Azure Fast Start for Mobile Application Development

Module 7: Azure Logic Apps

Student Lab Manual

Instructor Edition (Book Title Hidden Style)

Version 1.1

**Conditions and Terms of Use**

Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the 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.

© 2015 Microsoft Corporation. All rights reserved.

**Copyright and Trademarks**

© 2015 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in 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.

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.

For more information, see Use of Microsoft Copyrighted Content at  
<http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

Azure, Microsoft, OneDrive, and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

[Lab 1: Design a Logic App to Route Image Files 1](#_Toc432793708)

[Exercise 1: Prepare the Logic App 2](#_Toc432793709)

[Task 1: Create an Azure Storage Blob Connector 2](#_Toc432793710)

[Task 2: Create an Empty Logic App 3](#_Toc432793711)

[Task 3: Add a Trigger (Recurrence) 5](#_Toc432793712)

[Exercise 2: Design the Logic App 7](#_Toc432793713)

[Task 1: Add a OneDrive Connector to Enumerate Image Files 7](#_Toc432793714)

[Task 2: Add a OneDrive Connector to Retrieve the Image Files 8](#_Toc432793715)

[Task 3: Add an Azure Storage Blob Connector to Upload the Dropped Images Files 10](#_Toc432793716)

[Task 4: Add a OneDrive Connector to Delete the Dropped Image Files 11](#_Toc432793717)

[Task 5: Test your Logic App 13](#_Toc432793718)

[Lab 2: Provisioning a Database from the CSV File of a Dropbox Share 15](#_Toc432793719)

[Exercise 1: Prepare the Logic App 16](#_Toc432793720)

[Task 1: Create the empty Logic App 16](#_Toc432793721)

[Task 2: Add a Trigger (Recurrence) 16](#_Toc432793722)

[Exercise 2: Design the Logic App 17](#_Toc432793723)

[Task 1: Add a Dropbox Connector to Get a File 17](#_Toc432793724)

[Task 2: Add a Custom API App to Provision the Database 18](#_Toc432793725)

[Task 3: Add a Dropbox Connector to Delete the CSV File 19](#_Toc432793726)

[Exercise 2: Test your Logic App and analyze the result 21](#_Toc432793727)

[Task 1: Test your Logic App 21](#_Toc432793728)

[Task 2: Check the Database 21](#_Toc432793729)

[Lab 3: Get All Tweets for a Specific Product Brand 23](#_Toc432793730)

[Exercise 1: Prepare client-side app to share data on Twitter 24](#_Toc432793731)

[Task 1: Add the Sharing Contract to the Client Application 24](#_Toc432793732)

[Task 2: Install the Twitter App from the Windows Store 26](#_Toc432793733)

[Task 3: Use the Client App and Tweet About Some Products 26](#_Toc432793734)

[Exercise 2: Prepare the Logic App 28](#_Toc432793735)

[Task 1: Create a Twitter App to be Used by the Twitter Connector 28](#_Toc432793736)

[Task 2: Create a Twitter Connector 28](#_Toc432793737)

[Task 3: Create an empty Logic App 29](#_Toc432793738)

[Task 4: Add a Trigger (Recurrence) 29](#_Toc432793739)

[Exercise 3: Design the Logic App 30](#_Toc432793740)

[Task 1: Add a Twitter Connector 30](#_Toc432793741)

[Task 2: Add an Azure Storage Blob Connector to Upload the Content of Tweets Found 30](#_Toc432793742)

[Task 3: Use the guid Method to Upload Unique Files 31](#_Toc432793743)

[Exercise 4: Test your Logic App and analyze the result 33](#_Toc432793744)

[Task 1: Test Your Logic App and Analyze the Result 33](#_Toc432793745)

# Lab 1: Design a Logic App to Route Image Files

#### Introduction

This lab demonstrates the power and simplicity of Logic App and application programming interface (API) apps to create quickly a workflow routing some image files from one cloud location (OneDrive) to another cloud location (Microsoft Azure Storage Blob). Logic Apps are still in the Preview mode and will evolve continuously until their global availability.

#### Objectives

After completing this lab, you will know to:

* Create a Logic App.
* Add triggers.
* Create connectors.
* Test a Logic App and analyze the result.

#### Prerequisites

* A OneDrive account must have been created (<https://onedrive.live.com>).
* An Azure blob container must have been created and accessible by the Logic Apps.
* A *SampleImages* folder must have been created on the root of your OneDrive folder.

#### Estimated time to complete this lab

40 minutes

## Exercise 1: Prepare the Logic App

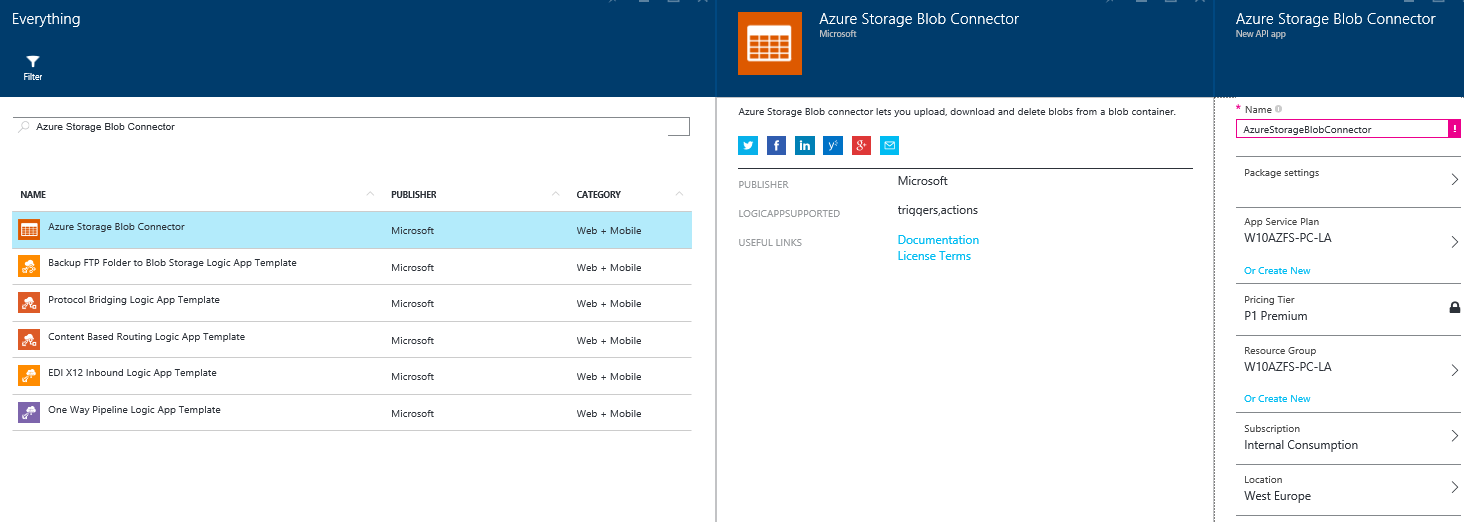
#### Objectives

In this exercise, you will:

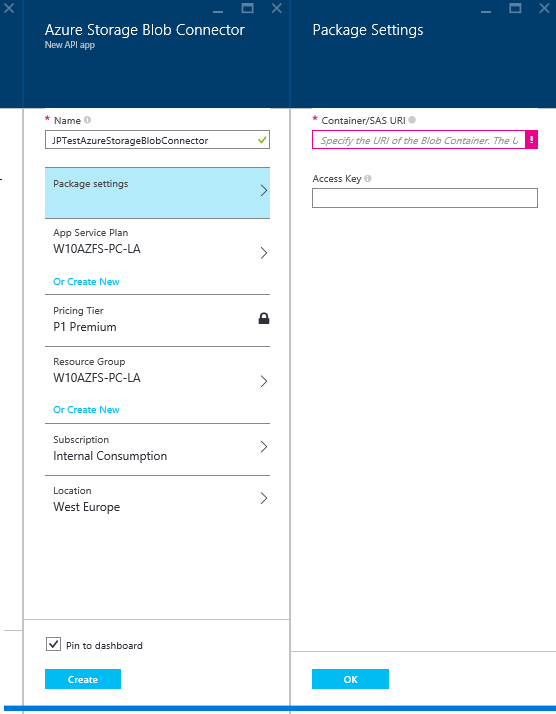
* Create an Azure Storage Blob Connector.
* Create an empty Logic App.
* Add a trigger in the Logic App.

### Task 1: Create an Azure Storage Blob Connector

1. On the home page of the Azure Preview portal, click the **New** button.
2. Select Marketplace.
3. Search for the **Storage Blob**words.
4. In the search result list, select the **Azure Storage Blob Connector** option*.*
5. Click **Create**. The Twitter Connector page will be displayed.



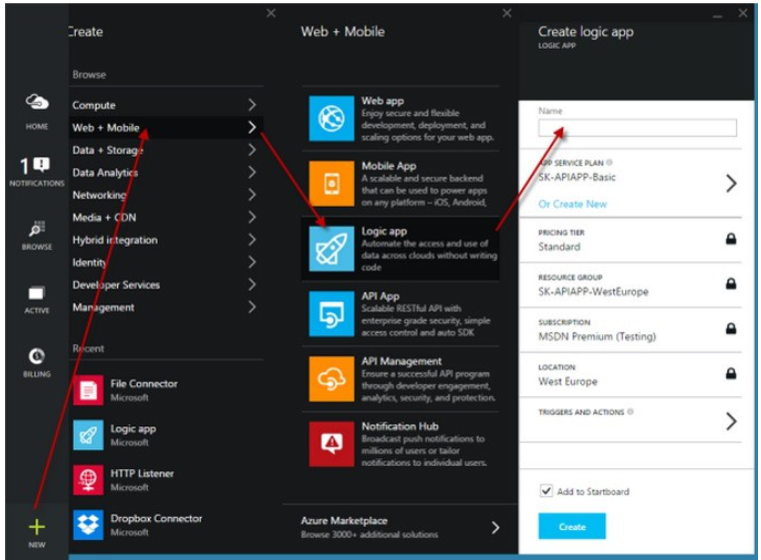
1. Enter a name for the connector.
2. Click **Package settings**and enter the **Container/SAS URI** and the **Access Key** corresponding to the container mentioned in the **prerequisites** of this lab.

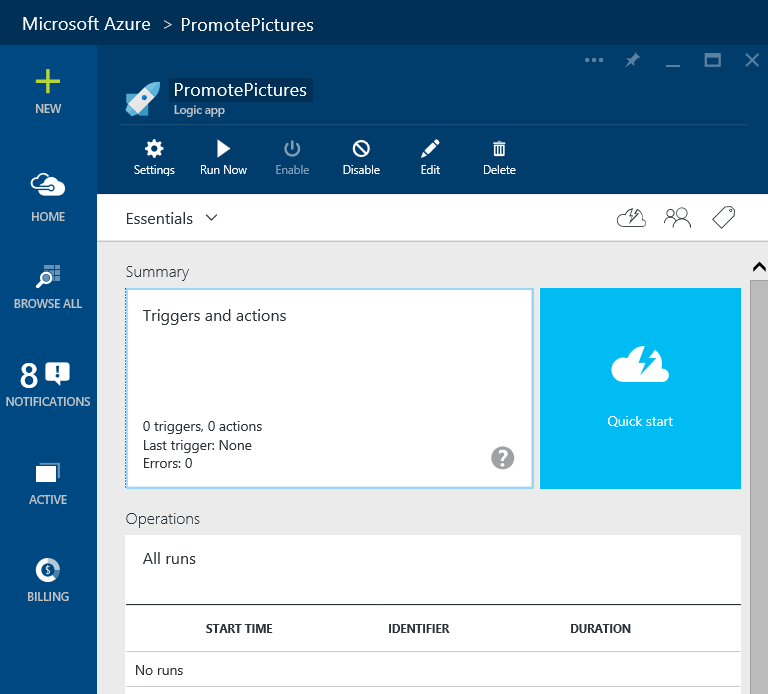


1. Click **OK** and then, to complete the creation of the connector, click **Create**.

### Task 2: Create an Empty Logic App

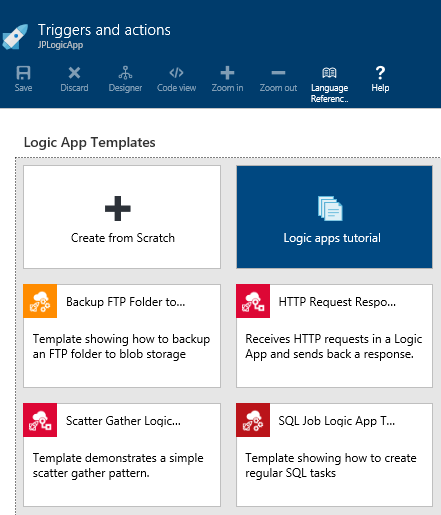
1. On the home page of the Azure Preview portal, click the **New** > **Web + Mobile >** **Logic App** tile*.*
2. In the **Name** box, enter a name for the new Logic App.



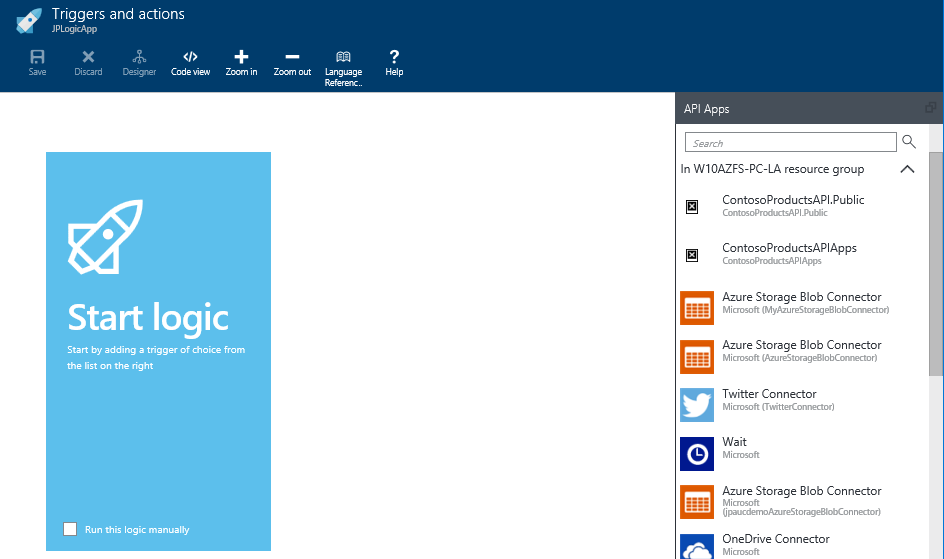
1. Click **Create** and let Azure create the Logic App.
2. When finished, a new tile will be pinned to your home page and the detailed page of your Logic App will be displayed.
3. 

### Task 3: Add a Trigger (Recurrence)

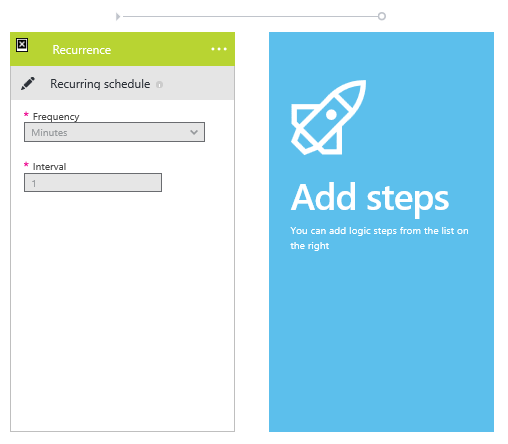
1. On the **Logic App** page, click the **Triggers and actions** tile.
2. In the Logic App Templates list, click Create from Scratch.



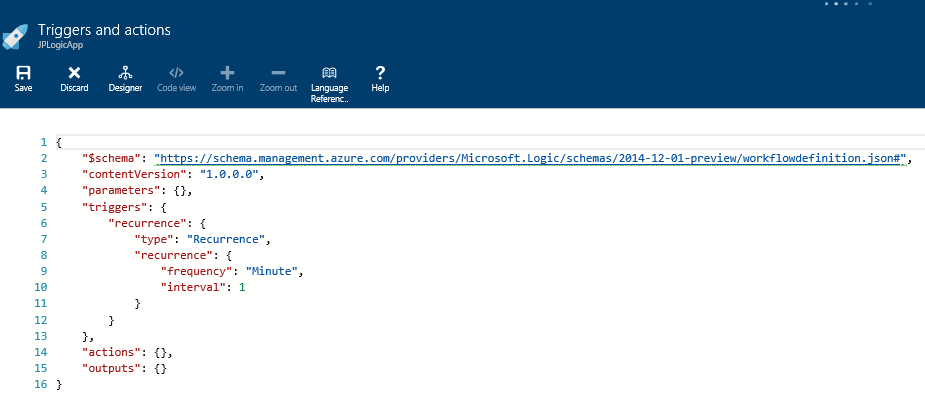
1. A designer view is created to design your Logic App workflow, and the **API Apps** list on the right will display all the API apps that you can reuse in the Logic App.



1. To add a trigger in the designer view, in the **API Apps** list, click **Recurrence**.  
   **Note** You can test a Logic App manually during its design time, but triggers allow you to schedule its startup at run time.
2. In the **Frequency** and **Interval** fields, enter the values that you want for the trigger.
3. Validate your changes on the connector by clicking on the green check mark .



1. To display the code behind the Logic App, at the top bar of the designer view, click the **Code view** link and notice the JSON code that defines the workflow you just created. In addition, examine the part corresponding to the trigger.



## Exercise 2: Design the Logic App

#### Objectives

In this exercise, you will:

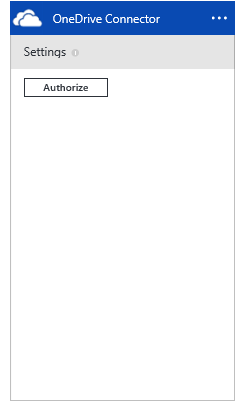
* Design the workflow of the Logic App.
* Add connectors in the Logic App.
* Configure all the connectors to create the workflow of the Logic App.
* Test the Logic App and analyze the result.

#### Scenario

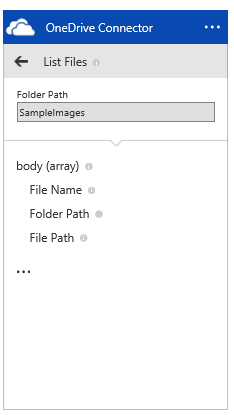
After you created a fresh, new Logic App and provision it with a trigger, you will design its workflow and test it.

### Task 1: Add a OneDrive Connector to Enumerate Image Files

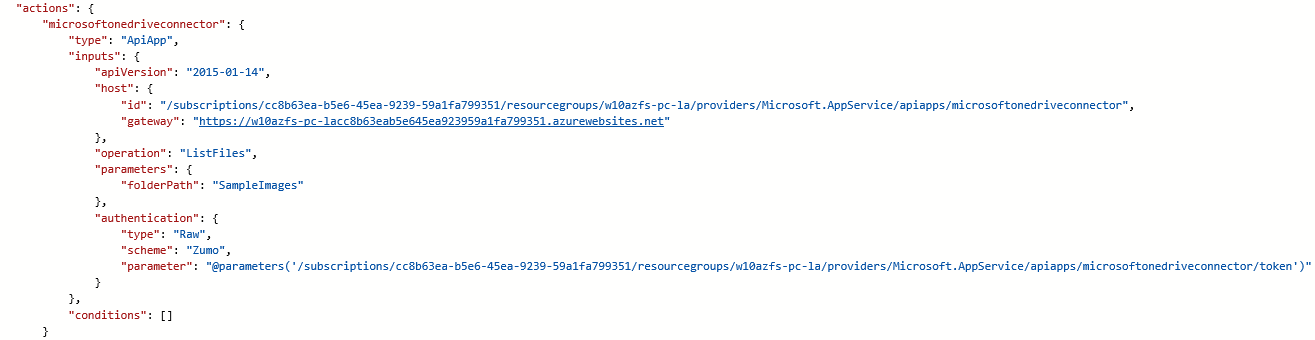
1. To open the Logic App designer view, on the Logic App page, click the **Triggers and actions** tile.
2. To create a connector, in the **API Apps** list, click the **OneDrive Connector** and add it into the designer view.  
   **Note** A connector can be created directly from the Azure Marketplace or within an existing Logic App.
3. To provide your OneDrive credentials, click **Authorize**.



1. Chose the **List Files** action.
2. In the **Folder Path** field, specify the **SampleImages** folder that you had created in the prerequisites section.
3. Validate your changes on the connector by clicking on the green check mark.

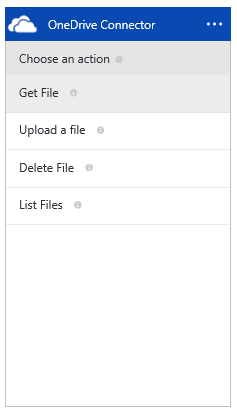


1. To display the JSON code of the Logic App, in the top bar of the designer view, click the **Code view** link and focus on the part corresponding to this connector and action.

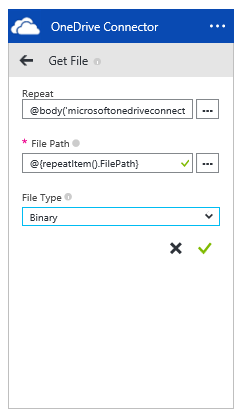


### Task 2: Add a OneDrive Connector to Retrieve the Image Files

1. To create a second OneDrive connector, in the **API Apps** list, click ***OneDrive Connector*** and add it into the designer view.
2. In this OneDrive connector, chose the **Get File** action.



1. Expand the connector menu and select **Repeat over a list**. A **Repeat** field will be displayed in the connector.
2. In the **Repeat** field, click the browse button and select **List Files Body**.
3. In the **File Path** field, click the browse button and select **List Files File Path**.
4. In the **File Type** drop-down list, select **Binary***.*
5. To validate your changes on the connector, click the green check mark.

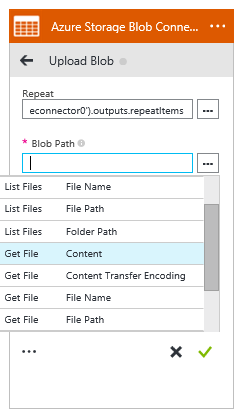


1. To display the JSON code of the Logic App, in the top bar of the designer view, click the **Code view** link and focus on the part corresponding to this connector and action. Notice how the dependency between connectors is managed.

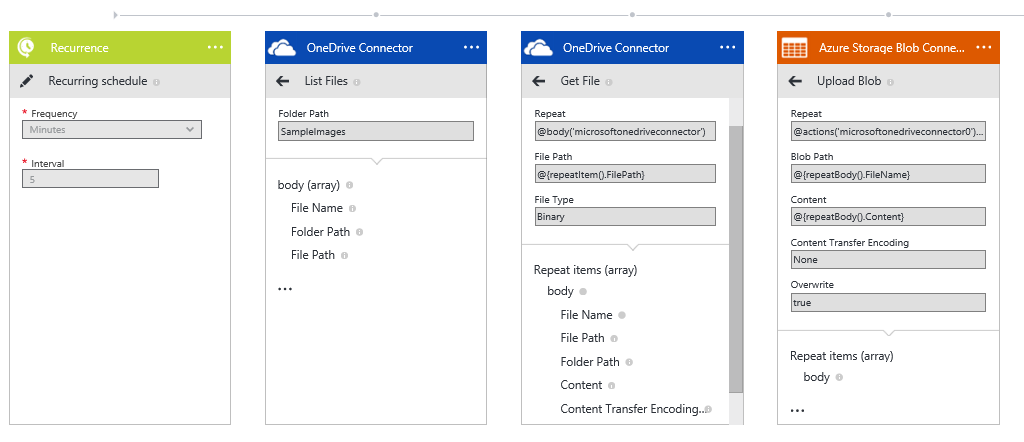


### Task 3: Add an Azure Storage Blob Connector to Upload the Dropped Images Files

1. In the **API Apps** list, click the**Azure Storage Blob Connector** that you created in **Task 1** to add to the Logic App the ability to upload datas to the Azure Storage.
2. In the Azure Storage Blob Connector, click the Upload Blob action.
3. Expand the connector menu and select**Repeat over a list**.
4. In the**Repeat** field, click the browse button and select **Get Files Repeat items**.
5. In the**Blob Path** field, click the browse button and select**Get File File Name**.
6. In the**Content**field, click the browse button and select**Get File Content**.

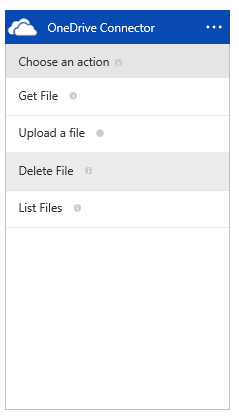


1. Select the **Overwrite** field to true.
2. Validate your changes on the connector by clicking on the green check mark.
3. To display the JSON code of the Logic App, in the top bar of the designer view, click the **Code view** link and focus on the part corresponding to this connector and action.
4.   
   10. This is the final Logic App you that will be displayed:

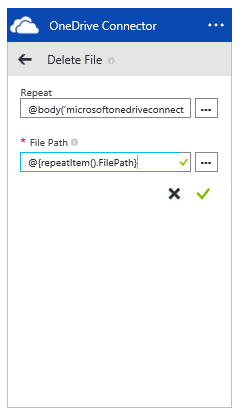


### Task 4: Add a OneDrive Connector to Delete the Dropped Image Files

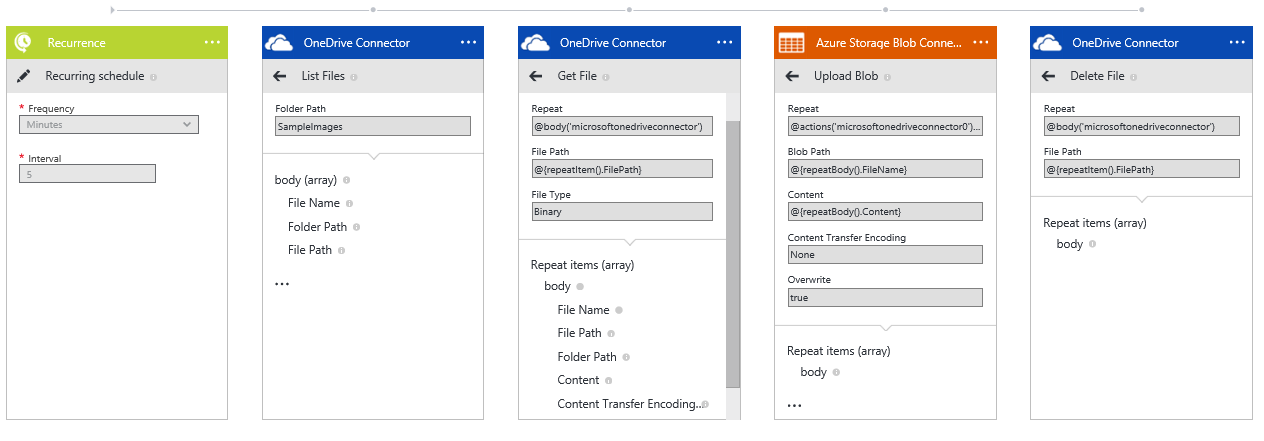
1. To create a third OneDrive connector, in the **API Apps** list, click **OneDrive Connector** and add it to the designer view.
2. In the OneDrive connector, click the **Delete File** action.



1. Expand the connector menu and select **Repeat over a list**.
2. In the **Repeat** field, click the browse button and select **List Files Body**.
3. In the **File Path** field, click the browse button and select **List Files File Path**.
4. Validate your changes on the connector by clicking on the green check mark.



1. This is the final Logic App that will be displayed:

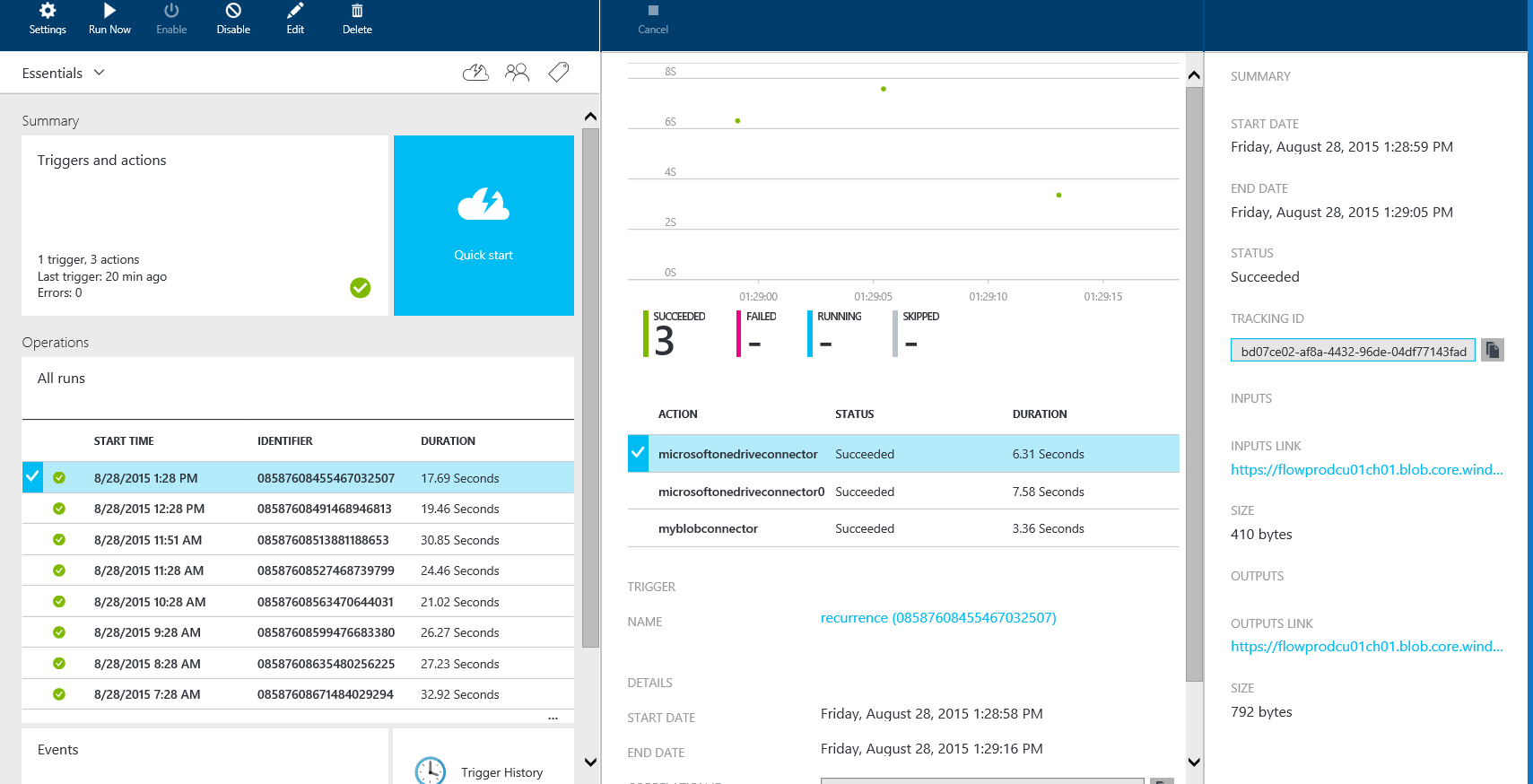


1. To display the JSON code of the Logic App, in the top bar of the designer view, click the **Code view** link and focus on the part corresponding to this connector and action.



### Task 5: Test your Logic App

1. Drop few image files within the OneDrive folder **SampleImages**.
2. To display the Logic App page, in the navigation bar of the designer view, click the Logic App name.
3. On the **Logic App** page, in the top bar, click the **Run Now** link and notice the new entry in the **Operations** tile*.*
4. Select the entry and review the corresponding **Logic App run** page on the right side.
5. Wait for the test results and notice the status of each action on the Run page.
6. Select each action line to display its details and open its input and output links to display the corresponding JSON codes.
7. If the test result is successful, check either with the Server Explorer view of Visual Studio or using the Azure portal that the images have been uploaded on the Azure blob container.



# Lab 2: Provisioning a Database from the CSV File of a Dropbox Share

#### Introduction

In this lab, you will design a Logic App provisioning automatically an Azure SQL Server database from the content of a comma-separated values (CSV) file dropped in a Dropbox share.

#### Objectives

After completing this lab, you will be able to:

* Use a Dropbox connector to get a file and delete it.
* Use a custom API App taking as parameter a file content to be parsed.

#### Prerequisites

* A Dropbox account must have been created (<https://www.dropbox.com>).
* A CVS file named **SampleProducts**must have been be created containing the following line:
* **New Demo Product,Test product,Demo,Gold,20, /Data/ProductImages/demo.png,1,11**
* The custom API App **WebCatalogue,** created in the module Module06-API Apps > Lab 1 > Exercice 2, must have been provisioned in your subscription and be visible in the API Apps list.

#### Estimated time to complete this lab

25 minutes

## Exercise 1: Prepare the Logic App

#### Objectives

In this exercise, you will:

* Create an empty Logic App.
* Configure a trigger.

### Task 1: Create the empty Logic App

1. On your home page of the Azure Preview portal, click the **New** button.
2. Select the **Web + Mobile** category*.*
3. Select the **Logic App** tile*.*
4. Enter a name for the new Logic App.
5. Click **Create** and let Azure create the Logic App.

### Task 2: Add a Trigger (Recurrence)

1. On your Logic App page, click the **Triggers and actions** tile.
2. In the list of Logic App Templates, click the **Create from Scratch** tile*.*
3. To add a trigger in the design view, in the **API Apps** list, click **Recurrence**.
4. Enter in the **Frequency** and **Interval** fields, enter the values you want for the trigger.
5. Validate these changes by clicking the green check mark of the trigger.

## Exercise 2: Design the Logic App

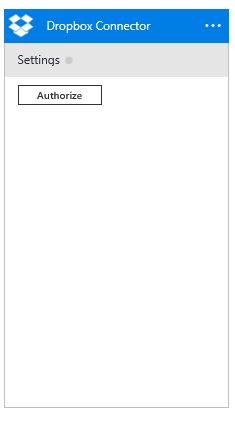
#### Objectives

In this exercise, you will:

* Use a Dropbox connector.
* Use a custom API App.

### Task 1: Add a Dropbox Connector to Get a File

1. In the API Apps list, click **Dropbox Connector** to add into the Logic App a connector getting a .CSV file dropped in the Dropbox.
2. To provide the Dropbox credentials, click **Authorize**.

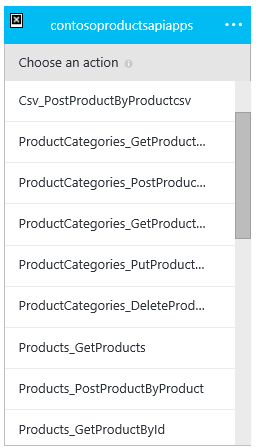


1. Chose the **Get File** action.
2. In the **File Path** field, specify the file name of the CSV file dropped in the Dropbox as **SampleProducts.csv**.
3. Validate your changes on the connector by clicking on the green check mark.
4. To display the JSON code corresponding to the connector and action, in the top bar of the designer view, click the **Code view** link and notice the **operation** and **parameters** nodes.

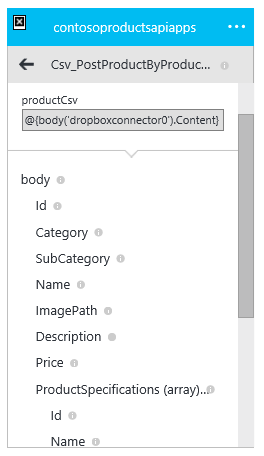


### Task 2: Add a Custom API App to Provision the Database

1. In the **API Apps** list, click the custom API App WebCatalogue to add it in the designer view.



1. In this API App action listing, select the **Csv\_PostProductByProductcsv** action, which accepts as parameter a file content and populate a database.
2. In the **productCsv** field, click the browse button and select **Get File Content**.

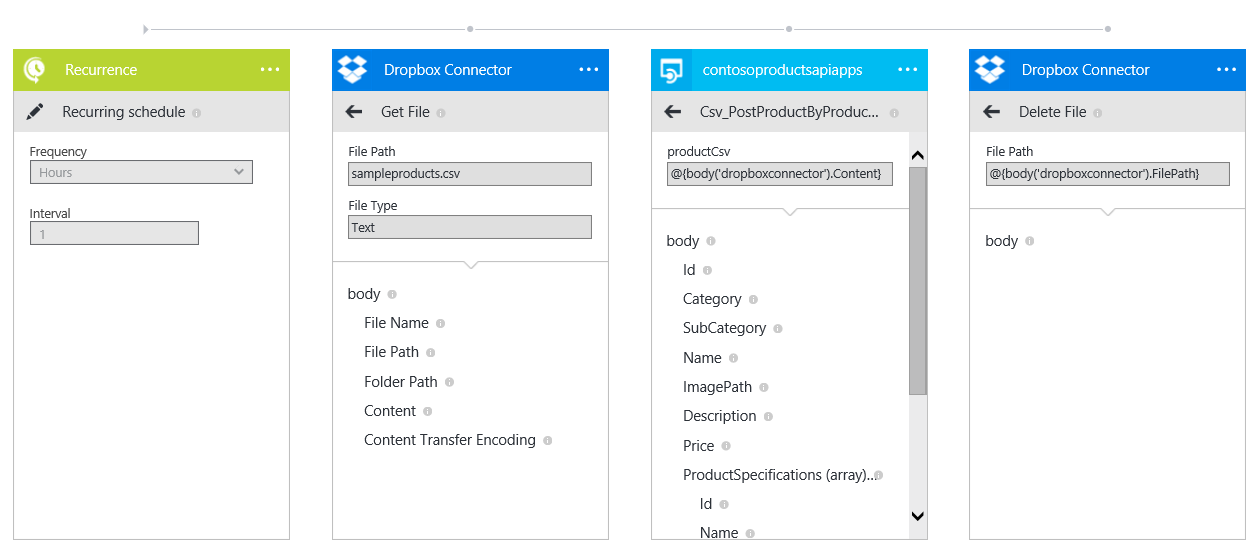


1. Validate your changes on the connector by clicking on the green check mark.
2. To display the JSON code corresponding to the connector and action, in the top bar of the designer view, click the **Code view** link.



### Task 3: Add a Dropbox Connector to Delete the CSV File

1. To remove the CSV file from Dropbox, in the **API Apps** list, click a Dropbox Connector.
2. In the Dropbox connector, click the **Delete File** action.
3. In the **File Path** field, click the browse button and select **Get File File Path**.
4. Validate the changes on the connector by clicking on the green check mark.
5. To display the JSON code corresponding to the connector and action, in the top bar of the designer view, click the **Code view** link.
6. This is the resulting Logic App work flow that you should have:



## Exercise 2: Test your Logic App and analyze the result

#### Objectives

In this exercise, you will:

* Test the Logic App created in Exercise 1.
* Analyze the test result on the Logic App Run page.
* Verify the test result in the Products database.

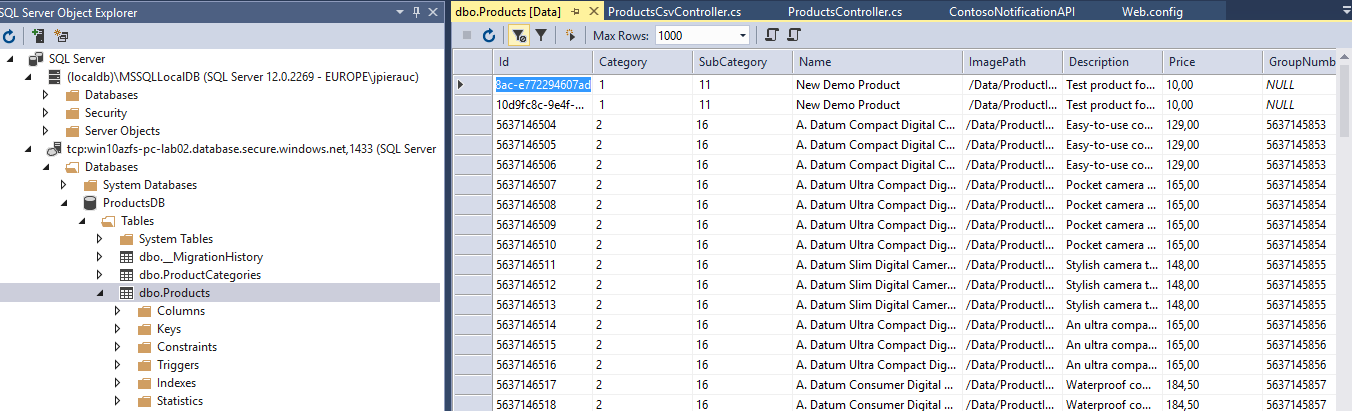
### Task 1: Test your Logic App

1. Drop the file **SampleProducts.csv** within the Dropbox root folder.
2. To display the Logic App page, in the navigation bar of the designer view, click the Logic App name.
3. On the **Logic App** page, in the top bar, click the **Run Now** link*.*
4. Select the corresponding entry in the **Operations** list and review the corresponding **Logic App run** page on the right side.
5. Wait for the test result and notice the status of each action on the Run page.
6. Click each action line to display its details and open its input and output links.

### Task 2: Check the Database

After successful test completion, make sure that the product contained in the CSV file is well inserted in the **Products** table. For this task, use the **SQL Server Object Explorer**.

1. On the home page of Azure portal, select **BROWSE ALL** and select **Search SQL Databases** in the list.
2. Right-click the **ProductsDB** line and select **Open in Visual Studio**.
3. To ensure the client machine can access the database, in the right pane, select **Configure your firewall settings**.
4. On the **Firewall Settings** page, add a rule including the **Client IP Address**.
5. To persist the rule, click the **Save** link and then close the Firewall Settings page.
6. Next, click **Open In Visual Studio***.* Visual Studio will be started and the Connect dialog box will prompt you for the password to connect to the database*.*
7. Connect to the database and explore the data of the **Products** table.



# Lab 3: Get All Tweets for a Specific Product Brand

#### Introduction

In this lab, you will design a Logic App to search for all tweets containing the brand name of a product and for each brand, upload a file on the Microsoft Azure Storage Blob.

#### Objectives

After completing this lab, you will be able to:

* Use a Twitter connector to search for Tweets.
* Use an Azure Storage Blob connector to interface with Twitter connector.
* Use a JSON function to create a unique identifier.

#### Prerequisites

* A Twitter account must have been created (<https://twitter.com>).
* An Azure Storage Blob connector must have been created as shown in **Lab 1**.

#### Estimated time to complete this lab

45 minutes

## Exercise 1: Prepare client-side app to share data on Twitter

#### Objectives

In this exercise, you will:

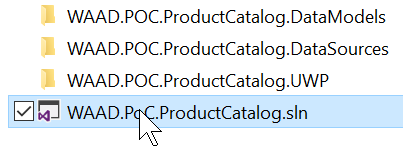
* Use share contract to share data on Twitter.

#### References

* Share data: <https://msdn.microsoft.com/en-us/library/windows/apps/xaml/mt243293.aspx>

### Task 1: Add the Sharing Contract to the Client Application

1. Using File Explorer, go to **\Module00-Apps\ProductCatalog\Source**, copy it into the **\Module07-Logic Apps\Lab3\Exercice0\Start** folder and double-click **WAAD.PoC.ProductCatalog.sln** to open the app in Visual Studio.



1. First, you need to add an extra button in the ViewProductPage in order to allow the user to share a tweet about the displayed product. To do it, open **ViewProductPage.xaml** and locate the **view.CommonPage.BottomAppBar** control.

<view:CommonPage.BottomAppBar>

...

1. Inside the <CommandBar> element, add the new button:

<AppBarButton x:Name="btnAppBarShare" Icon="ReShare" Label="Share" />

</CommandBar>

</view:CommonPage.BottomAppBar>

1. Locate **ctl:HeaderControl.ButtonsContent** now, which is used inside the application header:

<ctl:HeaderControl.ButtonsContent>

1. Inside the <CommandBar> element, add almost the same <AppBarButton> (Only the x:Name changes):

<AppBarButton x:Name="btnHeaderShare" Icon="ReShare" Label="Share" />

</CommandBar>

</ctl:HeaderControl.ButtonsContent>

1. Once the buttons are added, open the **ViewProductPage.xaml.cs** page and add the handler for the buttons’ Click event. You can do this in the class constructor which is the ViewProductPage() function:

public ViewProductPage()

:base(ViewModelType.ViewProduct)

{

InitializeComponent();

// ...

btnAppBarShare.Click += Share\_Click;

btnHeaderShare.Click += Share\_Click;

}

1. Add a DataRequested event handler on DataTransferManager instance of the page to fire whenever a user invokes share:

private void Share\_Click(object sender, RoutedEventArgs e)

{

DataTransferManager dataTransferManager = DataTransferManager.GetForCurrentView();

dataTransferManager.DataRequested += DataTransferManager\_DataRequested;

1. Finally, still in the Share\_Click function, immediately invoke the action of Sharing:

DataTransferManager.ShowShareUI();

}

1. When a DataRequested event occurs, your app receives a DataRequest object. This contains a DataPackage that you can use to provide the content that the user wants to share.

Implement the DataTransferManager\_DataRequested event handler in order to get called and fill the DataRequest object:

* We get the data for the displayed product through the DataContext property.
* We create a string with the Name/Id of the product and reference the application using #ContosoElectronics.
* We use the *SetText()* method from the Data property of the DataRequest in order to inform that we want to share the string previously created.
* Lastly, we add a Title and Description to the DataRequest we are about to share.

private void DataTransferManager\_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)

{

DataRequest request = args.Request;

var vm = DataContext as ViewProductViewModel;

request.Data.SetText(String.Format("Check out {0} (#Id{1}) #ContosoElectronics",

vm.ProductDetails.Name,

vm.ProductDetails.Id));

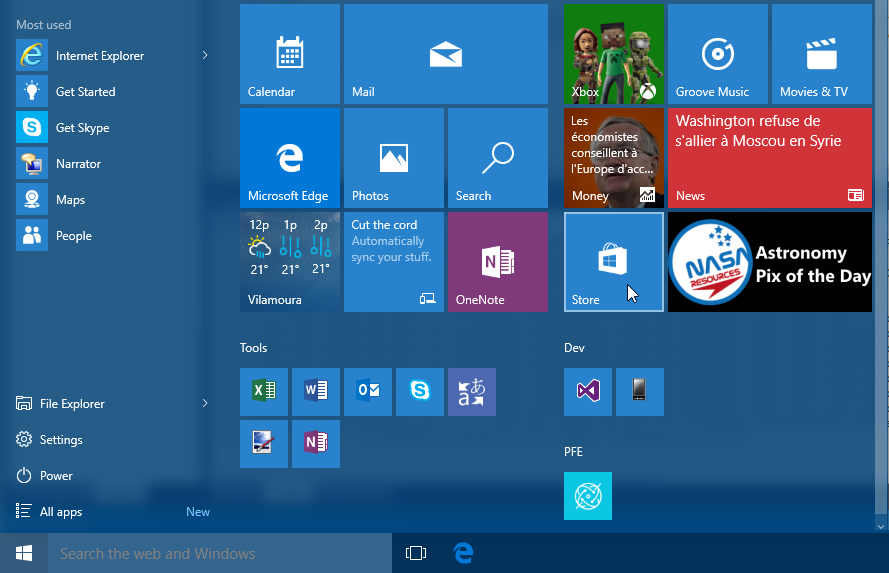
request.Data.Properties.Title = "Share this product to Twitter";

request.Data.Properties.Description = "Please select the Twitter App in order say what you think about this product.";

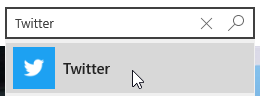
}

### Task 2: Install the Twitter App from the Windows Store

1. Click or tap the **Store** tile and search for **Twitter**.



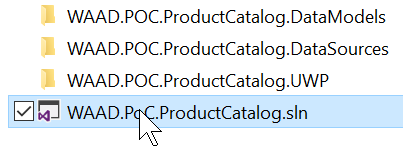
1. The Twitter App will be listed first. Install it and configure your Twitter account in the app. If you do not have a Twitter account, you can create one at <http://twitter.com/>.



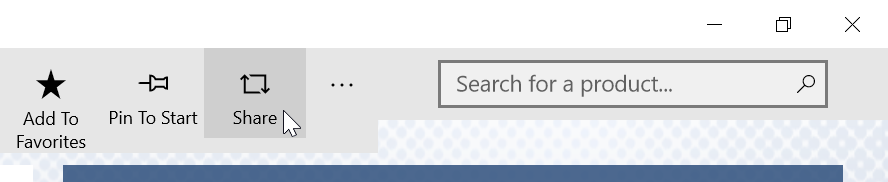
1. The Twitter app is now available and can receive text or/and URL that any Store Apps would like to share. Let's try it.

### Task 3: Use the Client App and Tweet About Some Products

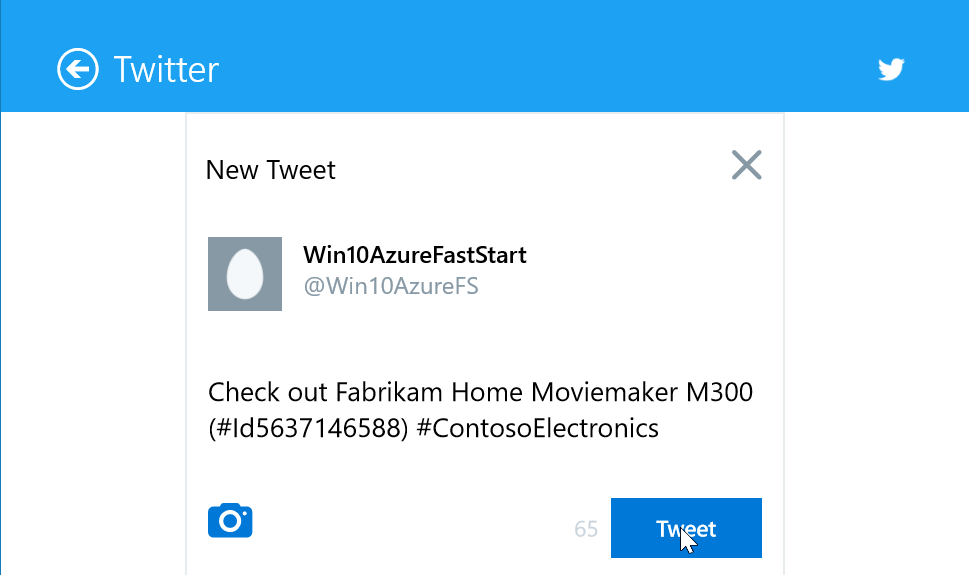
1. Get back to the solution we worked on in the folder **\Module07-Logic Apps\Lab3\Exercice0\Start** and double-click **WAAD.PoC.ProductCatalog.sln** to open the app in Visual Studio (if the solution was not still opened).

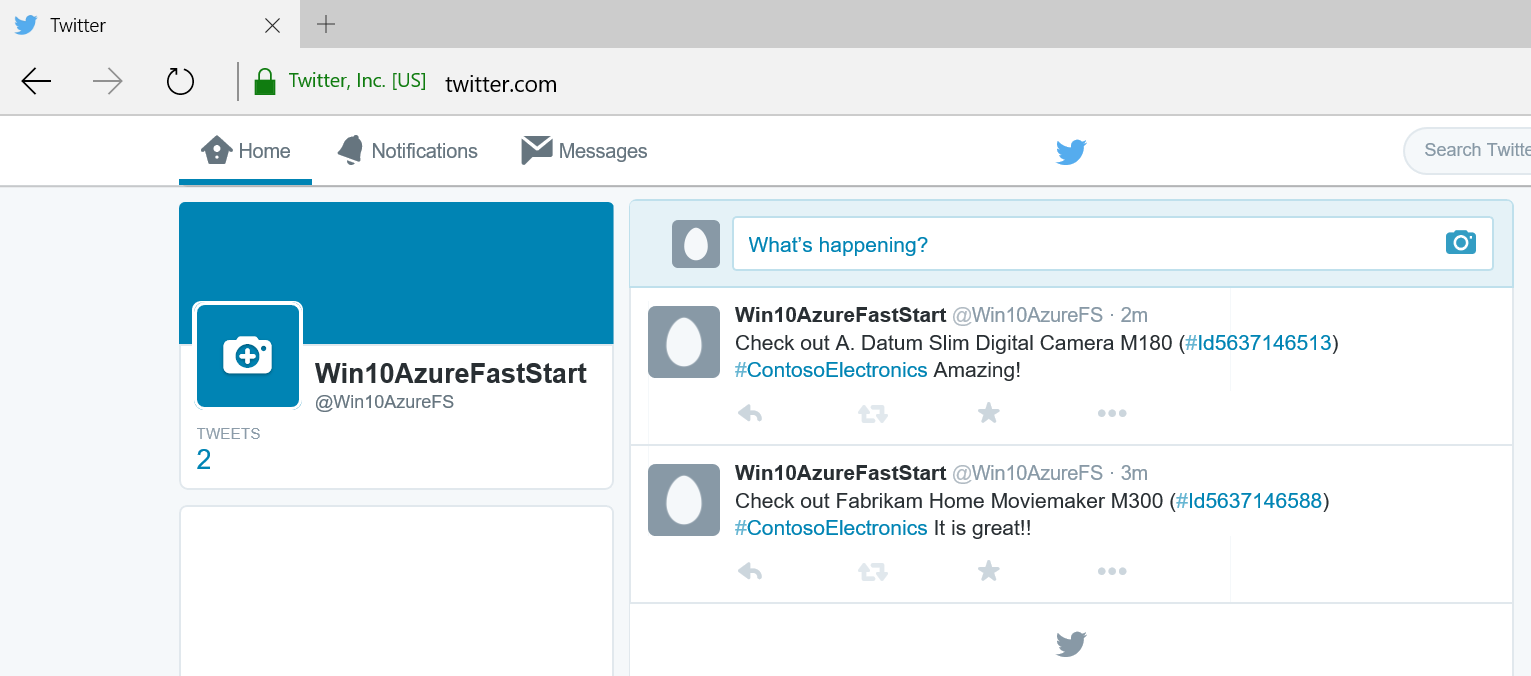


1. Execute the application
2. In the application, navigate to the Product page and on the page, click the **Share** button.



1. The **Share** pane is displayed, which lists all the Store apps that can receive text. Click the **Twitter** app.
2. The tweet is now public and we can work on using Azure.





## Exercise 2: Prepare the Logic App

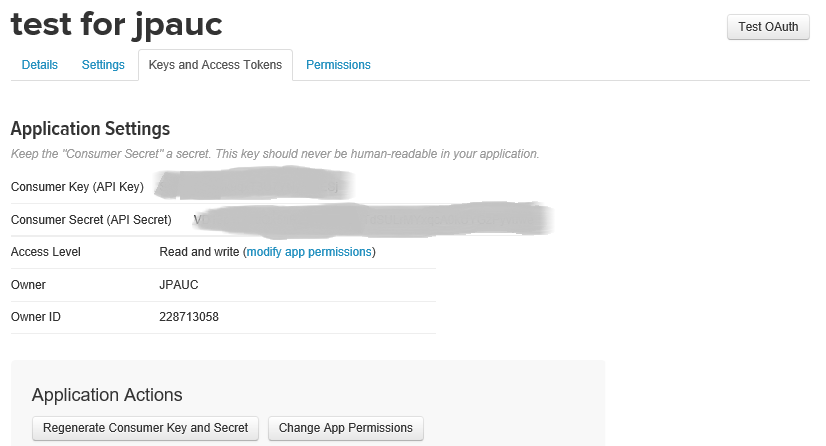
#### Objectives

In this exercise, you will:

* Create a Twitter app to use its keys in the Twitter connector.
* Create a Twitter connector.

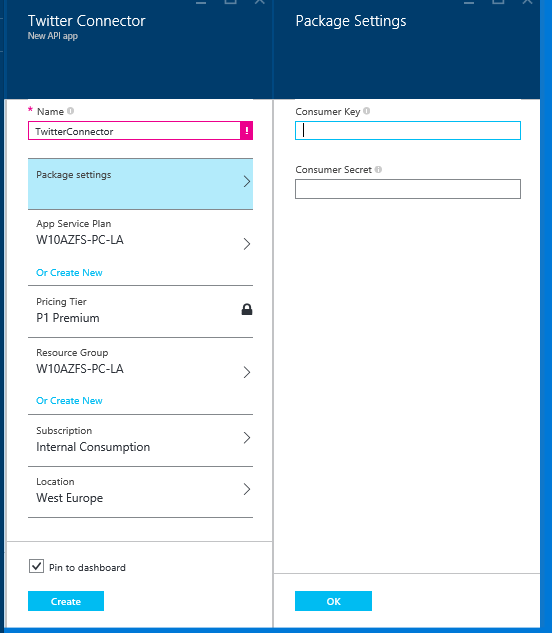
### Task 1: Create a Twitter App to be Used by the Twitter Connector

1. Go to <https://apps.twitter.com>.
2. Create a new app.
3. Fill the **Name**, **Description**, and **Website** fields.
4. In the field *Callback URL*, enter <https://api.twitter.com/1.1/>
5. Click **Create**.
6. On the **Keys and Access Tokens** tab, copy the **Consumer Key** and the**Consumer Secret** to be configured later in the connector.



### Task 2: Create a Twitter Connector

1. On the home page of the Azure Preview portal, click **New**.
2. Click Marketplace.
3. Search for the word **Twitter**.
4. In the search result, select **Twitter Connector***.*
5. Click **Create**. The Twitter Connector page will be displayed.
6. On the **Twitter Connector** page, enter a name for the connector.
7. Click **Package settings** and enter the **Consumer Key** and **Consumer Secret** retrieved previously.



1. Click **OK** and then click **Create** to complete the creation of the connector.

### Task 3: Create an empty Logic App

1. On the home page of the Azure Preview portal, click **New**.
2. Click the **Web + Mobile** category*.*
3. Click the **Logic App** tile*.*
4. Enter a name of the new Logic App.
5. Click **Create** and let Azure create the Logic App.

### Task 4: Add a Trigger (Recurrence)

1. On the **Logic App** page, click the **Triggers and actions** tile.
2. In the list of Logic App Templates, click the tile *Create from Scratch.*
3. To add a trigger in the designer view, in the **API Apps** list, click **Recurrence**.
4. In the **Frequency** and **Interval** fields, enter the values you want for the trigger.
5. Validate the changes by clicking the green check mark.

## Exercise 3: Design the Logic App

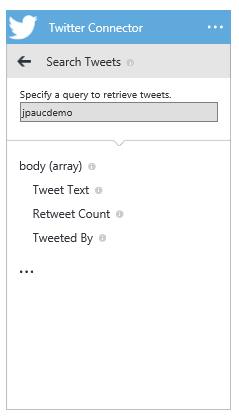
#### Objectives

In this exercise, you will:

* Use a Twitter connector.
* Use an Azure Storage Blob connector.

## Task 1: Add a Twitter Connector

1. In the **API Apps** list of the designer view, select the Twitter connector that you created in **Task 2**.
2. To validate the access, click **Authorize**.
3. Select the **Search Tweet** action*.*
4. Enter a brand name to search for.
5. Validate the changes by clicking the green check mark of the Trigger.



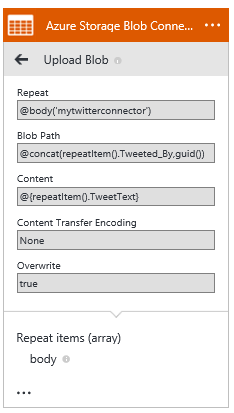
### Task 2: Add an Azure Storage Blob Connector to Upload the Content of Tweets Found

1. In the **API Apps** list, click the Azure Storage Blob Connector created in **Lab 1**.
2. In the Azure Storage Connector, click the **Upload Blob** action.
3. Expand the connector menu and select**Repeat over a list**.
4. In the**Repeat** field, click the browse button and select **Search Tweets Body**.
5. In the**Blob Path** field, click the browse button and select**Search Tweets Tweeted By**.
6. In the**Content**field, click the browse button and select**Search Tweets Tweet Text**.

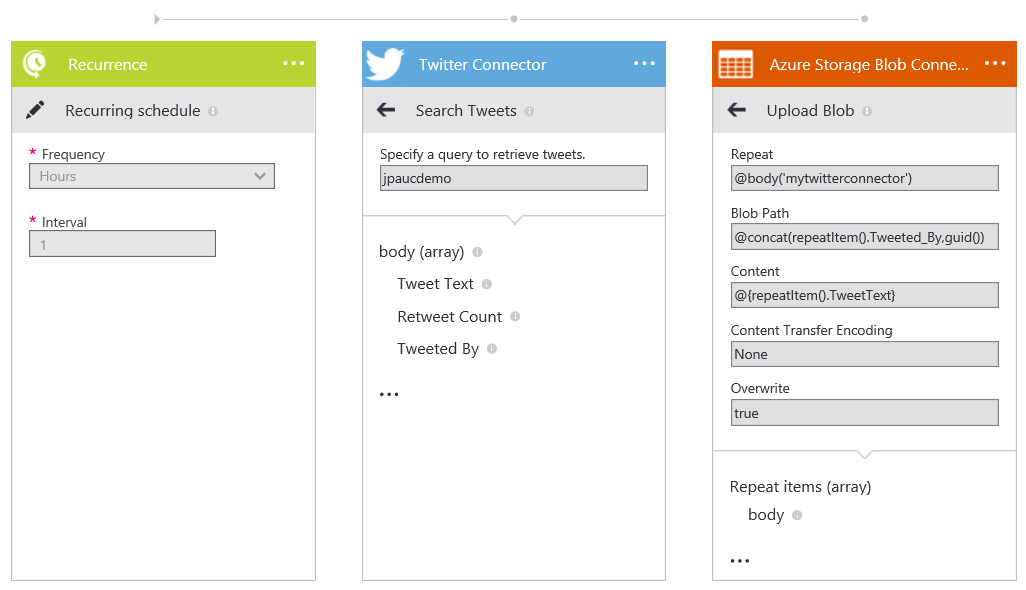
### Task 3: Use the *guid* Method to Upload Unique Files

To be sure to have one file uploaded per Tweet found to the Azure Storage blob, use the *guid()* method to make the file name unique.

1. Replace the value in the **Blob Path** field of the Azure Storage Blob Connector with the following line :  
   *@concat(repeatItem().Tweeted\_By,guid())*



1. Validate all the changes of the connector by clicking on the green check mark.
2. This is the resulting Logic App workflow you should have:



## Exercise 4: Test your Logic App and analyze the result

#### Objectives

In this exercise, you will:

* Test the Logic App.
* Analyze the test result on the Logic App Run page.
* Verify the test result in the Azure Storage Blob.

### Task 1: Test Your Logic App and Analyze the Result

1. Create Tweets containing the product brand name you configured in the Twitter Connector.
2. To display the Logic App page, in the navigation bar of the designer view, click the Logic App name.
3. On the top bar of the **Logic App** page, click the **Run Now** link*.*
4. Select the corresponding entry in the **Operations** list and review the corresponding **Logic App run** page on the right side.
5. Wait for the test result and notice the status of each action on the Run page.
6. Click each action line to display its details and open its input and output links.
7. Check in the Azure Blob Container (used by the Azure Storage Blob Connector) that the files, which have been uploaded, contain the text of the Tweets found.

