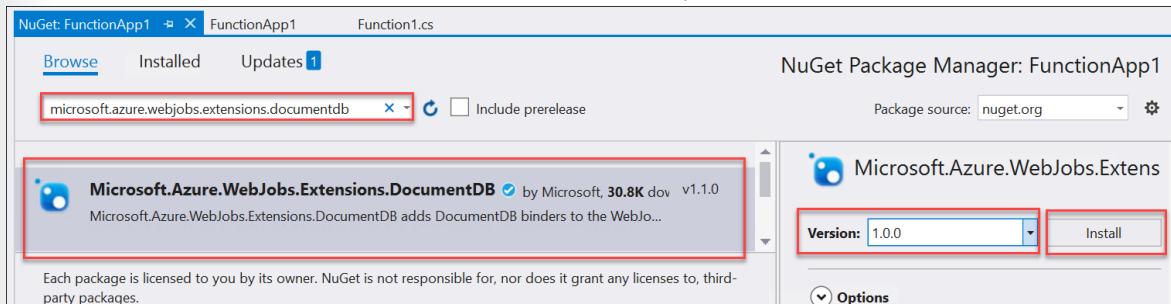


Step 2: Update Cosmos DB Document with Video Processing State

- With the **Function App** still open in Visual Studio, right-click on the **Function App**, then click **Manage NuGet Packages...**



- Search for, and install the **Microsoft.Azure.WebJobs.Extensions.DocumentDB** NuGet package. Be sure to select **Version 1.0.0** to install. Do NOT choose the latest version, as it will not install due to a version conflict.



- Open the **Function1.cs** file for the Azure Function code
- Modify the **Run** method declaration to include the following "dynamic inputDocument" parameter and **DocumentDB** attribute

```
[DocumentDB("learning", "videos", Id = "{documentId}", ConnectionStringSetting =
"contosovideodb_DOCUMENTDB")] dynamic inputDocument,
```

The first 2 parameters of the **DocumentDB** attribute define to connect to the "videos" Cosmos DB Collection within the "learning" database. And the value of "{documentId}" will enable it to retrieve the Document whose ID is set to the same value of the "documentId" value passed into the method via the HTTP POST. The "ConnectionStringSetting" parameter sets the name of the App Setting that will store the Cosmos DB Connection String.

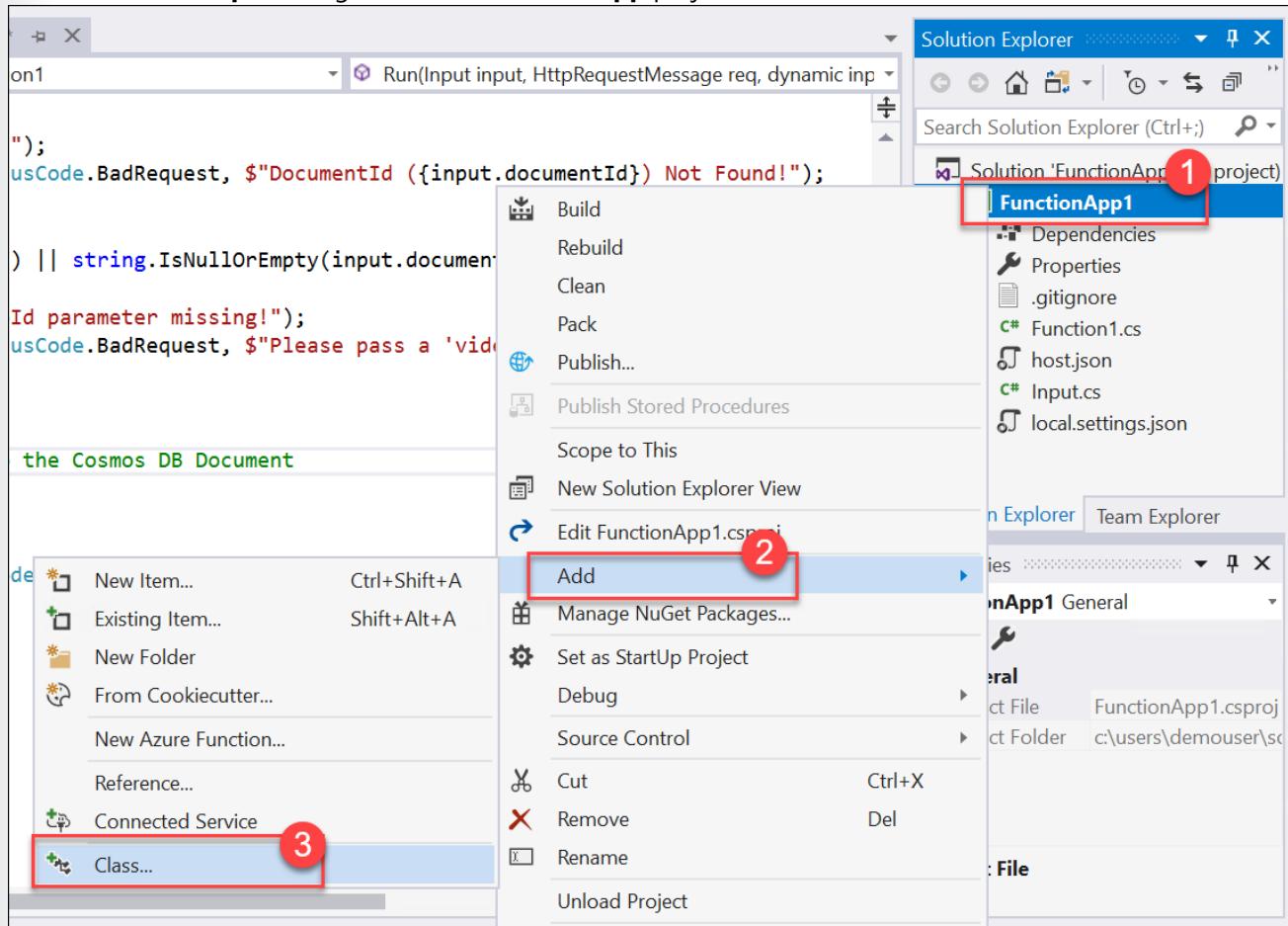
5. Now, the full **Run** method signature should look as follows:

```
public static async Task<object> Run(
    [HttpTrigger]Input input,
    HttpRequestMessage req,
    [DocumentDB("learning", "videos", Id = "{documentId}", ConnectionStringSetting =
    "contosovideodb_DOCUMENTDB")] dynamic inputDocument,
    TraceWriter log)
```

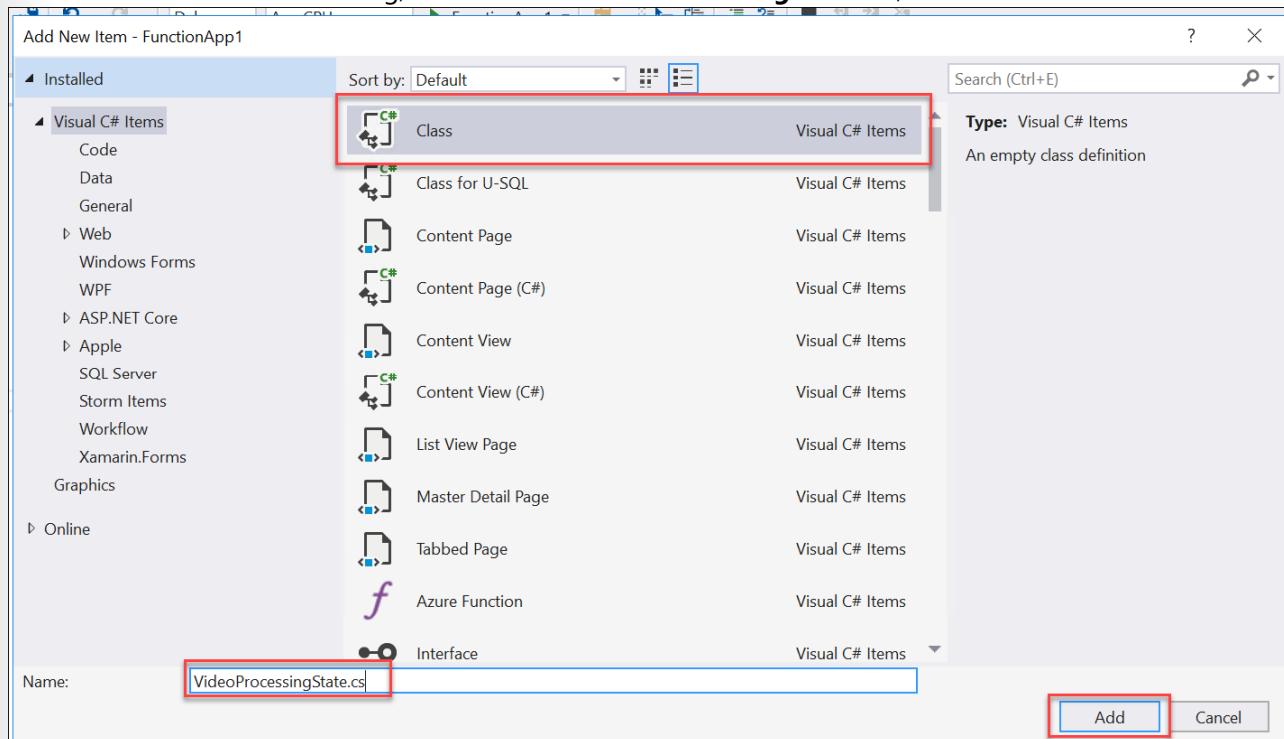
6. Immediately before the fine **return** statement of the **Run** method, insert the following line of code that will update the **videoId** property of the Cosmos DB Document to contain the **videoId** that is passed into the Function.

```
// Save the Video Indexer 'videoId' to the Cosmos DB Document
inputDocument.videoId = input.videoId;
```

7. Within **Solution Explorer**, right-click the **FunctionApp** project, then click **Add**, then **Class...**



8. Within the **Add New Item** dialog, name the file "**VideoProcessingState.cs**", then click **Add**



9. Replace the autogenerated **VideoProcessingState** class stub with the following code:

```
public class VideoProcessingState
{
    public string state { get; set; }
    public string progress { get; set; }

    public string ErrorType { get; set; }
    public string Message { get; set; }
}
```

10. Save the file

11. Open the **Function.cs** code file, and paste in the following **GetVideoProcessingState** method below the **Run** method:

```
private static async Task<VideoProcessingState> GetVideoProcessingState(string videoId, TraceWriter log)
{
    var client = new HttpClient();

    // Request headers
    string subscriptionKey =
System.Configuration.ConfigurationManager.AppSettings["VideoIndexerAPI_Key"];
    client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key", subscriptionKey);

    var uri = $"https://videobreakdown.azure-api.net/Breakdowns/Api/Partner/Breakdowns/{videoId}/State";

    var response = await client.GetAsync(uri);

    var content = await response.Content.ReadAsStringAsync();

    log.Info($"Processing State JSON: {content}");

    return JsonConvert.DeserializeObject<VideoProcessingState>(content);
}
```

12. Go back to the **Run** method, and add the following source code immediately below the "*inputDocument.videoId* = " line that will call the newly created function to load the Video Processing State

```
// Load Video Processing State
dynamic processingState = await GetVideoProcessingState(input.videoId, log);
```

13. Next, paste in the following code that checks if an Error Message was returned from the call to the Video Indexer API

```
if (!string.IsNullOrEmpty(processingState.ErrorType))
{
    log.Error($"{processingState.ErrorType}: {processingState.Message}");
    return req.CreateResponse(
        HttpStatusCode.InternalServerError,
        new { ErrorType = processingState.ErrorType, Message = processingState.Message }
    );
}
```

14. Next, paste in the following code that sets the values of the **Processing State** and **Processing Progress** to properties on the **Cosmos DB Document**; as well as logs their values to the Azure Function Logs

```
log.Info($"Video Processing State: {processingState.state}");
log.Info($"Video Processing Progress: {processingState.progress}");

// Save Video Processing State in Cosmos DB Document
inputDocument.processingState = processingState.state;
inputDocument.processingProgress = processingState.progress;
```

15. The final **Run** method, should look as follows:

```
[FunctionName("Function1")]
public static async Task<object> Run(
    [HttpTrigger]Input input,
    HttpRequestMessage req,
    [DocumentDB("learning", "videos", Id = "{documentId}", ConnectionStringSetting =
    "contosovideodb_DOCUMENTDB")] dynamic inputDocument,
    TraceWriter log)
{
    log.Info("Function triggered...");

    // Log the parameters passed in through the Request Body
    log.Info($"DocumentId: {input.documentId}");
    log.Info($"VideoId: {input.videoId}");

    if (inputDocument == null)
    {
        log.Error("'documentId' not found!");
        return req.CreateResponse(HttpStatusCode.BadRequest, $"DocumentId ({input.documentId}) Not
        Found!");
    }

    if (string.IsNullOrEmpty(input.videoId) || string.IsNullOrEmpty(input.documentId))
    {
        log.Error("DocumentId and/or VideoId parameter missing!");
        return req.CreateResponse(HttpStatusCode.BadRequest, $"Please pass a 'videoId' and
        'documentId' in the Http request body");
    }

    // Save the Video Indexer 'videoId' to the Cosmos DB Document
    inputDocument.videoId = input.videoId;

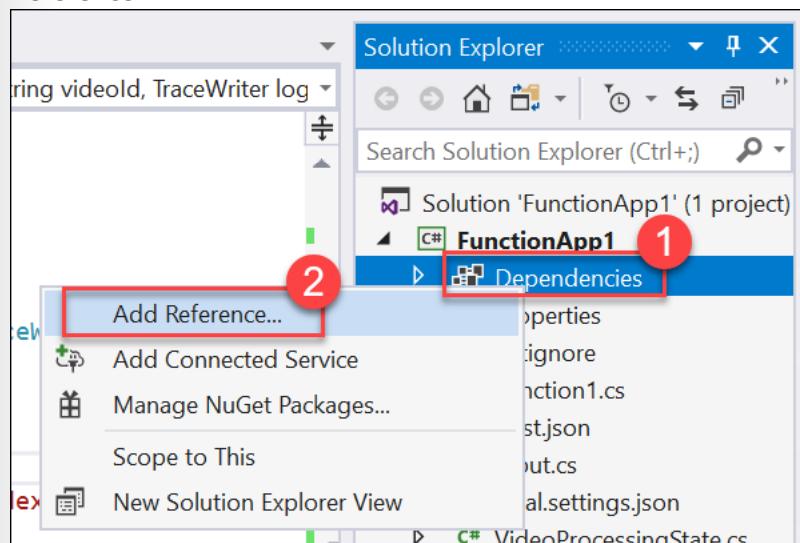
    // Load Video Processing State
    dynamic processingState = await GetVideoProcessingState(input.videoId, log);

    if (!string.IsNullOrEmpty(processingState.ErrorType))
```

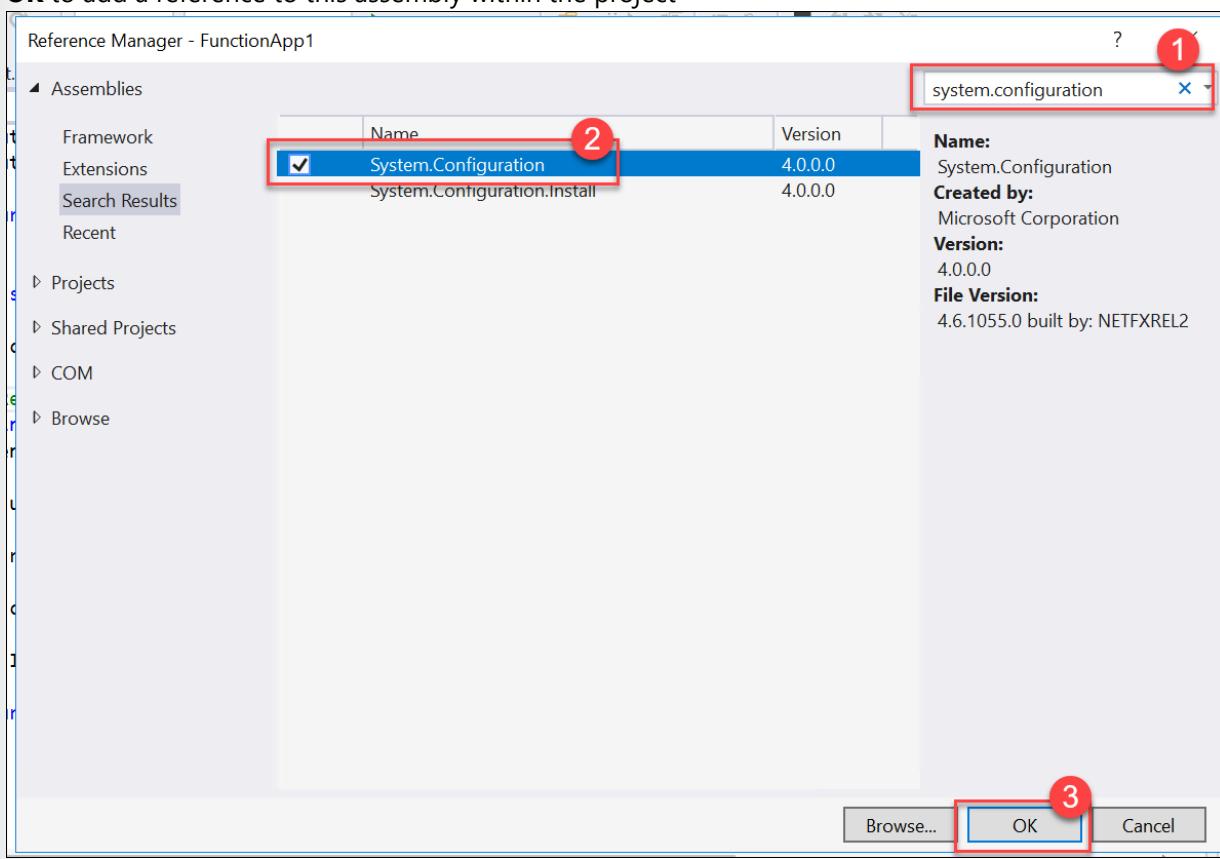
```
{  
    log.Error($"{processingState.ErrorType}: {processingState.Message}");  
    return req.CreateResponse(  
        HttpStatusCode.InternalServerError,  
        new { ErrorType = processingState.ErrorType, Message = processingState.Message }  
    );  
}  
  
log.Info($"Video Processing State: {processingState.state}");  
log.Info($"Video Processing Progress: {processingState.progress}");  
  
// Save Video Processing State in Cosmos DB Document  
inputDocument.processingState = processingState.state;  
inputDocument.processingProgress = processingState.progress;  
  
return req.CreateResponse(HttpStatusCode.OK, "Success");  
}
```

16. Save the file

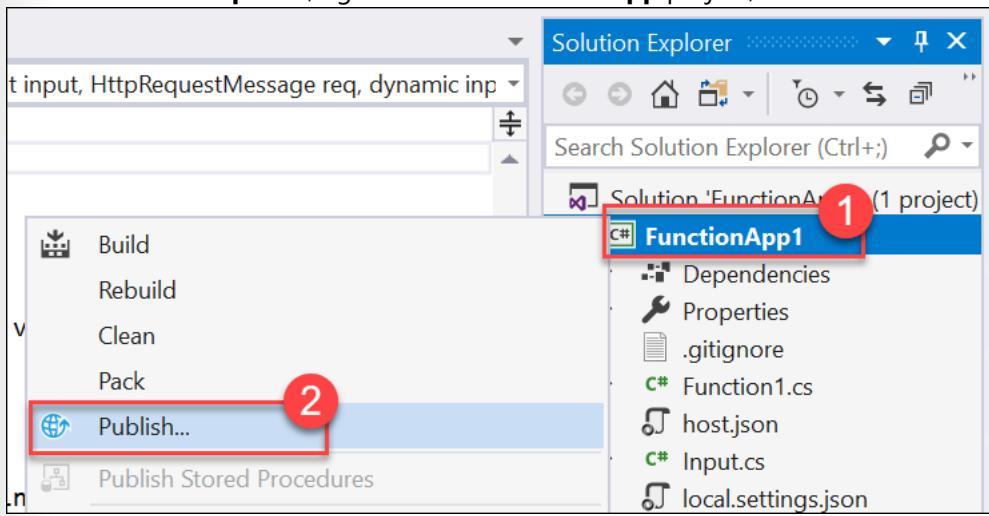
17. Within **Solution Explorer**, right-click the **Dependencies** node for the **Function App** project, then click on **Add Reference**



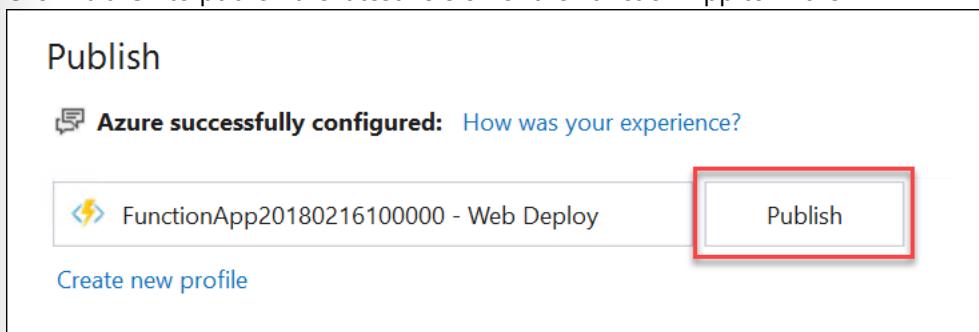
18. In the **Reference Manager** dialog, search for and locate the **System.Configuration** assembly, select it, then click **OK** to add a reference to this assembly within the project



19. Within **Solution Explorer**, right-click the **Function App** project, then click **Publish...**



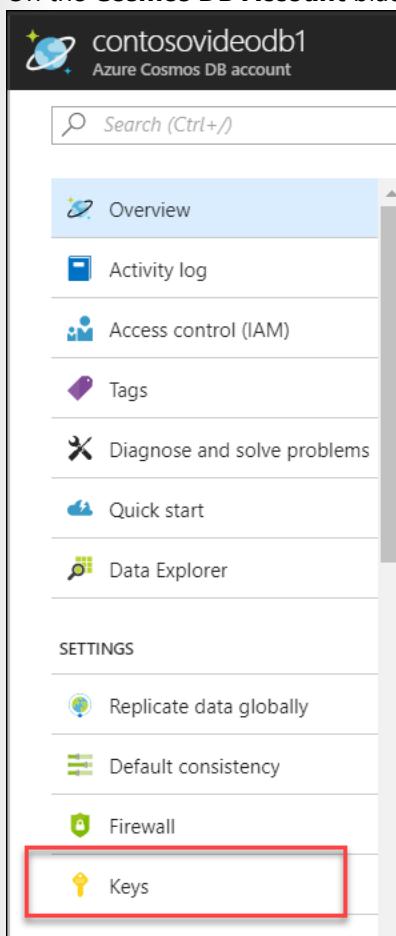
20. Click **Publish** to publish the latest version of the Function App to Azure



21. Wait for the Publish to complete, this should take about 1 minute

22. Open the **Azure Portal**, and navigate to the **Cosmos DB Account** that was previously created

23. On the **Cosmos DB Account** blade, click on **Keys**



24. Copy the **PRIMARY CONNECTION STRING** for the Cosmos DB Account

The screenshot shows the 'Keys' tab of a Cosmos DB account. It displays two keys: PRIMARY KEY and SECONDARY KEY. Below them are the PRIMARY CONNECTION STRING and SECONDARY CONNECTION STRING. The PRIMARY CONNECTION STRING is highlighted with a red box.

| URI |
|---|
| https://contosovideodb1.documents.azure.com:443/ |

| PRIMARY KEY |
|--|
| qHDBJfE4VdzdQP6MqPV7RHfxclnoec4YjB7u7AD9Ad6SjqMRYYr3gpsYpGUJrrwz2KrlxFaAqWiMOAIYTxrJ0g== |

| SECONDARY KEY |
|--|
| UpghORolqUzlpU7sAwFyQPVRSCZ3BTWqU3Omnnd7B7LeWKSU7skyfla08EsV3Ls3ilsVjYt0GKULmUBugQYLUQ== |

| PRIMARY CONNECTION STRING |
|--|
| AccountEndpoint=https://contosovideodb1.documents.azure.com:443/;AccountKey=qHDBJfE4VdzdQP6MqPV7RHfxclnoec4YjB7u7AD9Ad6SjqMRYYr3gpsYpGU... |

| SECONDARY CONNECTION STRING |
|--|
| AccountEndpoint=https://contosovideodb1.documents.azure.com:443/;AccountKey=UpghORolqUzlpU7sAwFyQPVRSCZ3BTWqU3Omnnd7B7LeWKSU7skyfla08... |

25. Navigate to the **ContosoVideo** Resource Group, then navigate to the **Azure Function** that was created and published from Visual Studio

The screenshot shows the 'Overview' page of the 'ContosoVideo' resource group. It lists various resources including storage accounts, a web admin plan, and several functions. One function, 'FunctionApp20180216100000', is highlighted with a red box.

| Subscription (change) |
|-----------------------|
| Super Awesome Cloud |

| Subscription ID |
|--------------------------------------|
| 41850220-19f0-49c6-87ea-766c9b0da1f1 |

| Filter by name... |
|-------------------|
| All types |

| NAME |
|---------------------------------------|
| ac522a1fb7735437ea9d614e |
| azureblob |
| ContosoLearningWebAdmin20180216090352 |
| ContosoLearningWebAdminPlan |
| contosovideo |
| contosovideo1 |
| contosovideodb1 |
| FunctionApp20180216100000 |
| FunctionApp20180216100000Plan |
| videoindexer |

26. On the **Azure Function** blade, click on **Application settings** under the Configured features section

The screenshot shows the Azure Functions blade for a function app named 'FunctionApp20180216100000'. The left sidebar lists 'Functions (Read Only)' and 'Function1'. The main area shows the 'Overview' tab with the status 'Running' and subscription information. Below this is the 'Configured features' section, which includes 'Function app settings' and 'Application settings', with 'Application settings' highlighted by a red box.

27. Scroll down, and add a new **Application setting** with the following values:

- Name: **contosovideodb_DOCUMENTDB**
- Value: **paste in the Cosmos DB Account Connection String that was copied**

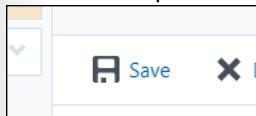
The screenshot shows the 'Application settings' blade. A new setting is being added, with the name 'contosovideodb_DOCUMENTDB' and the value 'AccountEndpoint=https://contosovideodb1.documents.azure.com:443/;AccountKey=qH...' highlighted by red boxes.

28. Add another **Application setting** with the following values:

- Name: **VideoIndexerAPI_Key**
- Value: **paste in the Video Indexer API Key that was copied previously**

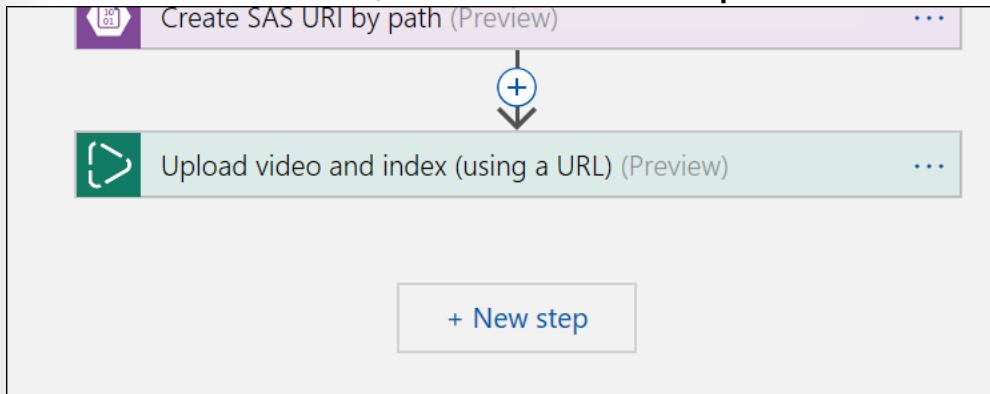
The screenshot shows the 'Application settings' blade. Another new setting is being added, with the name 'VideoIndexerAPI_Key' and the value 'bafa7860d1b44a239ee2baad6f0f0be9' highlighted by red boxes.

29. Scroll back up and click **Save**

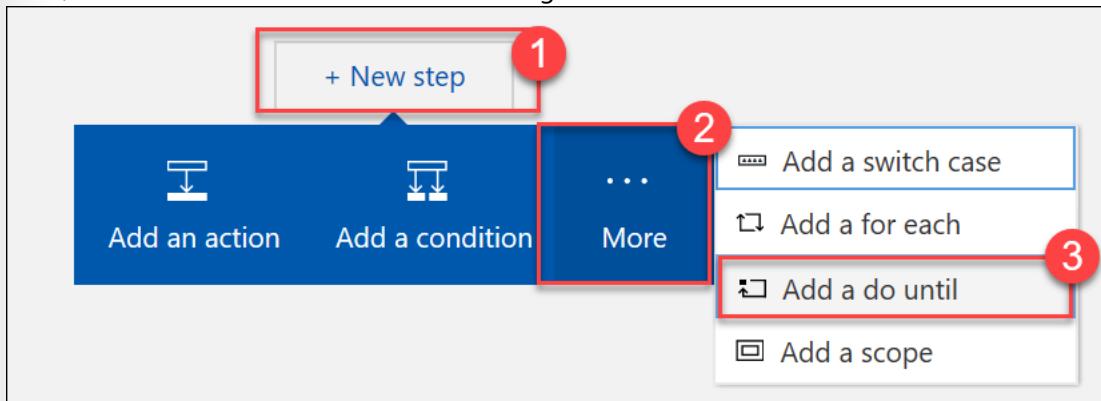


Step 3: Update Video State when processing is complete

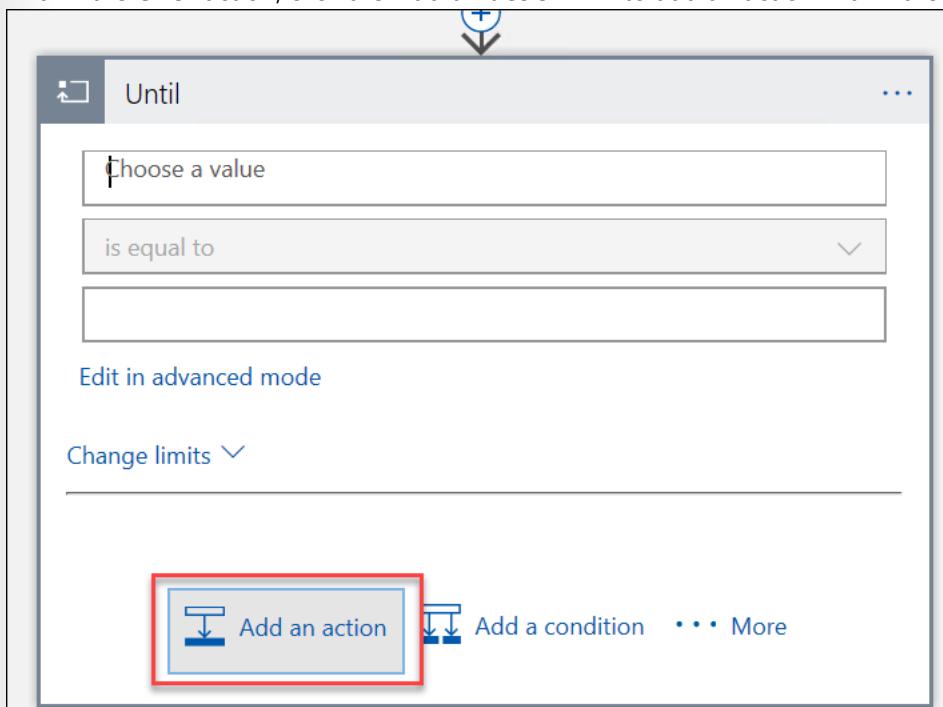
1. Open the **Logic App** within Visual Studio that was previously created
2. Locate the end of the Workflow, after the **Video Indexer – Upload video and index (using a URL)** action



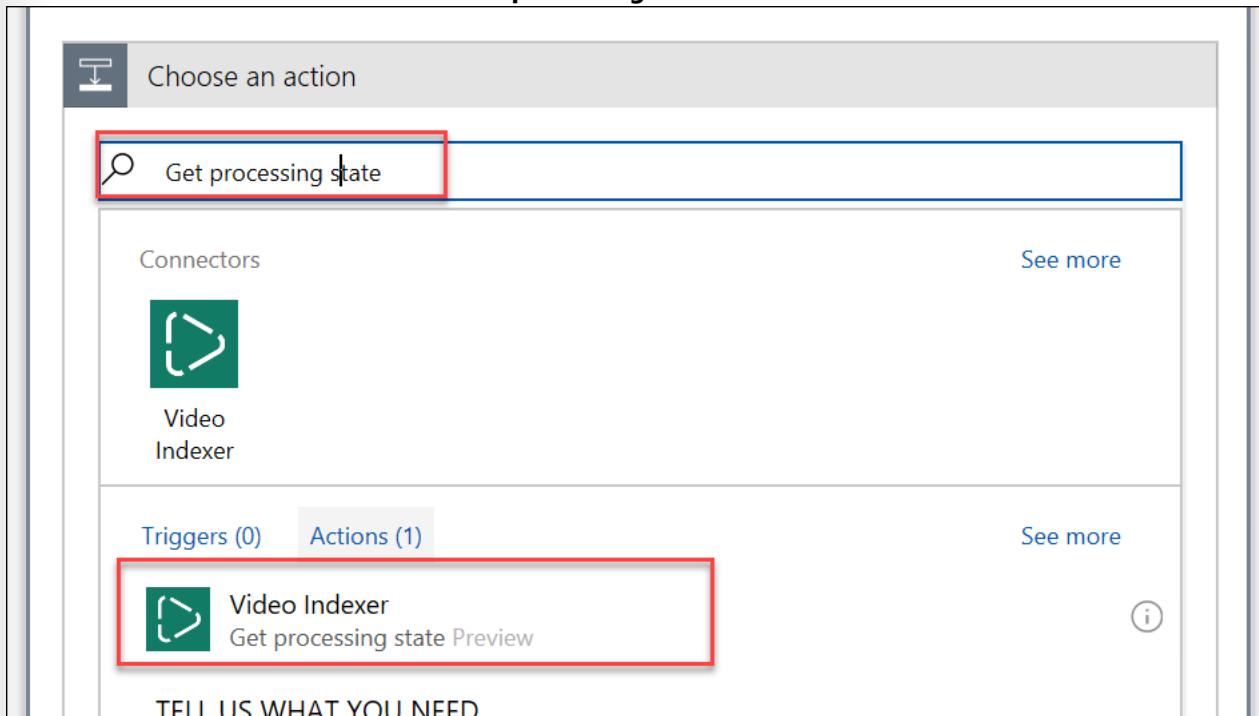
3. Click **+New step**, then **More**, then **Add a do until**. This will be used to periodically check the video processing state, and wait until it's finished before moving on with the workflow.



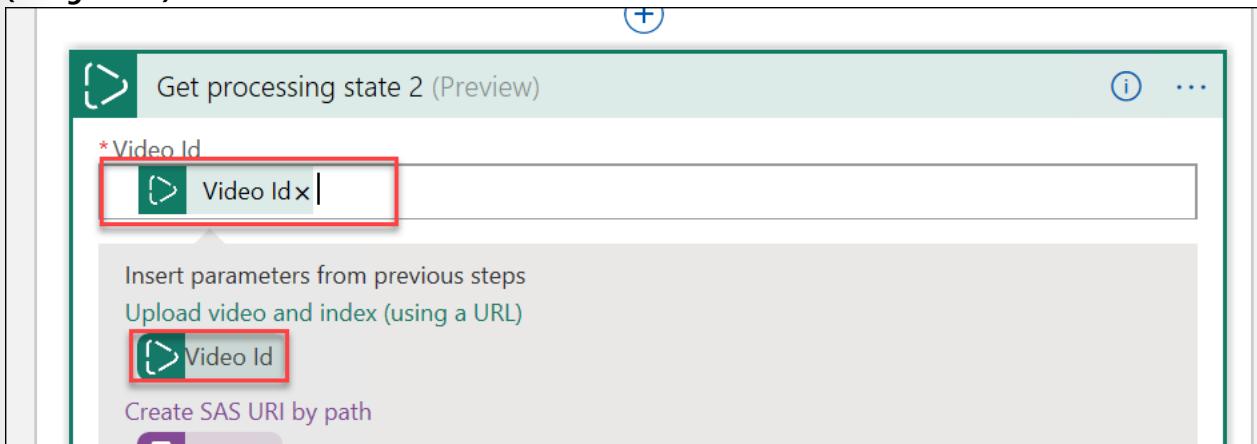
4. Within the **Until** action, click the **Add an action** link to add an action within the "do until" block



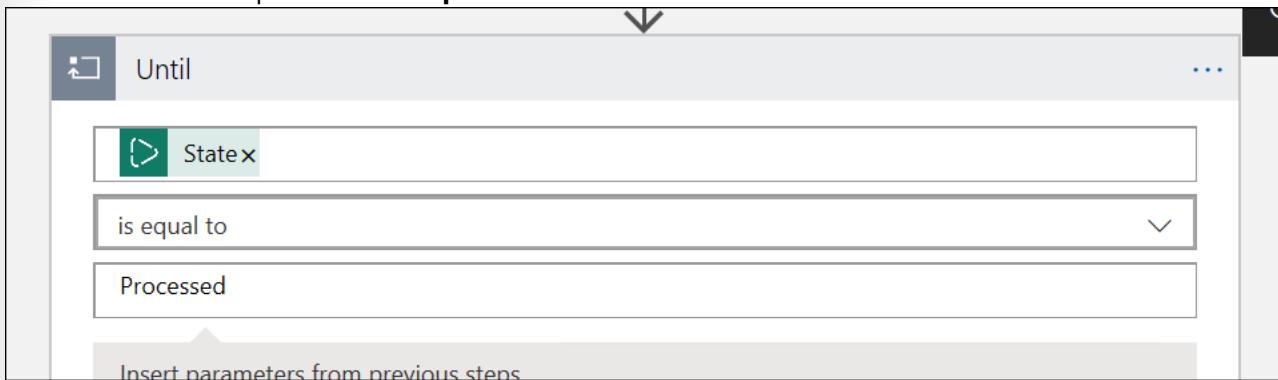
5. Search for and add a **Video Indexer – Get processing state** action



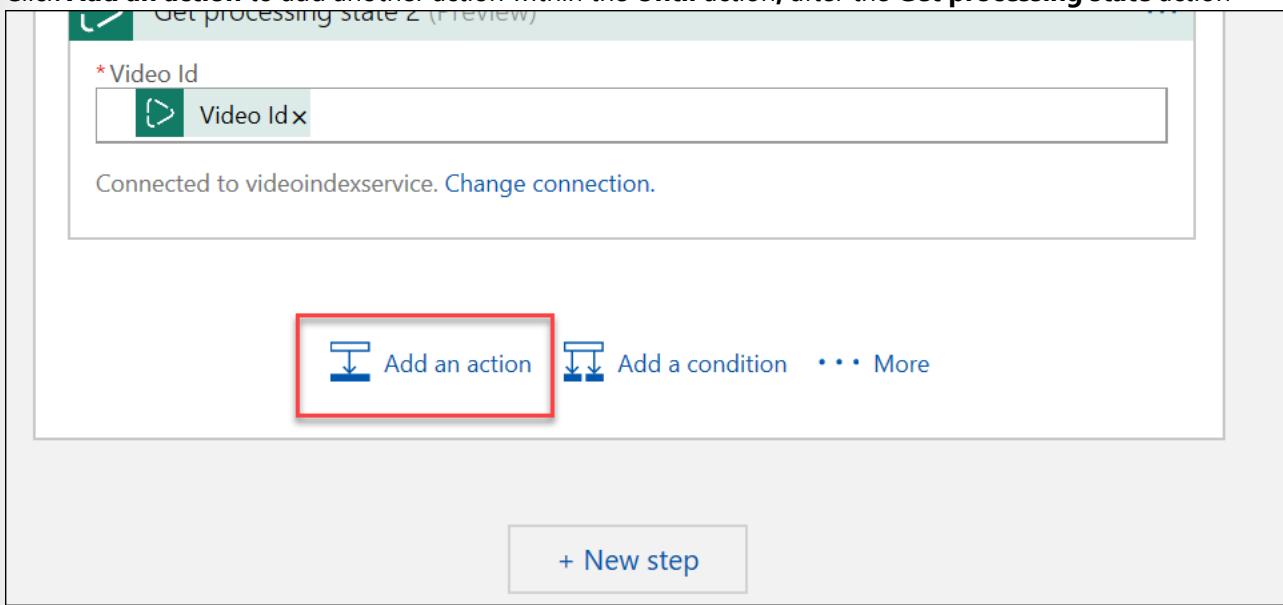
6. On the **Get processing state** action, enter the **Video Id** parameter value from the **Upload video and index (using a URL)** action into the **Video Id** field



7. At the top of the **Until** action, set the check condition to look at the **State** parameter from the **Get processing state** action and compare that it **is equal to** the value of **Processed**

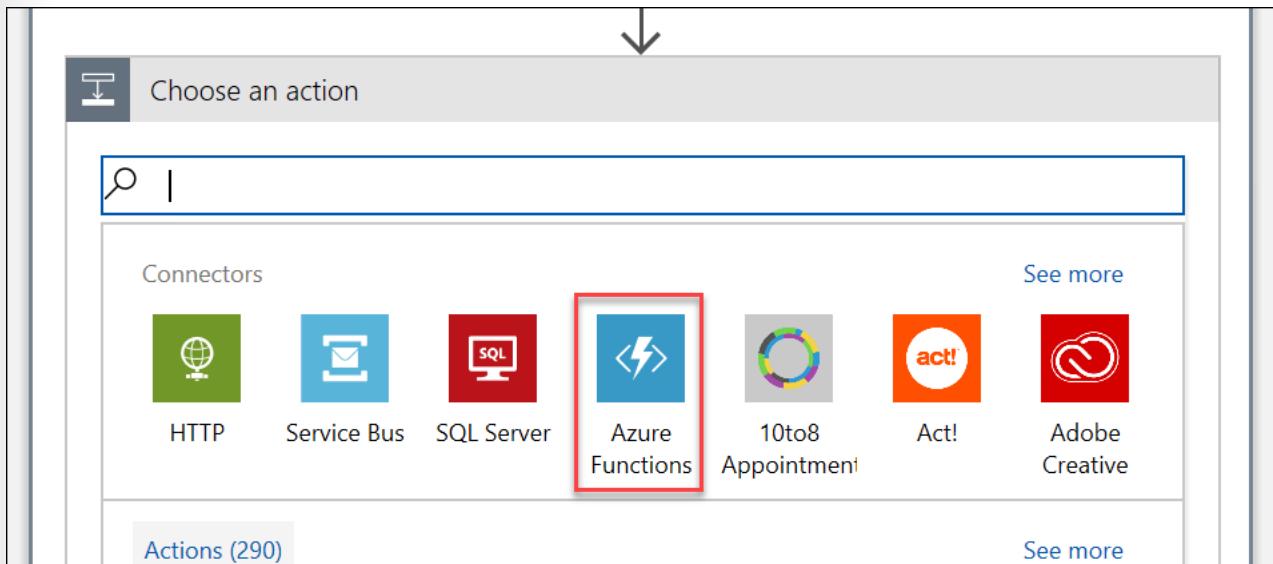


8. Click **Add an action** to add another action within the **Until** action, after the **Get processing state** action

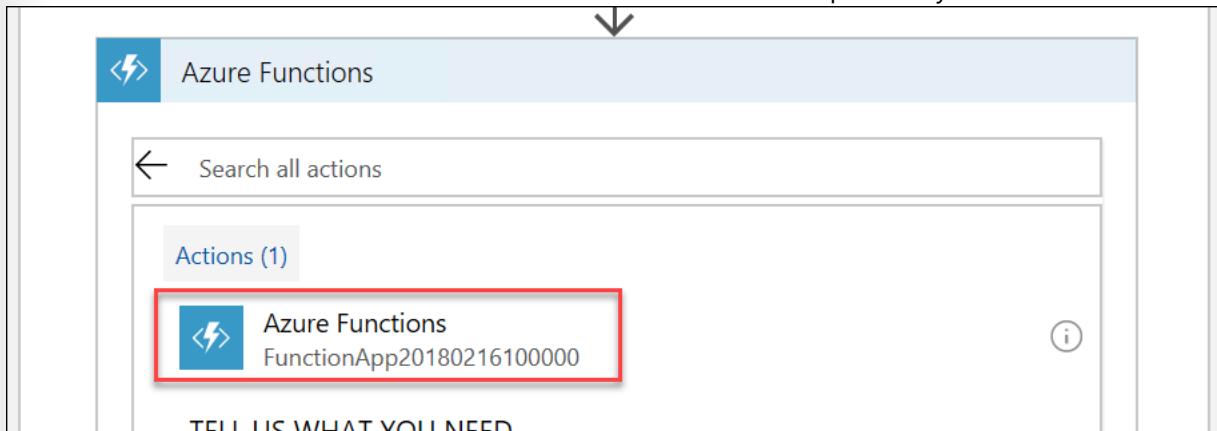


NOTE: The Processing State is being polled here so it can easily be added to the database so the percentage complete can be displayed to the end-user within the web app more easily. Without this feature, the best practice would be to configure a callback with the initial Video Indexer call, so it can asynchronously notify when processing is completed.

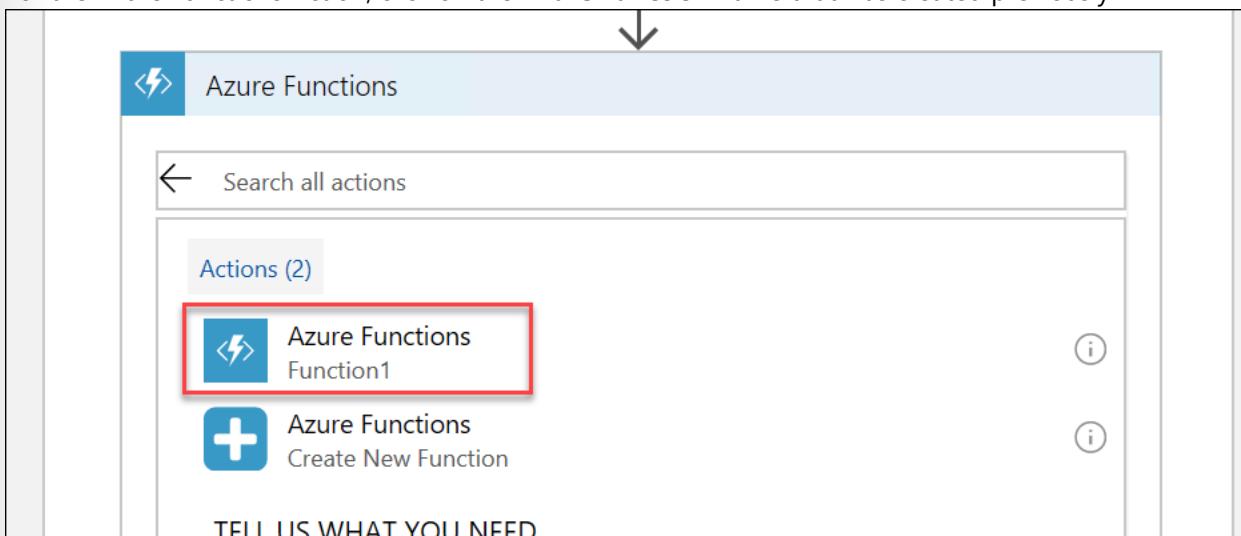
9. Click on the **Azure Functions Connector**



10. Click on the **Azure Functions** Action for the Azure Functions that was previously created



11. For the Azure Functions Action, click on the **Azure Function** name that was created previously

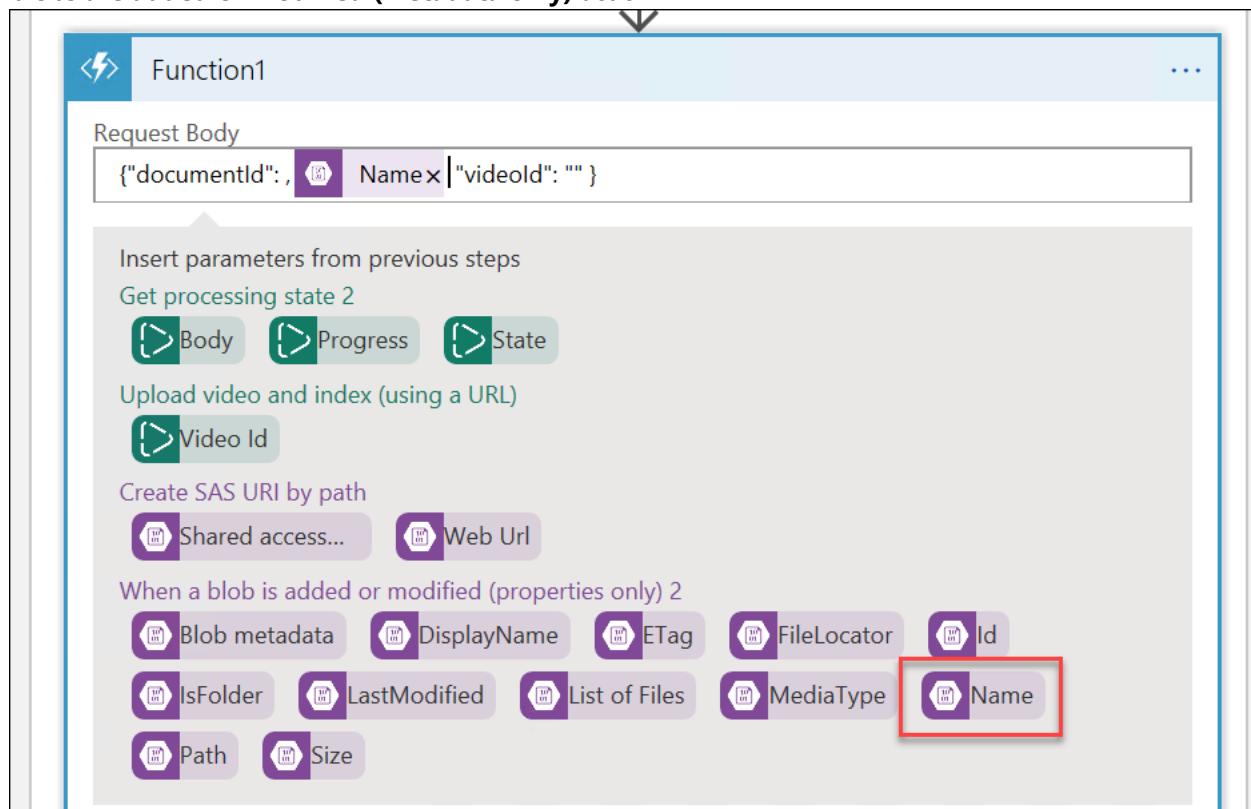


12. Update the **Request Body** field for the Azure Functions Action to contain a JSON object that includes **documentId** and **videoId** values

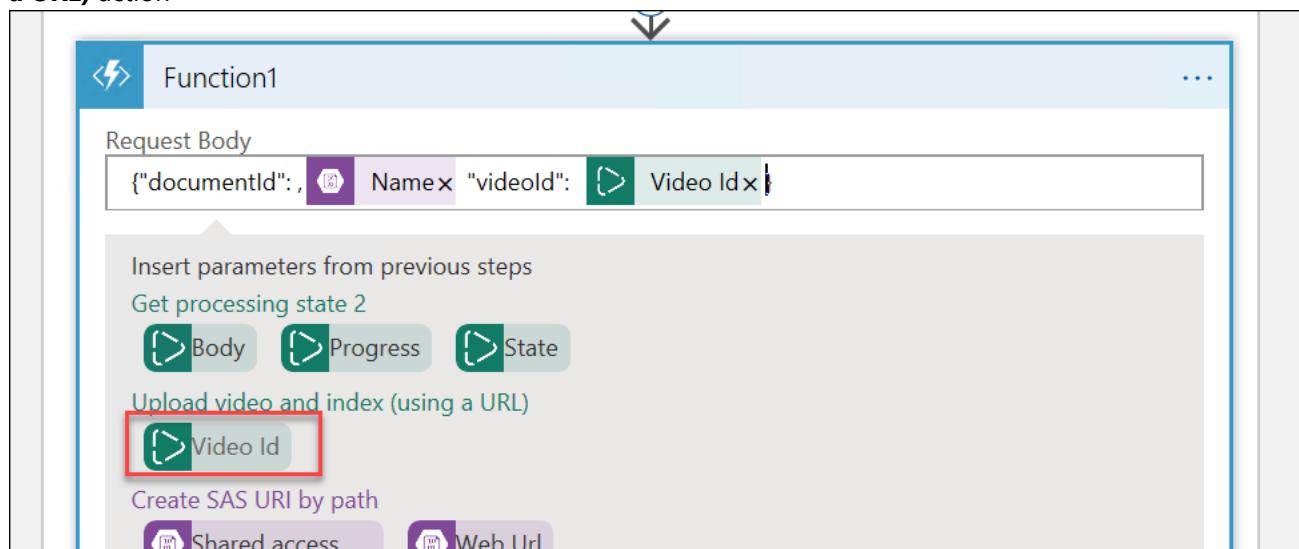
The screenshot shows the 'Request Body' configuration for the 'Function1' action. The JSON object is currently set to `{"documentId": "", "videoId": "" }`. A downward arrow points from the instruction text above to the 'Request Body' input field.

13. Modify the JSON properties to have the following values. Also, be sure to remove the empty double quotes ("") from the JSON when adding the new property values as shown below.

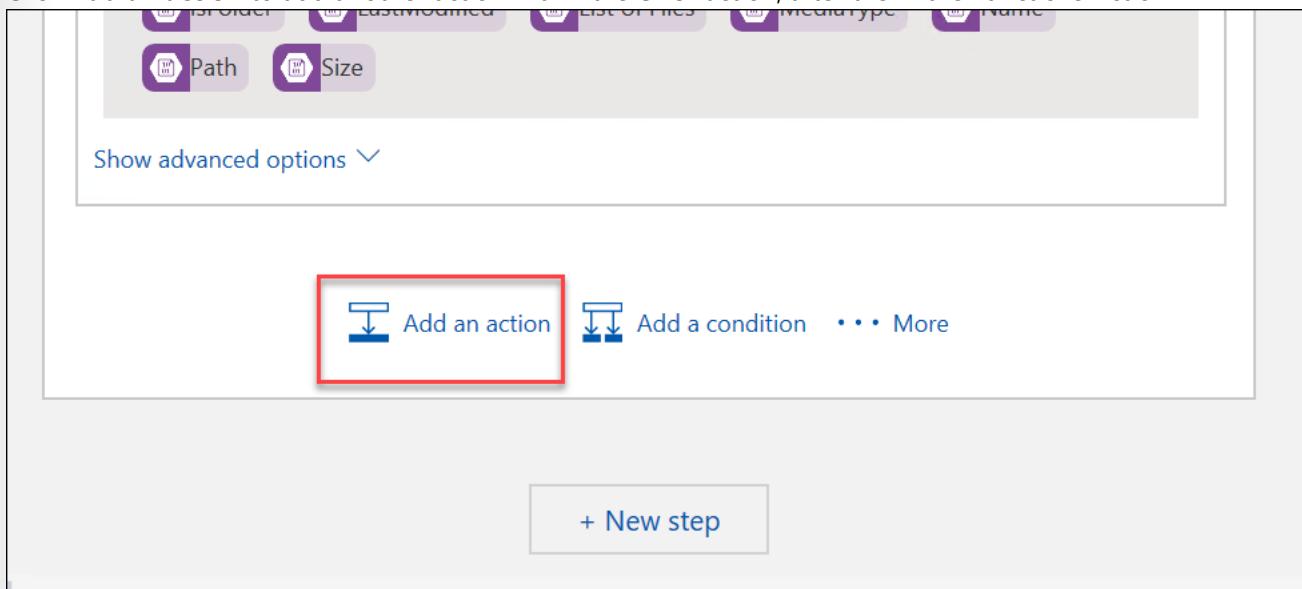
- a. Set the **documentId** property to the **Name** parameter from the **Blob Storage – When one or more blobs are added or modified (metadata only)** action



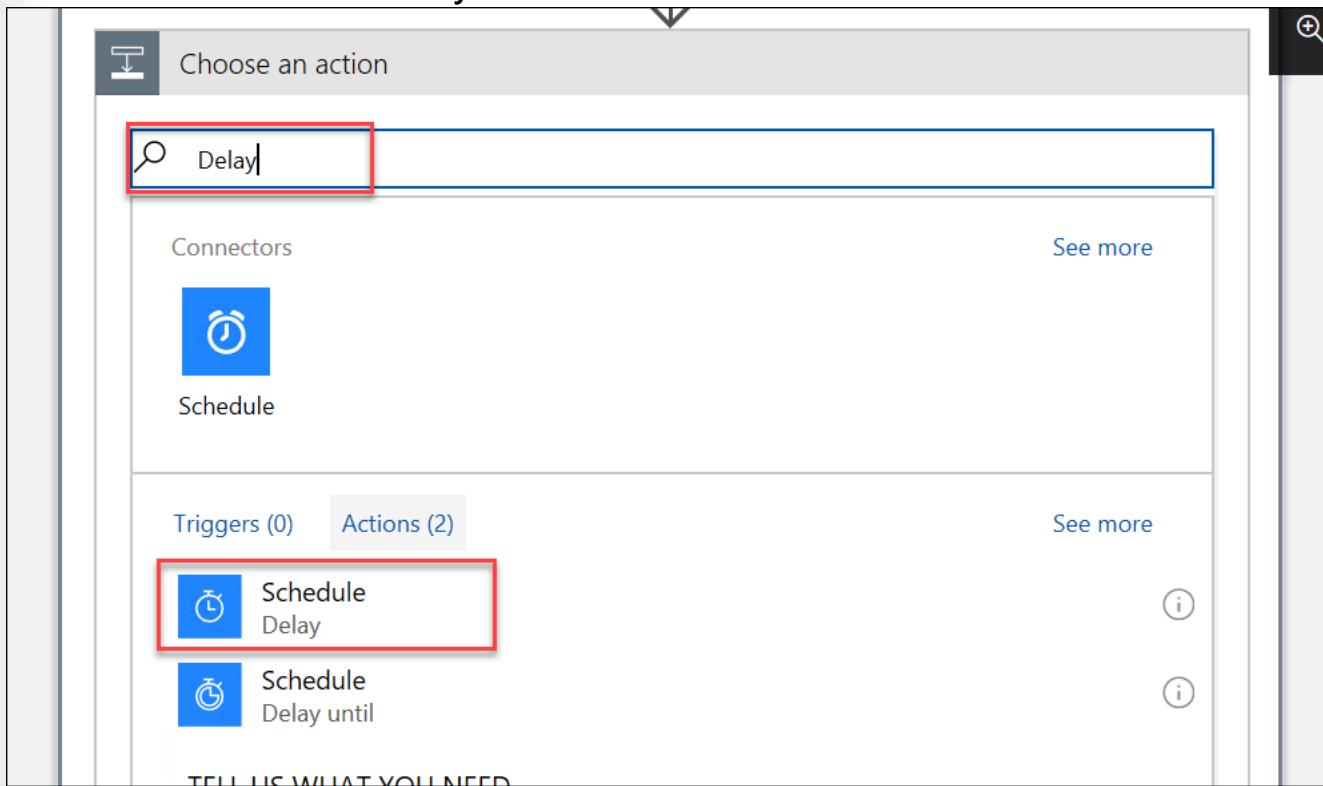
- b. Set the **videoid** property to the **Video Id** parameter from the **Video Indexer – Upload and index (using a URL)** action



14. Click **Add an action** to add another action within the **Until** action, after the Azure Functions Action

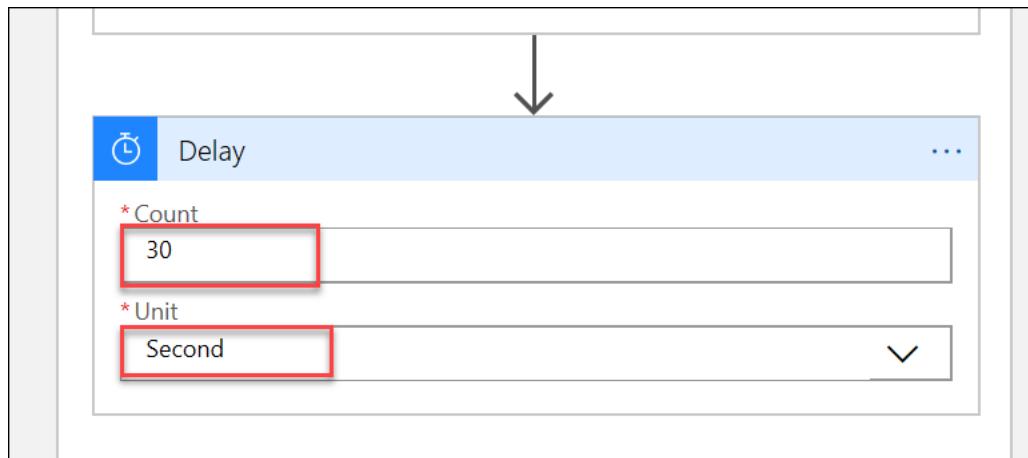


15. Search for and add a **Schedule – Delay** action

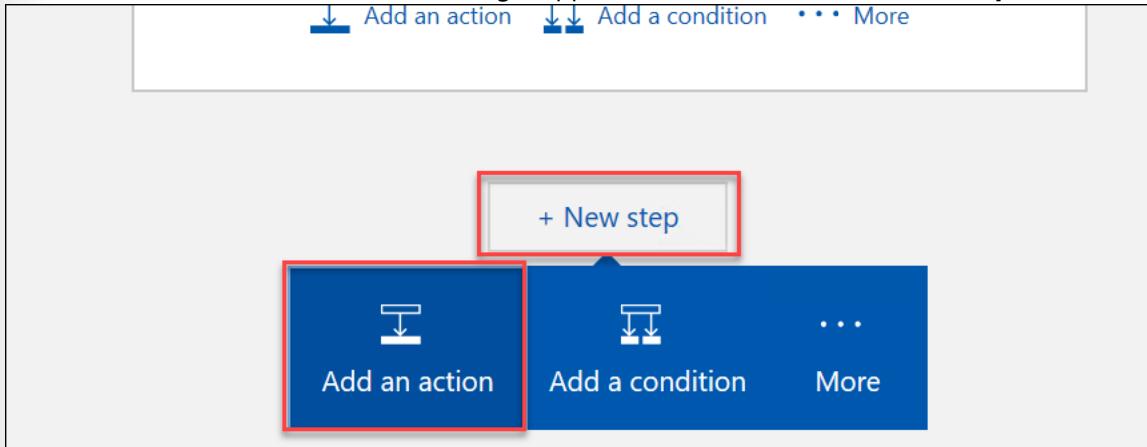


16. On the **Delay** action, enter the following values:

- Count: **30**
- Unit: **Second**



17. Scroll down to the bottom / end of the Logic App Workflow, and click the **+New step**, then click **Add an action**



18. Add another **Azure Functions Action** here that is configured identical to the one previously created within the **Until** loop action. The reason for this is the first one within the **Until** action will periodically update the status of the Video Processing within the Cosmos DB Document with each iteration of the loop. This new Azure Function at the end of the Logic App Workflow will update the Video Processing State one final time before the Workflow finished execution.
19. **Save** the Logic App
20. **Deploy** the updated Logic App to Azure

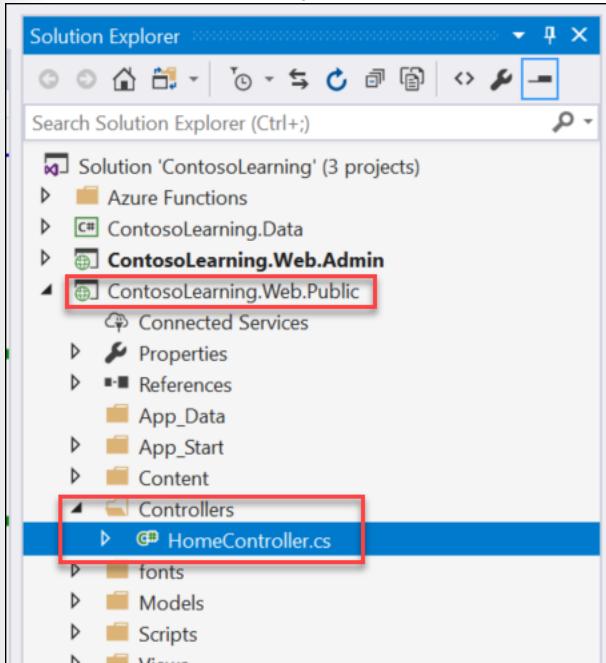
Exercise 5: Add Video Player to Front-End Application

Duration: 30 minutes

In this exercise, you will extend the Front-End Application foundation to include a video player and Cognitive Services Insights for the Videos.

Step 1: Integrate Cosmos DB into Front-End Application

1. Open the **ContosoLearning.sln** solution within Visual Studio 2017
2. Within Solution Explorer, locate and expand the **ContosoLearning.Web.Public** project, then expand the **Controllers** folder, and open the **HomeController.cs** file



3. Within the **HomeController** class, locate the **Index()** Action method, and replace the methods contents with the following code that uses the **VideoRepository** to load all the Videos from the Cosmos DB Collection and returns the data in the Model so the view can display it in the UI.

```
var model = new HomeIndexModel();

var videoRepo = VideoRepositoryFactory.Create();

model.Videos = (from v in await videoRepo.GetAll()
                orderby v.Title, v.Created
                select new VideoListModel
                {
                    Video = v
                }).ToArray();

return View(model);
```

4. Locate the **Video(string id)** method. This Action method is used to display the video player for individual videos. Replace the contents of this method with the following code that loads the Video info from the Cosmos DB Collection.

```
var model = new HomeVideoModel();

var courseRepo = VideoRepositoryFactory.Create();
model.Video = await courseRepo.Get(id);

if (model.Video == null)
{
    throw new Exception("Video not found!");
}

// Get Access Token
var client = new System.Net.Http.HttpClient();
client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
System.Configuration.ConfigurationManager.AppSettings["VideoIndexerAPI_Key"]);

var uriResponse = await client.GetAsync($"https://videobreakdown.azure-
api.net/Breakdowns/Api/Partner/Breakdowns/{model.Video.VideoId}/GetAccessToken");
var jsonUriResponse = await uriResponse.Content.ReadAsStringAsync();

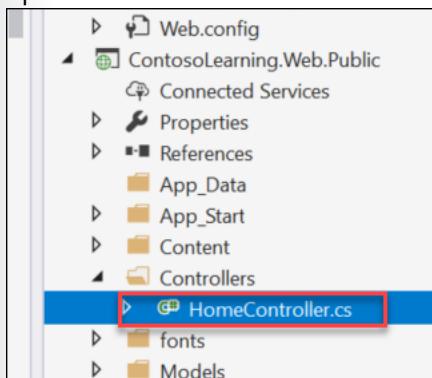
model.AccessToken = jsonUriResponse.Replace("\\\"", string.Empty);

return View(model);
```

5. Save the file.

Step 2: Display Video Thumbnail Image

1. Within Solution Explorer, expand the **Controllers** folder within the **ContosoLearning.Web.Public** project, then open the **HomeController.cs** file



2. Locate the **Index()** action method within the **HomeController** class

3. Paste in the following code immediately before the **return** statement in the **Index()** method. This code will loop through all the Videos and load the **Video Thumbnail URL** for each by calling the **Video Indexer API**.

```
var client = new System.Net.Http.HttpClient();
var queryString = HttpUtility.ParseQueryString(string.Empty);

// Request headers
client.DefaultRequestHeaders.Add("Ocp-Apim-Subscription-Key",
System.Configuration.ConfigurationManager.AppSettings["VideoIndexerAPI_Key"]);

// Request parameters
queryString["language"] = "{string}";

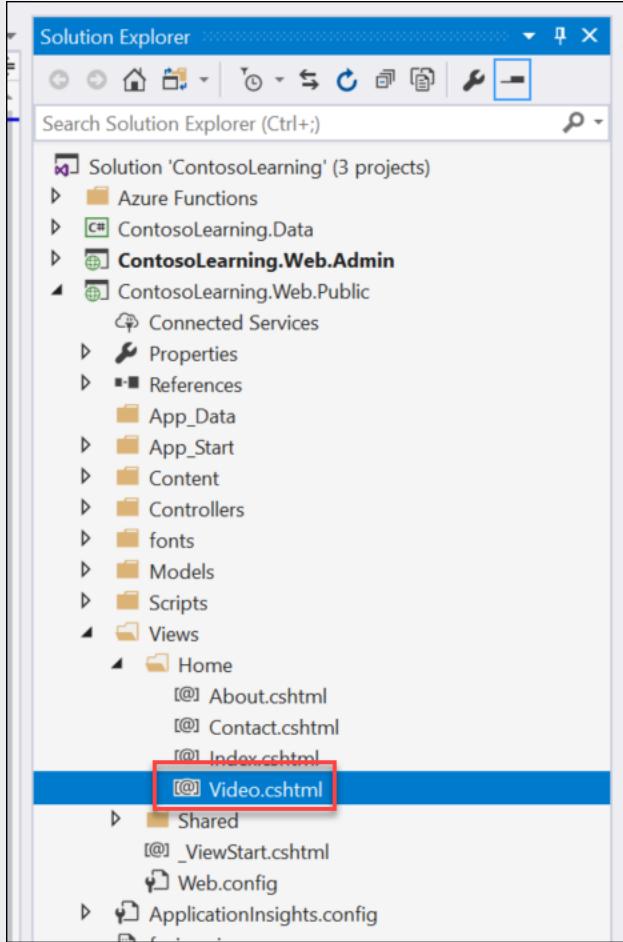
foreach (var v in model.Videos) {
    var uri = $"https://videobreakdown.azure-
api.net/Breakdowns/Api/Partner/Breakdowns/{v.Video.VideoId}";
    var response = await client.GetAsync(uri);
    var json = await response.Content.ReadAsStringAsync();

    dynamic breakdown = Newtonsoft.Json.Linq.JObject.Parse(json);

    v.ThumbnailUrl = breakdown?.summarizedInsights?.thumbnailUrl;
}
```

Step 3: Add Video Player

1. Within Solution Explorer, locate and expand the **Views/Home** folder within the **ContosoLearning.Web.Public** project, and open the **Video.cshtml** file



2. Locate the "[Video Here]" placeholder text within the **Video.cshtml** view

```
18     <div class="col-lg-6">
19
20         [Video Here]
21
22     </div>
```

3. Replace the placeholder text with the following code that will include the **Video Player** within an IFrame. Notice the **VideoId** property from the Video is appended to the URL within the IFrame to tell Video Indexer which video to play.

```
<iframe width="560" height="315"
src="https://www.videobreakdown.com/embed/player/@(Model.Video.VideoId)?accessToken=@(Model.AccessToken)" frameborder="0" allowfullscreen></iframe>
```

Step 4: Add Video Insights

1. Within the **Video.cshtml** file, locate the "[Insights Here]" placeholder text

```
23     <div class="col-lg-6">
24
25         [Insights Here]
26
27     </div>
```

2. Replace the placeholder text with the following code that will include the **Video Insights** within an IFrame. Notice the **VideoId** property from the Video is appended to the URL within the IFrame to tell Video Indexer which video to display Insights for.

```
<iframe style="width: 100%; height: 60em;"  
src="https://www.videobreakdown.com/embed/insights/@(Model.Video.VideoId)?accessToken=@(Model.AccessToken)" frameborder="0" allowfullscreen="true"></iframe>
```

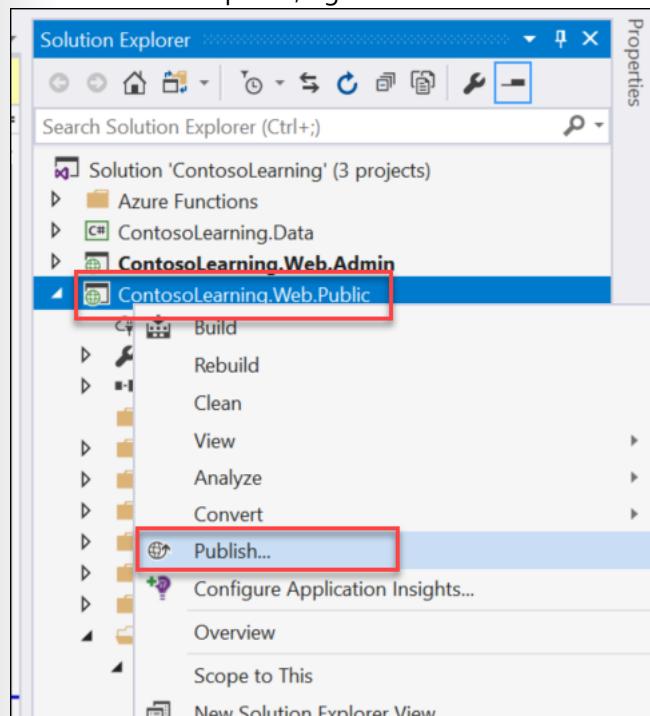
Step 5: Integrate Video Player and Insights together

1. As coded previously, the Video Player and Insights will display, but are disconnected
2. To connect the Video Player and Insights so the insights can update as the video plays, add the following JavaScript file include to the bottom of the **Video.cshtml** View below the IFrame code:

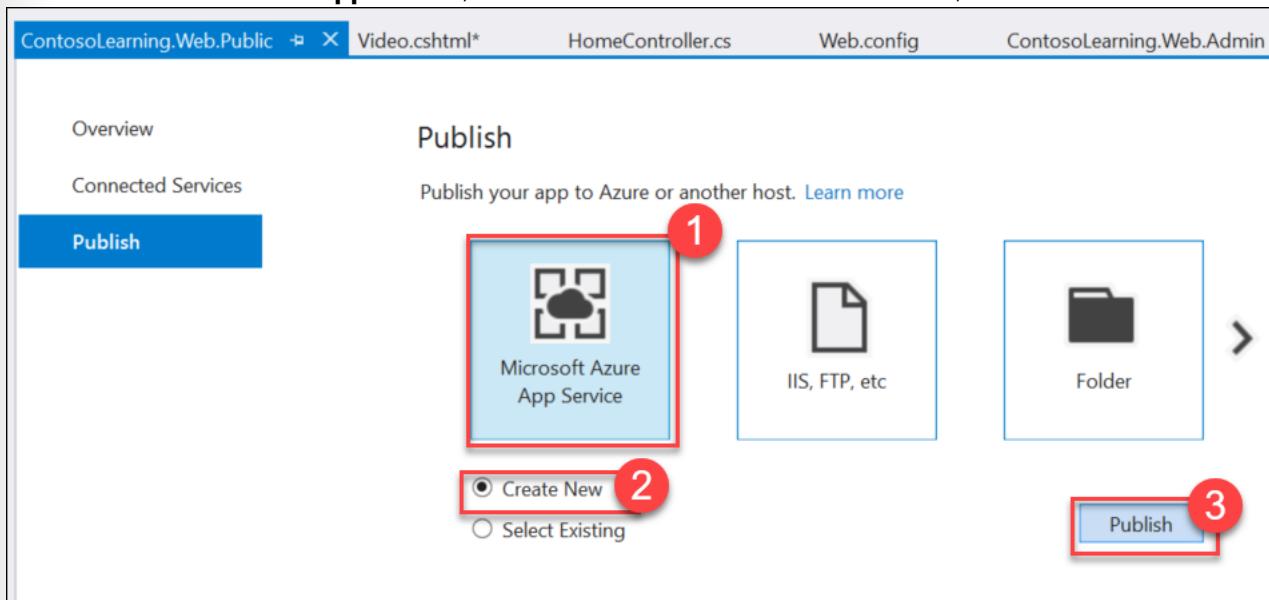
```
<script src="https://breakdown.blob.core.windows.net/public/vb.widgets.mediator.js"></script>
```
3. Save the file

Step 6: Deploy Public website to an Azure Web App

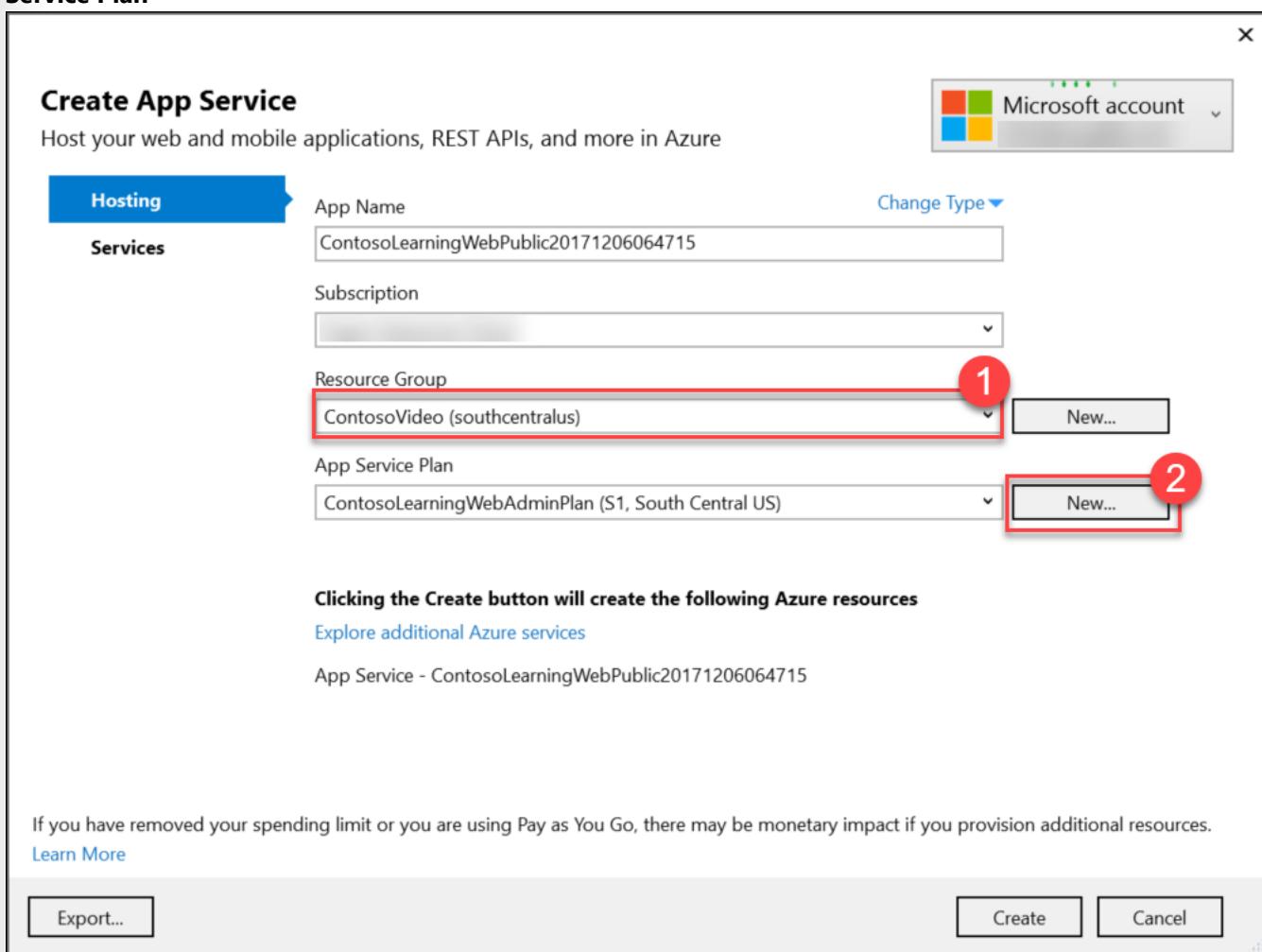
1. Within Solution Explorer, right-click the **ContosoLearning.Web.Public** project, then click **Publish...**



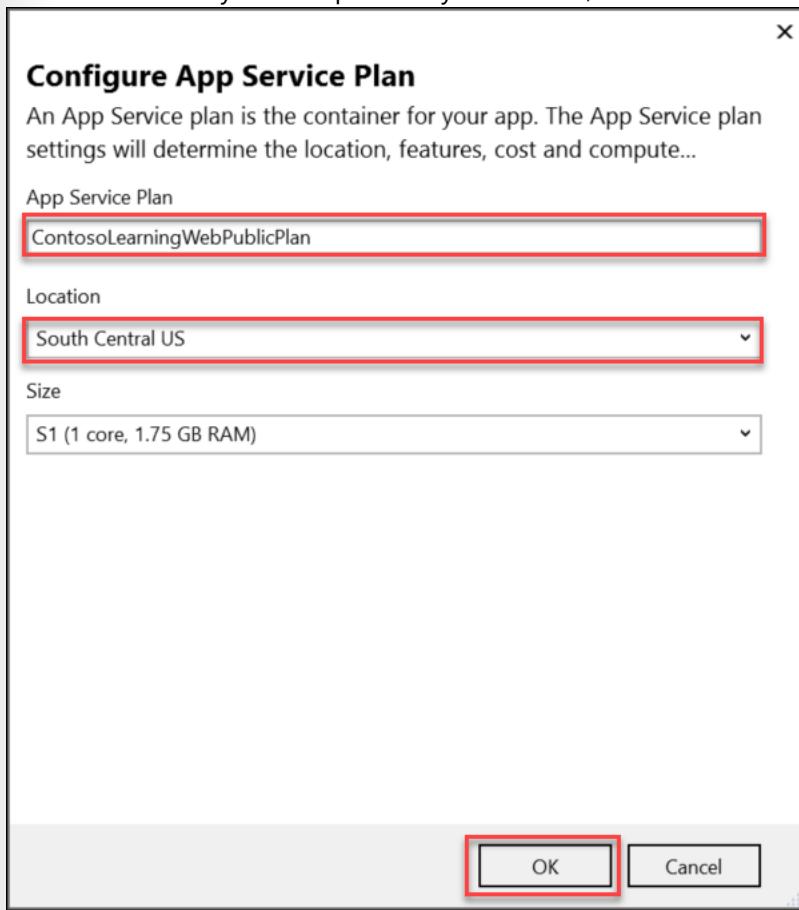
2. Click on **Microsoft Azure App Service**, then select the **Create New** radio button, then click **Publish**



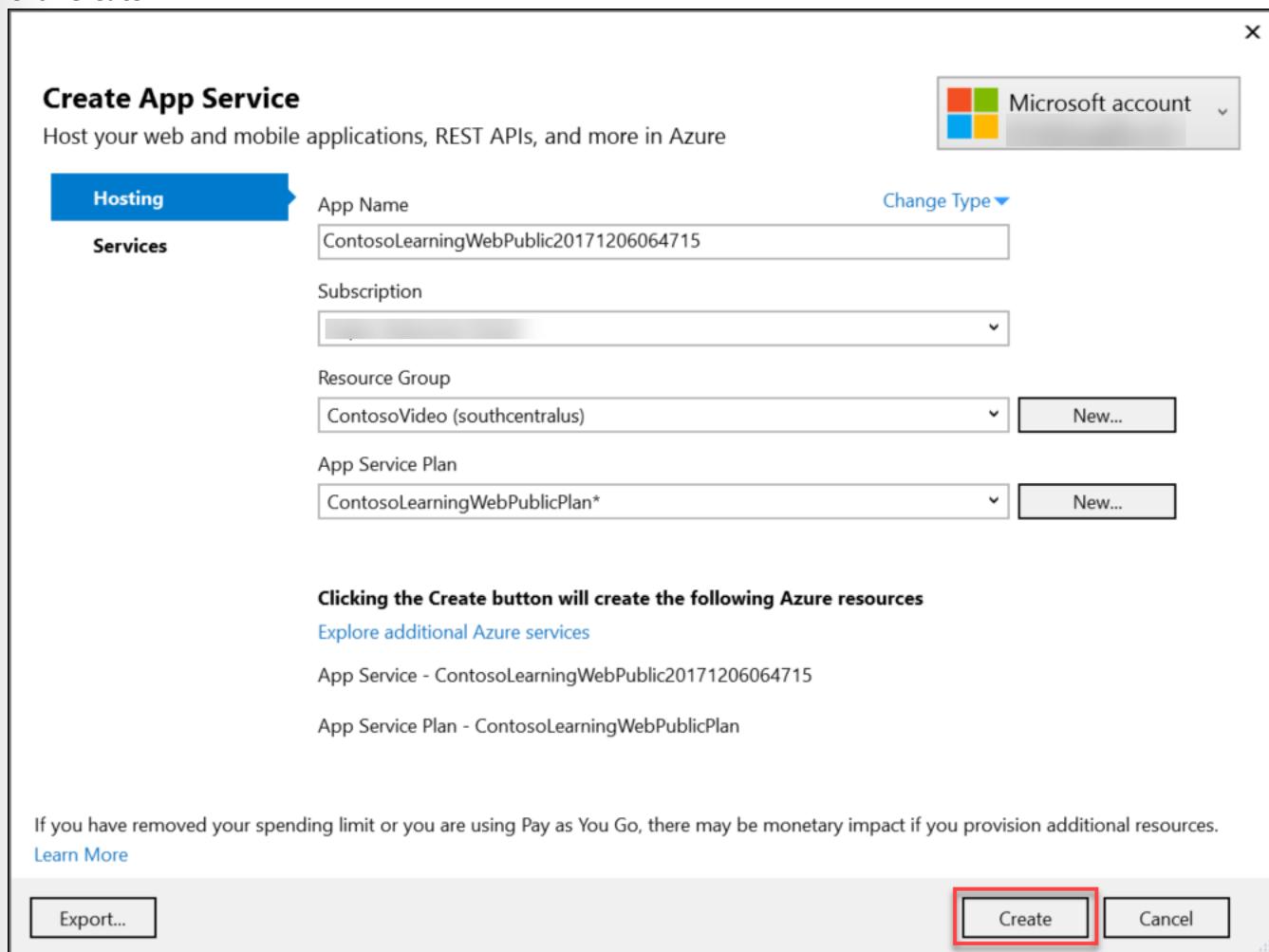
3. On the **Create App Service** dialog, select the **ContosoVideo** Resource Group, then click **New** next to **App Service Plan**



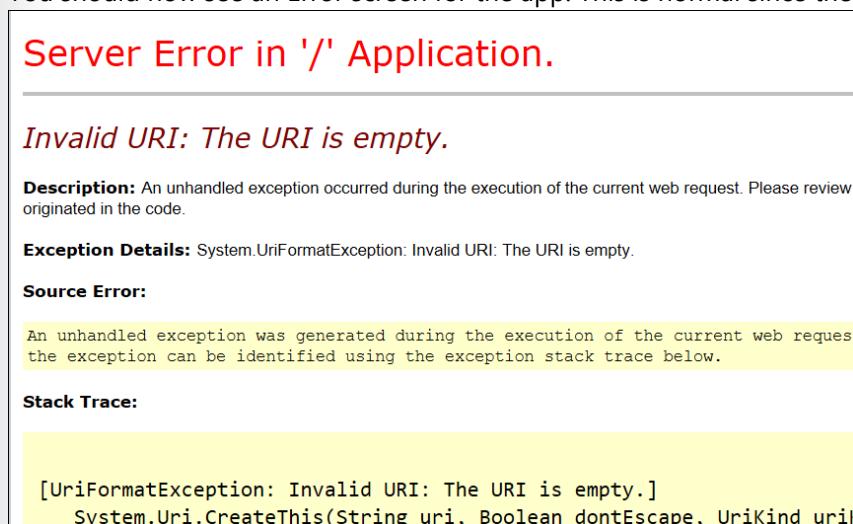
- On the **Configure App Service Plan** dialog, enter a valid name into the **App Service Plan** name field, then select the **Location** that you used previously for this lab, then click **OK**



5. Click **Create**



6. Once the application has finished deploying to the Azure Web App, a new browser window will be opened and navigated to the web app running in Azure
7. You should now see an Error screen for the app. This is normal since the AppSettings haven't been configured yet.



Step 7: Configure Application Settings

1. Open the **Azure Portal**, click on **Resource groups** in the left-side menu, then click the **ContosoVideo** Resource Group, then click on the **Azure Web App** that was just created for the Public website

The screenshot shows the Microsoft Azure portal interface. On the left, the navigation menu is visible with items like 'Create a resource', 'All services', 'Dashboard', 'Resource groups' (which is highlighted with a red box and has a red '1' over it), 'All resources', 'App Services', 'Virtual machines', 'Virtual machine scale sets', 'SQL databases', 'Security Center', and 'Subscriptions'. In the center, the 'Resource groups' blade is open, showing a list of resource groups: 'ContosoVideo' (highlighted with a red box and has a red '2' over it), 'ac522a1fb7735437ea9d614e', 'azureblob', 'ContosoLearningWebAdmin20180216090352', 'ContosoLearningWebAdminPlan' (highlighted with a red box and has a red '3' over it), 'ContosoLearningWebPublic20180217123854' (highlighted with a red box), and 'ContosoLearningWebPublicPlan'. On the right, the 'ContosoVideo' resource group details page is shown, with sections for Overview, Activity log, Access control (IAM), Tags, SETTINGS, Quickstart, Resource costs, Deployments, Policies, Properties, and Locks.

2. On the **App Service** blade, click on **Application settings**

The screenshot shows the 'App Service' blade for the 'ContosoLearningWebPublic20180217123854' web app. The left sidebar has three options: 'Continuous Delivery (Preview)', 'SETTINGS', and 'Application settings' (which is highlighted with a red box). Below the sidebar, there are sections for 'WEBSITE_NODE_DEFAULT_VERSION' (set to '6.9.1') and a button '+ Add new setting'.

3. On the **Application settings** pane, scroll down to the **App settings** section

The screenshot shows the 'Application settings' pane. At the top, it says 'Application settings'. Below that, there is a table with one row: 'WEBSITE_NODE_DEFAULT_VERSION' and '6.9.1'. At the bottom of the pane, there is a link '+ Add new setting'.

4. Add a new App Setting with the Key of **CosmosDB_Endpoint** with the Value set to the **Cosmos DB Account URI** that was copied

Application settings

| | |
|-----------------------------------|---|
| WEBSITE_NODE_DEFAULT_VERSION | 6.9.1 |
| CosmosDB_Endpoint | https://contosovideodb1.documents.azure.com:443/ |
| + Add new setting | |

5. Add a new App Setting with the Key of **CosmosDB_AuthKey** with the Value set to the **Cosmos DB Account Primary Key** that was copied

Application settings

| | |
|-----------------------------------|---|
| WEBSITE_NODE_DEFAULT_VERSION | 6.9.1 |
| CosmosDB_Endpoint | https://contosovideodb1.documents.azure.com:443/ |
| CosmosDB_AuthKey | 'RHfxlnoec4YjB7u7AD9Ad6SjqMRYYr3gpsYpGUJrrwz2KrlxFaAqWiMOAIYTxrJ0g== |
| + Add new setting | |

6. Add a new App Setting with the Key of **CosmosDB_Database** with the Value set to **learning**

Application settings

| | |
|-----------------------------------|---|
| WEBSITE_NODE_DEFAULT_VERSION | 6.9.1 |
| CosmosDB_Endpoint | https://contosovideodb1.doc |
| CosmosDB_AuthKey | qHDBJfE4VdzdQP6MqP |
| CosmosDB_Database | learning |
| + Add new setting | |

7. Add a new App Setting with the Key of **CosmosDB_Collection** with the Value set to **videos**

Application settings

| | |
|-----------------------------------|---|
| WEBSITE_NODE_DEFAULT_VERSION | 6.9.1 |
| CosmosDB_Endpoint | https://contosovideodb1.doc |
| CosmosDB_AuthKey | qHDBJfE4VdzdQP6MqP |
| CosmosDB_Database | learning |
| CosmosDB_Collection | videos |
| + Add new setting | |

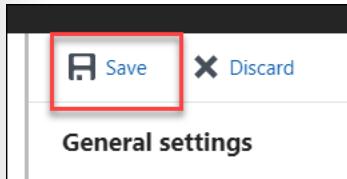
8. Add a new App Setting with the Key of **VideoIndexerAPI_Key** with the Value set to the **API Key** for the **Video Indexer API** subscription

Application settings

| | |
|------------------------------|---|
| WEBSITE_NODE_DEFAULT_VERSION | 6.9.1 |
| CosmosDB_Endpoint | https://contosovideodb1.documents.azure.com |
| CosmosDB_AuthKey | qHDBJfE4VdzdQP6MqPV7RHfxlnoec4YjB7u7... |
| CosmosDB_Database | learning |
| CosmosDB_Collection | videos |
| VideoIndexerAPI_Key | bafa7860d1b44a239ee2baad6f0f0be9 |

+ Add new setting

9. Click **Save**



Exercise 6: Test the application

Duration: 15 minutes

In this exercise, you will test out the Admin and Public web applications.

Step 1: Upload Video to Admin website

1. Open a browser window to the **Admin Website** running in Azure Web Apps. You can use the window that was open previously if it's still open

The screenshot shows the 'Contoso Learning Admin' website. At the top, there is a navigation bar with links for 'Contoso Learning Admin', 'Home', and 'Add Video'. The main content area features a large title 'Contoso Learning Admin' and a section titled 'Video Catalog:' with a table. The table has columns for 'Title', 'Description', 'Created (UTC)', and 'State'. A message in the table states: 'No courses have been created yet. Please click [Add Video](#) to begin.' At the bottom of the page, there is a copyright notice: '© 2017 - Contoso Learning'.

2. Click the **Add Video** link to begin uploading and adding a new Video to the catalog in the application

The screenshot shows the same 'Contoso Learning Admin' website as before, but with a red box highlighting the 'Add Video' button in the top navigation bar. This indicates that the user should click this button to proceed with the task.

3. Enter some text into the form for the **Title** and **Description** fields, then click to choose a **Video** file to upload. Choose the "Introduction-to-the-Azure-Portal_mid.mp4" video file downloaded with the Student Files for this Lab, then click **Upload Video**.

The screenshot shows a web-based application interface titled 'Add Course'. At the top, there is a navigation bar with 'Contoso Learning Admin', 'Home', and 'Add Video' links. Below the title, there are three input fields: 'Title' containing 'Intro to Azure Portal', 'Description' containing 'This video shows an intro to the Azure Portal.', and 'Video' which includes a 'Choose File' button and a preview area showing 'Introductio...tal_mid.mp4'. At the bottom right of the form are 'Upload Video' and 'Cancel' buttons, where 'Upload Video' is highlighted with a red box.

NOTE: Uploading the video file may take a few minutes depending on your Internet connection.

- Once the video file has been uploaded, the homepage of the Admin app will load displaying the Video Processing State and Progress

The screenshot shows the 'Contoso Learning Admin' application. At the top, there is a dark header bar with the text 'Contoso Learning Admin', 'Home', and 'Add Video'. Below the header, the main content area features a large, bold title 'Contoso Learning Admin' centered on the page. Underneath the title, there is a section titled 'Video Catalog:' followed by a table. The table has four columns: 'Title', 'Description', 'Created (UTC)', and 'State'. It contains one row with the following data:
Title: Intro to Azure Portal
Description: This video shows an intro to the Azure Portal.
Created (UTC): 12/7/2017 1:33:25 AM
State: Processing (0%)
At the bottom of the content area, there is a copyright notice: '© 2017 - Contoso Learning'.

| Title | Description | Created (UTC) | State |
|-----------------------|--|----------------------|-----------------|
| Intro to Azure Portal | This video shows an intro to the Azure Portal. | 12/7/2017 1:33:25 AM | Processing (0%) |

Step 2: View Video and Insights in Public website

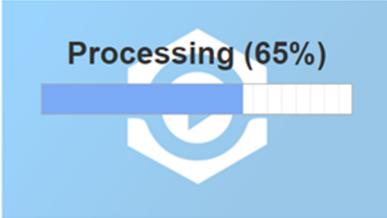
- Open a browser window to the **Public Website** running in Azure Web Apps. If using the window previously opened, just refresh the page to reload it.

2. While the Videos are Processing within the **Video Indexer** service, the Public Website will display the Processing State and Progress

Contoso Learning

At Contoso Learning you learn from the best!

Video Catalog:



Intro to Azure Portal

This video shows an intro the the Azure Portal.

3. Once the videos have finished processing, click on the Thumbnail image or the Title of the Video to view that Video

Contoso Learning

At Contoso Learning you learn from the best!

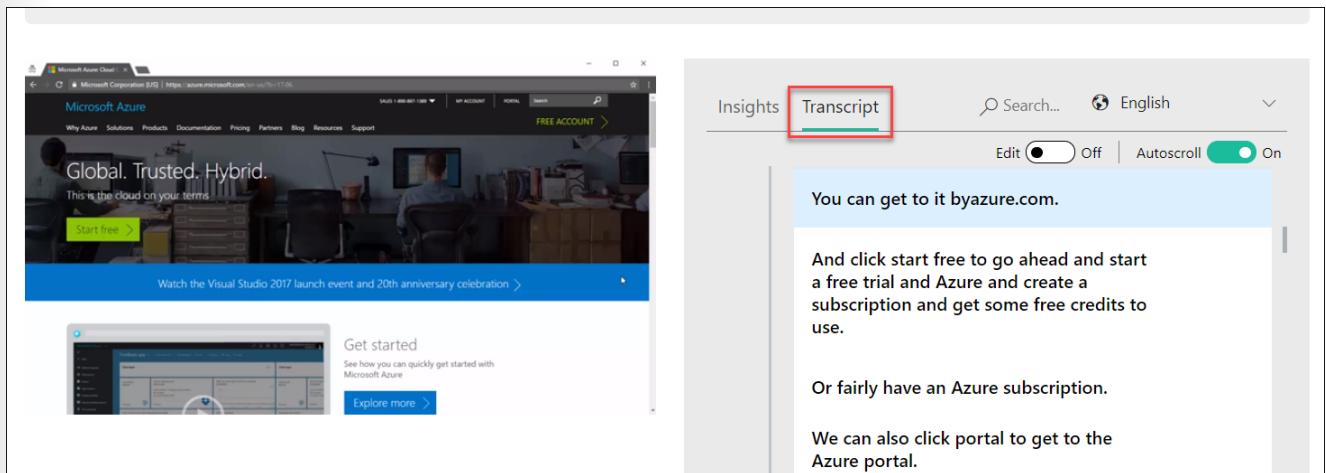
Video Catalog:



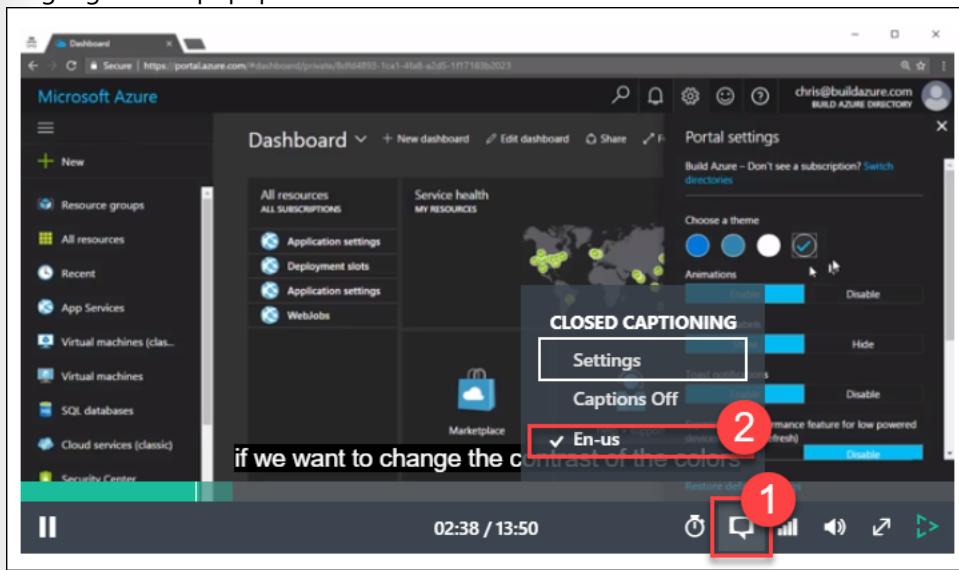
Intro to Azure Portal

This video shows an intro the the Azure Portal.

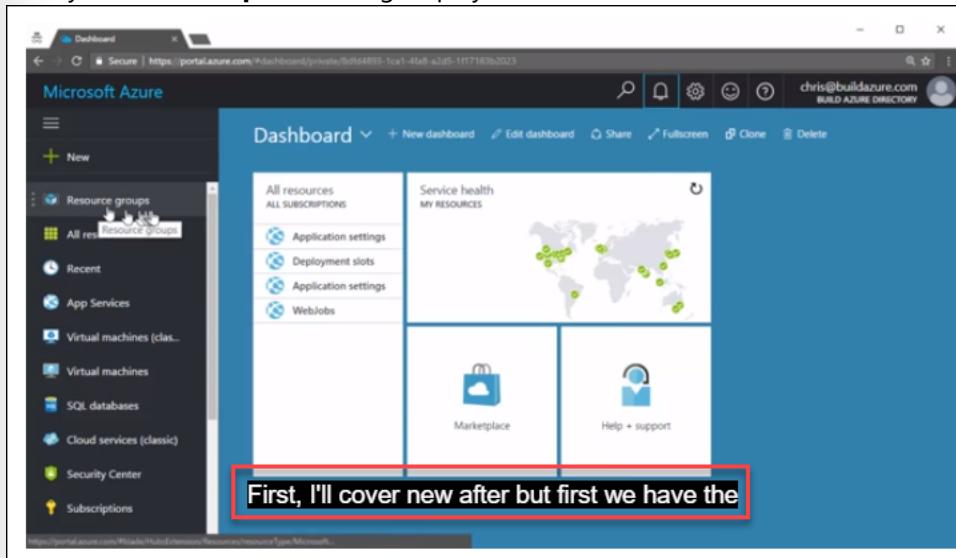
- On the Video Player page, click the **Transcript** tab within the Video Insights. This will show you the Transcription of the Audio from the Video.



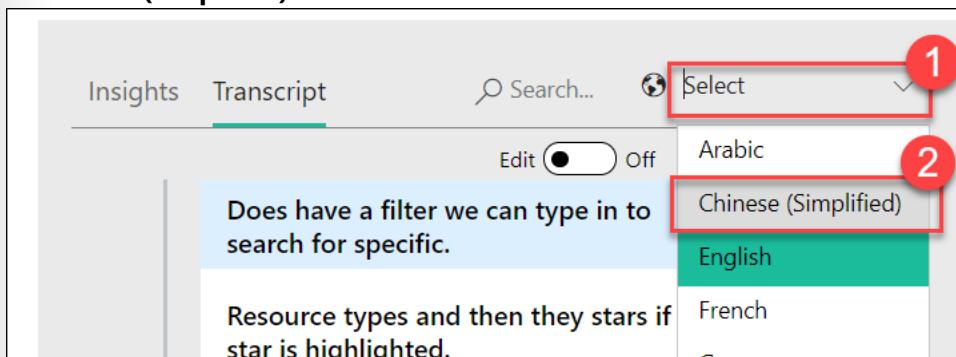
- Notice as the video plays through, the Transcript highlights and automatically scrolls to show you the text you're hearing in the video
- Hover over the **Video Player**, then hover over the **Closed Captions** icon, then click on the **En-us** (English) language in the popup menu



7. Now you will see **Captions** being displayed over the video



8. Click on the **Language** dropdown in the top-right of the **Video Insights** pane, then click to change the language to **Chinese (Simplified)**



9. The **Closed Captions** and the **Transcript** will now be automatically translated into **Chinese (Simplified)**

A screenshot showing two windows side-by-side. On the left is the Microsoft Azure Compute blade, showing a list of services like Marketplace, Compute, Networking, Storage, etc. A red box highlights the text '它, 所以 Windows 服务器2012是2数据中心。'. On the right is the Video Insights pane. It has tabs for 'Insights' and 'Transcript', a search bar, and a language dropdown set to 'Chinese (Simplified)'. A red box highlights the transcript area, which contains the same text as the Compute blade, indicating automatic translation. A red box also highlights the 'Autoscroll' button at the bottom of the pane.

After the hands-on lab

Duration: 10 minutes

Task 1: Delete resources

1. Now that the hands-on lab is complete, go ahead and delete all the Resource Groups that were created for this lab. You will no longer need those resources and it will be beneficial to clean up your Azure Subscription.

You should follow all steps provided *after* attending the Hands-on lab.

For Trainer use or to accompany paid student lab manuals in class only