Moving Data to the Cloud

with Azure Data Factory

Table of Contents

Setup

Before working with the Data Factory exercises in this document, we need to set up the Azure environment first:

1. Create a new Resource Group to put all the exercise work in.
    a. In the Azure Portal, click on Resource Groups on the sidebar.
    b. Click the Add button.
    c. Give it a name in the Resource group box and click Review + Create at the bottom.
    d. Then click Create at the bottom.
2. Create an Azure SQL database in that resource group to put the data in.
    a. From the Azure Portal, click on SQL databases on the sidebar.
    b. Click the Add button.
    c. Change the Subscription as necessary
    d. Change the Resource Group to the one you just created.
    e. Enter a Database Name
    f. Click on Create new under the Server box.
    g. Enter a Server Name
    h. Enter a Server admin login name
    i. Enter a complex Password and Confirm it
    j. Write down the SA name and Password somewhere
    k. Click Select
    l. Click Review + Create
    m. Click Create
    n. Watch and wait as your deployment happens
    o. Open the newly created database
    p. Add a Firewall Rule
    q. Click Add client IP
    r. Click Allow access to Azure services to ON
    s. Click Save
3. Create a local SQL Server DB on your local machine.
    a. Launch SSMS
    b. Connect to your local server
    c. Right-click on Databases in Object Explorer and select New Database
    d. Give it a name
    e. Click Ok
4. Add these tables to both the Azure SQL database and the local SQL DB you created above:

```
CREATE TABLE [TableSource](

   [Id] [int] IDENTITY(1,1) NOT NULL,
   [Field1] [int] NULL,
   [Field2] [varchar](50) NULL,
PRIMARY KEY CLUSTERED
(
   [Id] ASC
)
```

```
)

CREATE TABLE [TableTarget](
    [Id] [int] IDENTITY(1,1) NOT NULL,
    [Field1] [int] NULL,
    [Field2] [varchar](50) NULL,
PRIMARY KEY CLUSTERED
(
    [Id] ASC
)
)

INSERT INTO TableSource (Field1, Field2)
    VALUES (123, 'Apple'),( 234, 'Orange'), (345, 'You glad it''s not another
fruit.');
```

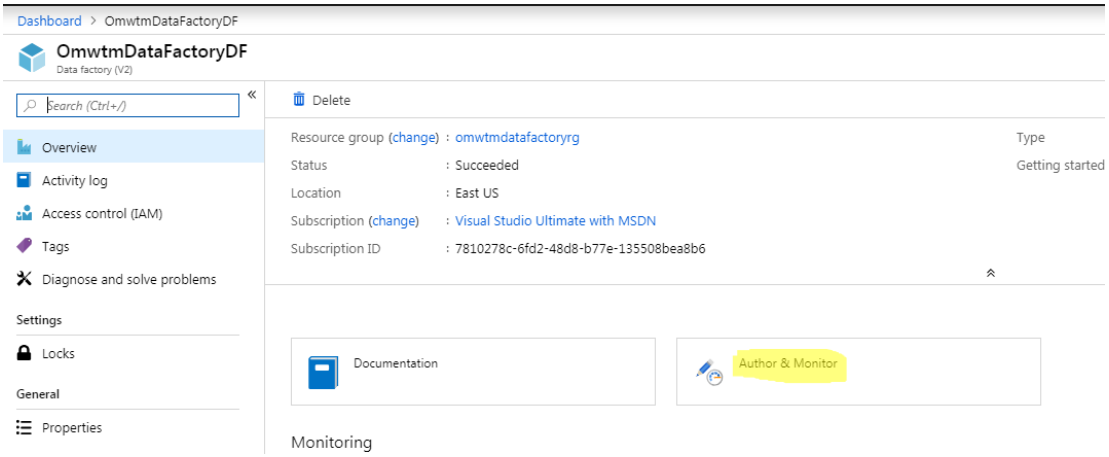
5. Create a Data Factory
    a. Open the Resource Group you created above.
    b. Click the Add button
    c. Search for Data Factory
    d. Select the Data Factory option
    e. Click Create
    f. Enter a name for your data factory
    g. Select the right subscription
    h. Select to Use existing Resource Group
    i. Pick the resource group you created above
    j. Click Create
6. Create a Storage Account
    a. Open the Resource Group you created above.
    b. Click the Add button
    c. Search for Storage
    d. Pick Storage Account
    e. Click Create
    f. Select the right subscription
    g. Pick the resource group you created above
    h. Enter a Storage account name
    i. Select the Cool Access tier
    j. Click Review + Create
    k. Click Create
7. Create an SSIS project in Visual Studio
    a. Make sure you have the SSDT installed in Visual Studio
    b. Make sure you have the Azure pack for SSIS installed too
    c. Open Visual Studio (2017 preferred, 2015 if you must, not 2013 or older.)
    d. Click on File | New | Project
    e. Select Business Intelligence | Integration Services
    f. Select Integration Services Project
    g. Give it a Name and Location
    h. Click OK

i. In the Package.dtsx SSIS package
j. Drag over a Data Flow Task
k. Double-click it to go to the Data Flow pane
l. Drag in an ADO NET Source
m. Dbl-click it to open it
n. Click New to create a connection manager
o. Click New… to create a New Connection
p. Enter LocalHost as the server name
q. Select the database you created above
r. Click OK
s. Click OK again
t. Select the table named TableSource you created above
u. Click on Columns on the left
v. Click OK
w. Drag in a FlatFile Destination
x. Click on the ADO NET Source
y. Click the Blue Line and connect it to the Flat File Destination
z. Dbl-click the Flat File Destination to open it
aa. Click New to create a connection manager
bb. Pick Delimited, Click OK
cc. Click Browse and pick a file name and location
dd. Check the Unicode box if your data has NVARCHAR's in it
ee. Click on Columns on the left
ff. Change the Column delimiter to **Vertical Bar {|}**
gg. Click OK
hh. Click Mappings
ii. Click OK
jj. Click Save All
kk. Done (for now)

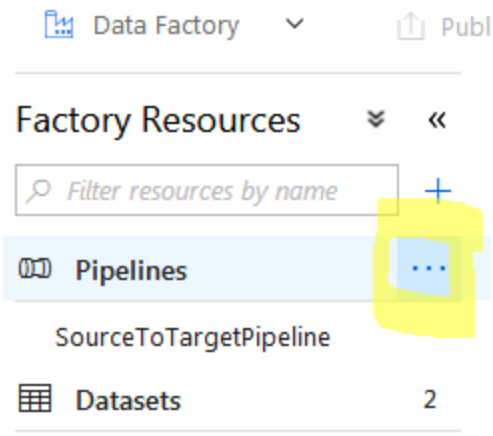
Part 1 – Setting up a simple Copy Activity

How can we use Data Factory to copy the data from the Source to the Target?

1. Open the Data Factory blade in the Azure Portal
2. Click on the Author & Monitor button

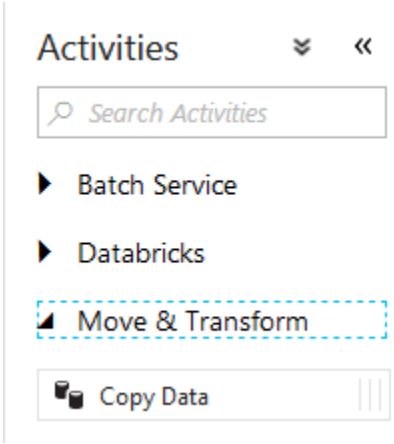


3. Or, navigate to https://datafactoryv2.azure.com, select your Azure Active Directory, Subscription and Data Factory instance.
4. Start by clicking the Pencil icon in the left nav bar. (It's the second icon)
5. Click the Ellipses next to the Pipelines menu:



6. Select Add Pipeline
7. Enter a Name for your Pipeline on the General tab, such as **`MySourceToTargetPipeline`**

8. Next, expand the Move & Transform Activities menu:



9. Drag the Copy Data activity into the Pipeline
10. The Copy Data task has a Source and a Sink (target). Click on the Source tab.
11. Next to the Source dataset dropdown, click +New
12. Select the data source type. Pick 'Azure SQL Database'
13. Click Finish
14. Now you are creating a Dataset named AzureSqlTable1.
15. Click on the Connection tab
16. Next to the Linked service dropdown, click +New
17. Now you are configuring the database connection to your Azure SQL Database
18. Select the Azure Subscription, Server name, and Database name to connect to
19. Enter the User Name and Password to use to connect
20. Click Test connection.
21. When it succeeds, click Continue
22. Back on the Dataset Connection tab, pick the Source table to copy from. If you have a lot of tables, the dropdown may be slow, click the Edit checkbox to be able to type in a table name instead. At this point, you have to pick a table, but you can also enter a query at a later point.
23. Click on Preview data to see the first few rows of data
24. Click the Schema tab
25. Click Import schema
26. Now, click over onto the Pipeline tab at the top (MySourceToTargetPipeline)
27. Click on the Sink tab in the middle
28. Next to the Sink dataset dropdown, click +New
29. Select the data source type for the target dataset. Pick 'Azure SQL Database'
30. Now you are creating a dataset named AzureSqlTable2.
31. Click on the Connection tab
32. Select the Azure SQL Database Linked service you already created (typically, you wouldn't do this, you would create another Linked service for the target database)
33. Select the target table
34. Click on Preview data
35. Click the Schema tab
36. Click Import schema
37. Now, click over onto the Pipeline tab (MySourceToTargetPipeline)

38. Enter **`TRUNCATE TABLE <TARGET TABLE NAME>`** in the Pre-copy script box
39. If you're using the tables created above, this would be:
    `TRUNCATE TABLE dbo.TableTarget`
40. Click the Mapping tab
41. Click Import schemas
42. Make sure the fields map properly. This is not the place for transformations. If the data types don't match, the mappings might not be valid.
43. Click on Settings
44. Note the selections available. Leave the settings at Auto, Auto and Abort. If you enable staging, you will need to setup a blob storage account for the Copy activity to use as a staging area.
45. There is no Save button in Data Factory, unless you bind it to an Azure DevOps GIT repository
46. Click on Validate All
47. If all is well, click on Publish All
48. When the deployment succeeds, we're done.
49. You can run the Pipeline by clicking Pipeline tab at the top, selecting Add trigger, and selecting Trigger Now.
50. To see the Pipeline's progress, you can click the red circle on the left nav bar.

Part 2 – Azure Vault: Securing your secrets

First, create a Key Vault resource in the resource group we created above using the Azure Portal.

1. Open the Resource Group you created above.
2. Click the Add button
3. Search for Vault
4. Choose Key Vault
5. Click Create
6. Give it a name
7. Change the subscription
8. Select the resource group
9. Click Create

Second, create secrets (not keys) for your connection strings and SQL account passwords.

1. Open the Key Vault
2. Select Secrets on the left
3. Click Generate/Import
4. Enter the secret name: DbPassword
5. Enter the secret value: <the password you created above>

Third, hop over to Data Factory and set up a Linked Service to the Azure Key Vault:

1. Open your Data Factory resource
2. Click on Author & Monitor
3. Click the pencil icon to Author/Edit your Factory Resources
4. Click on Connections in the bottom left corner
5. Click on +New on the Linked Services tab

6. Locate and click on the Azure Key Vault data store
7. Give your Linked Service a name
8. Select your Subscription
9. Select your Key Vault
10. Click on the little Grant data factory managed identity access to your Azure Key Vault link
11. In the Key Vault blade, Access policies tab, click +Add new
12. Leave the template blank
13. Click Select principal
14. Search for your Data Factory resource by name
15. Click on your Data Factory resource
16. Click Select at the bottom of the screen
17. Click on Secret permissions
18. Select the checkbox next to Get (required) and List (optional)
19. Click Ok at the bottom of the screen
20. Click Save at the top of the screen

Fourth, use the Key Vault linked service in your other Data Factory Linked Services to obtain the necessary Connection Strings and Passwords from the Secrets you set up in the Key Vault.

1. Back in the Data Factory, navigate to the Connections (bottom left corner)
2. Publish the newly created Key Vault connection by clicking the Publish All button
3. Then, create a new Linked Service
4. Select Azure SQL Database as the Data Store
5. In the Connection String area:
6. Select your Subscription
7. Select your Server
8. Select your Database
9. Enter the SQL User name
10. Click on the Azure Key Vault option
11. Select your Key Vault linked service
12. Enter the Secret name for the SQL User's password
13. Leave the Secret version blank
14. Click Test Connection
15. Click Finish if everything's Ok

Part 3 – Using Variables and Controlling the Data Flow

Using Variables

Using a variable in a Copy Activity is straightforward enough, once you know the syntax.

1. Start with the Copy pipeline created above.
2. Select the Variables tab to define the variable
3. Click +New
4. Enter a variable name (**`SourceID`**) and select String as the type. Your only choices are String, Boolean or Array.

5. Now, select the General activity menu (on the left) and drag a Set Variable activity into the pipeline
6. Click on the Variables tab
7. Select your variable from the dropdown
8. Give it a value. Enter the number 1 in the value box.
9. Next, grab the little green box hanging off the right side of the Set Variable activity in the diagram.
10. Drag that box over to the Copy Data activity and drop it
11. Select the Copy Data activity in the diagram
12. Select the Source tab
13. Select the Query radio button
14. Click in the box and then click under the box on the Add dynamic content link
15. In the middle of the Dynamic Content screen, click String Functions and click **concat()**
16. At the top of the screen, between the parentheses, enter: **'select * from TableSource where Id = ',** (Note the comma)
17. Then click on the Variable SourceId at the bottom of the screen
18. The content will change to:
    @concat('select * from [TableSource] where Id = ', variables('SourceID'))
19. Click Finish
20. You can now Publish and Trigger your Pipeline to test it out

Controlling the Data Flow

Let's say we have a table with fields that name the source table and target table for our copy operation, defined like this:

```
CREATE TABLE LookupTable (
SourceTable SYSNAME,
TargetTable SYSNAME
)

INSERT INTO LookupTable (SourceTable, TargetTable)
    VALUES ('TableSource', 'Tabletarget');
```

How will we load the table names into Data Factory and do the copy activity for each pair?

1. In Data Factory, click Author & Monitor
2. Then click the Pencil to edit the Factory Resources
3. Create a new Pipeline
4. Grab the Lookup activity from the General section and drop it in the pipeline
5. Click on the Settings tab
6. Click the +New to add a new source dataset
7. Select Azure SQL Database
8. Click Finish
9. Click the Connection tab
10. Select your Azure SQL Linked Service (the one that uses the Key Vault)
11. Select your Lookup table
12. Switch back to the pipeline editor

13. Uncheck the First row only checkbox
14. Next, grab the For Each activity from the Iteration & Conditionals section and drop it in the pipeline
15. Drag the green box over from the Lookup activity to the For Each activity to connect them
16. Click on the For Each activity again
17. Click on the Settings tab
18. Set the batch count to the number of parallel batches you want to run (say 5)
19. Click in the items box and then click the link below it to Add dynamic content
20. Enter the expression: `@activity('Lookup1').output.value`
    This pulls the value of the output dataset from the Lookup1 activity as an array.
21. Click Finish
22. Click on the Activities tab
23. Click the Add activity button
24. Grab the Copy Data activity from the Move & Transform section and drop it on the work surface
25. Click on the Source tab
26. Pick the Source Dataset (we will override the table in the Dataset by using a Query)
27. Select the Query option
28. Click in the Query textbox and click the link below it to Add dynamic content
29. Enter the expression: `SELECT * FROM @{item().SourceTable}`
    This uses the item() from the Copy Data activity's context (it is inside the ForEach activity, so item() refers to the current item/row in the array we looked up above). SourceTable is the name of the field from the lookup table.
30. Click Finish
31. Click on the Sink tab
32. Click the +New to create a new dataset
33. Pick Azure SQL Database
34. Click Finish
35. Click the Connection tab
36. Select the Linked service for the target database
37. Click on the Parameters tab
38. Click New
39. Enter the name of the parameter: **TargetTable**
40. Click back on the Connection tab
41. Check the Edit checkbox
42. Click in the Table textbox and click the link below it to Add dynamic content
43. Enter the expression: `@{dataset().TargetTable}`
    This creates a property that we can fill in from the pipeline.
    Data Factory will mark it invalid for now as we haven't finished yet
44. Switch back to the pipeline tab
45. Now that you have selected a dataset with an expression in it, you have to put in the value for the TargetTable property
46. Click in the Value textbox next to the TargetTable property and click the link below it to Add dynamic content
47. Enter the expression: `@{item().TargetTable}`

TargetTable is the name of the field from the lookup table.

48. Lastly, put in a Pre-copy script to Truncate the table before loading it
49. Click in the Pre-copy script textbox and click the link below it to Add dynamic content
50. Enter the expression: **`TRUNCATE TABLE @{item().TargetTable}`**
51. Done.
52. Click the Validate All button.
53. Click the Publish All button.
54. Wait for the Pipeline to Publish, then click Trigger the pipeline to run now to test it.
55. Click on the red dot icon to view the pipeline in action.

For more info, see here.

Part 4 – The Self-Hosted Integration Runtime

Installing and Configuring the Runtime

1. Uninstall "On-premises data gateway" if it's there.
2. OPTION A: Use the GUI
    a. In Data Factory, click Author & Monitor
    b. Then click the Pencil to edit the Factory Resources
    c. Then select Connections at the bottom left of the screen
    d. Then select the tab for Integration Runtimes
    e. Click New
    f. Click the Perform data movement option, click Next
    g. Click Self-Hosted, click Next
    h. Give your Self-Hosted runtime a name, click Next
    i. If you haven't installed the runtime on the local server/machine, then pick Option 1
    j. Or, pick Option 2 and install it yourself
    k. Then Copy Key1 to the clipboard and paste it into the Registration screen in the on-premises Integration Runtime

    OPTION B: Use PowerShell
    a. https://docs.microsoft.com/en-us/azure/data-factory/create-self-hosted-integration-runtime
    b. In PowerShell, Run this code to create the Self-Hosted Integration Runtime in Azure:

```
Import-Module Az
$dataFactoryName = "<DataFactory>"
$resourceGroupName = "<resourcegroup"
$selfHostedIntegrationRuntimeName = "<IntegrationRuntime>"
Login-AzAccount
Select-AzSubscription -SubscriptionName "<Subscription>"

Set-AzureRmDataFactoryV2IntegrationRuntime -ResourceGroupName
$resourceGroupName -DataFactoryName $dataFactoryName -Name
$selfHostedIntegrationRuntimeName -Type SelfHosted -Description
"selfhosted IR description"
```


c. Download and install the self-hosted integration runtime on a local machine.
d. In PowerShell, Run this code to get a key (Key1 or Key2):

```
Get-AzureRmDataFactoryV2IntegrationRuntimeKey -ResourceGroupName
$resourceGroupName -DataFactoryName $dataFactoryName -Name
$selfHostedIntegrationRuntime
```

e. Copy the key to the clipboard and paste it into the Registration screen in the on-premises Integration Runtime.

> *NOTE: If you need to CHANGE the Integration Runtime Key, there is a PowerShell script in the Runtime's installation folder named "C:\Program Files\Microsoft Integration Runtime\3.0\PowerShellScript\RegisterIntegrationRuntime.ps1". Run this in a PowerShell window that is Running as Administrator. Run the script, using this command line:*
> `.\RegisterIntegrationRuntime.ps1 "<gatewayKey1>" -IsRegisterOnRemoteMachine false`

3. Now that you have set up the SHIR (Self-hosted Integration Runtime), it can access any data that the machine it is installed on can access. For example, if you have the IBM Client Access ODBC drivers set up properly, you can access a DB2 database. If you have the Oracle drivers set up, you can access an Oracle database. Etcetera.

Using the Self-Hosted Integration Runtime

1. On the server configured as the SHIR, set up an ODBC connection to a database on-premise
2. In the Azure Key Vault, create two secrets:
   a. Create a secret for the connection string (something like: **DSN=ODBCSOURCE**)
   b. Create a secret for the login password
3. Back in the Data Factory blade, click Author & Monitor
4. Create a new Dataset
5. Select the ODBC data store (or whatever you have drivers for on the host machine)
6. Click Continue
7. Give your Linked Service a name
8. Select your SHIR for the integration runtime
9. Enter the secret name for the connection string from the Key Vault
10. Enter the User name to connect with
11. Enter the secret name for the password from the Key Vault
12. Click Test connection
13. Click Finish if everything is Ok

Part 5 – Uploading files to Blob storage and Importing them

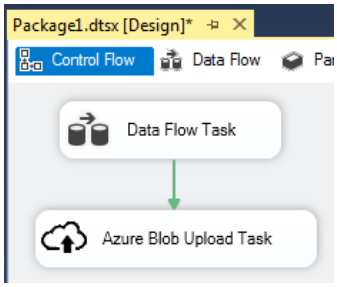
The Self-Hosted Integration Runtime is fine for small tables. Once you get over 10-50 million rows, it's just too slow. That's when you need to extract the tables to files and upload them to blob storage in Azure. Then you can bring them into Azure SQL with data factory or a bulk insert stored proc.

Extracting and Uploading

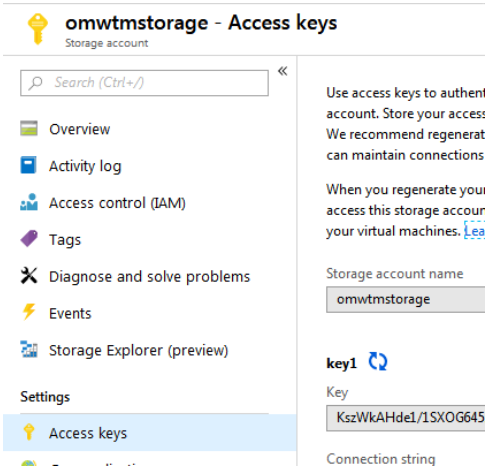
One way to do the extract to a file and upload to blob storage is with the [Azure Feature Pack for Integration Services (SSIS)].

Setup an Extract Data flow as you normally would to extract your table source to a text file. (See the Setup instructions for specifics.)

1. Then, drag in an Azure Blob Upload Task to copy the file up to blob storage:

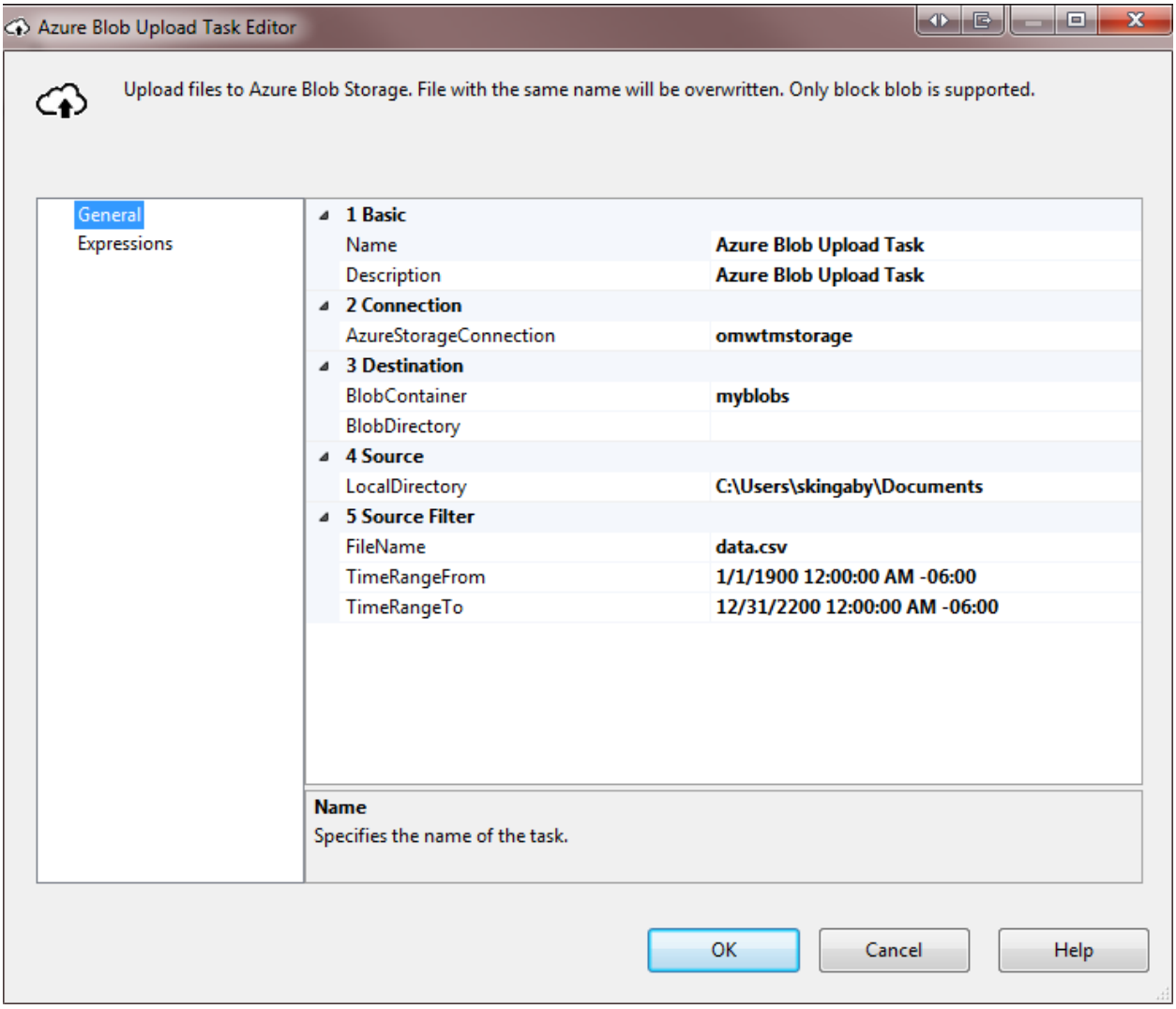


2. Dbl-click to open the Azure Blob Upload Task
3. Pick item 2, AzureStorageConnection
4. Click the dropdown at the right end of that row
5. Select New connection
6. Enter the Storage account name from Azure that you created above
7. Open the Storage account in the Azure Portal
8. Select Access keys on the left



9. Copy Key1's Key to the clipboard
10. Paste it into the Account key slot in SSIS.
11. Go back to the Azure Portal
12. Click on Overview in the Storage account

13. Click on Blobs in the middle
14. Click Container to add a new container
15. Give it a name
16. Click OK
17. Back in SSIS
18. Configure the Azure Blob Upload Task
19. Enter the Blob Container name in the BlobContainer slot
20. Change the Source: LocalDirectory to the location of your exported file from above
21. Enter the file name from above in the FileName slot:



Run the package to upload the data file to blob storage.

Importing the Data File

Using Data Factory

One option for importing the data file is to use Data Factory.

1. In Data Factory, click Author & Monitor

2. Then click the Pencil to edit the Factory Resources
3. Create a new Pipeline
4. Drag the Copy Data activity into the Pipeline
5. Click the Source tab
6. Create a new dataset
7. Select Azure Blob Storage
8. Click Finish
9. Click the Connection tab
10. Create a new Linked Service
11. Enter a Name for the service
12. Leave the Connect via and Authentication method at the defaults
13. Select the Azure subscription
14. Select the Storage account name
15. Click Test connection
16. Click Finish if all is well
17. Browse to find the blob folder you uploaded to
18. Enter the file name (data.csv)
19. Select the compression type (you can gzip the files in SSIS before you upload to reduce the time taken to upload)
20. Click the Detect Text Format button
21. Check that the File format settings are correct
22. Expand the Advanced settings and verify those too (You might have to remove the quote character if you didn't use one)
23. Click the Preview data button
24. The data should look good
25. Click the Schema tab
26. Click the Import schema button
27. Change the data types where necessary
28. Switch over to the pipeline
29. Uncheck the Copy file recursively
30. Click the Sink tab
31. Select the Sink dataset
32. Enter the TRUNCATE TABLE statement in the Pre-copy script if you want to
33. Click on the Mapping tab
34. Click the Import Schemas button
35. Validate the changes you've made
36. Publish the changes
37. Trigger the Pipeline to test it
38. Done

Using SQL Bulk Insert

A second option – that may be faster in some circumstances – for importing the data file is to use the SQL BULK INSERT statement in a stored procedure. This requires four things.

First, create a Database Scoped Credential to access the blob storage. In the Azure portal:

1. Open the Storage account resource
2. Select the Shared access signature option under Settings
3. Under Allowed services, uncheck everything except Blob
4. Under Allowed resource types, uncheck everything except Object
5. Under Allowed permissions, uncheck everything except Read
6. The Start date/time defaults to now, leave that
7. The End date/time defaults to 8 hours from now. Change the End date to a later date. Like a year from now.
8. Leave the Allowed IP addresses blank
9. Leave the Allowed protocols as HTTPS only
10. Click the Generate SAS and connection string button
11. Copy the Connection string and SAS token to Notepad, we'll need them in a minute
12. Switch over to SSMS and connect to your database as a database owner
13. Open a query window and run the following code:

```
CREATE DATABASE SCOPED CREDENTIAL AzureBlobStorageCredential
WITH IDENTITY = 'SHARED ACCESS SIGNATURE',
SECRET = 'sv=2018-03-28&ss=b&srt=o&sp=r&se=2019-02-28T02:14:55Z&st=2019-02-
27T18:14:55Z&spr=https&sig=****'
```

Note: The Secret is the SAS Token minus the leading question mark.

Note: If you get an error saying, 'Please create a master key in the database or open the master key in the session before performing this operation', then run the following statement in your database:

```
CREATE MASTER KEY;
```

Second, create an External Data Source for the blob storage account. In the Azure Portal:

1. Open the Storage account resource
2. Click on the Blobs service icon
3. Click on your blob container to open it
4. Click on Properties under Settings
5. Copy the URL to Notepad, we'll need it in a minute
6. Switch over to SSMS and connect to your database as a database owner
7. Open a query window and run the following code:

```
CREATE EXTERNAL DATA SOURCE AzureBlobStorage01
WITH (TYPE = BLOB_STORAGE,
LOCATION = 'https://<enter the URL from step 5 above>',
CREDENTIAL= AzureBlobStorageCredential);
```

Third, create a stored procedure that uses the External Data Source to Bulk Insert the data. For example:

```
CREATE PROCEDURE BulkInsertDataFromBlob
AS
BEGIN

      TRUNCATE TABLE TableTarget;

      BULK INSERT TableTarget
      FROM 'data.csv'
      WITH (DATA_SOURCE = 'AzureBlobStorage01',
```



```
            FIELDTERMINATOR = '|',
            FIRSTROW=1,
            TABLOCK,
            BATCHSIZE=50000);

END;
```

Fourth, create a Data Factory job to execute the stored procedure:

1. In Data Factory, click Author & Monitor
2. Then click the Pencil to edit the Factory Resources
3. Create a new Pipeline
4. Give it a name
5. Drag a Stored Procedure activity from the General section into the Pipeline
6. Click on the SQL Account tab
7. Select the Linked service of your target database
8. Click on the Stored Procedure tab
9. Select the Stored Procedure you created above
10. Publish the Pipeline
11. Now you can trigger it to test it

Part 6 – Triggering Data Factory Activities

Before using the Events to trigger Data Factory activities, you need to register the Microsoft.EventGrid resource provider with your Subscription. In the Azure portal:

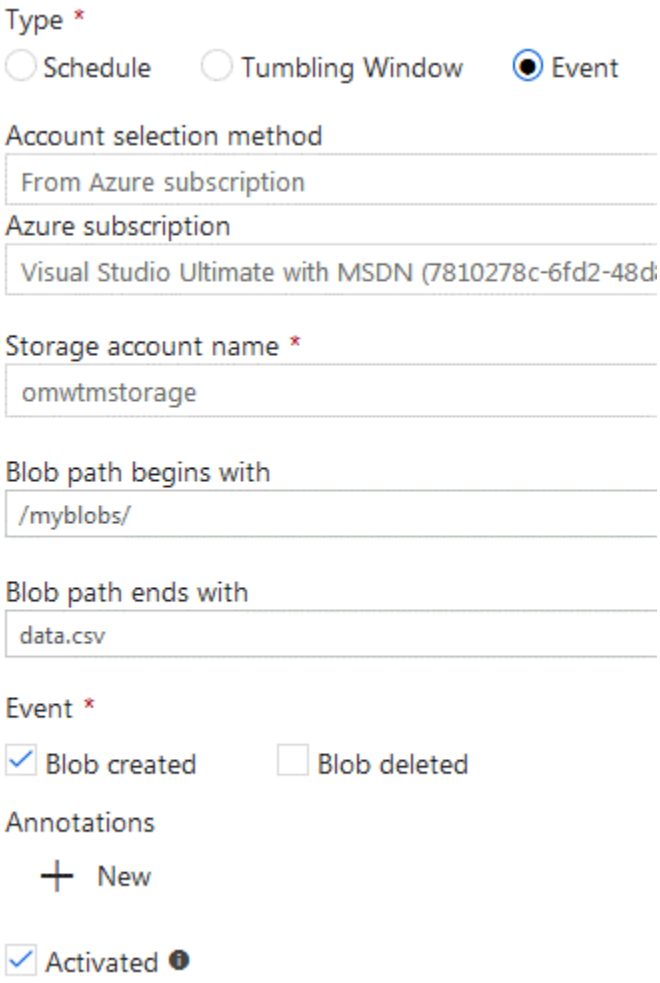
1. Select Subscriptions.
2. Select the subscription you're using for Event Grid.
3. Under Settings, select Resource providers.
4. Find Microsoft.EventGrid.
5. If not registered, select Register.

It may take a moment for the registration to finish. Select Refresh to update the status. When Status is Registered, you're ready to continue.

Next, you need to create a Trigger in Data Factory:

1. In Data Factory, click Author & Monitor
2. Then click the Pencil to edit the Factory Resources
3. Select Triggers in the bottom left
4. Click +New to create a Trigger
5. Give your trigger a name
6. Click on Event
7. Select your Subscription
8. Select your Storage Account name
9. Enter the name of the container inside slashes as the 'Blob path begins with'. E.g. /myblobs/
10. Enter the file name for the 'Blob path ends with'. E.g. data.csv
11. Check the Blob created checkbox

12. Check the Activated checkbox.
    The dialog will look like this:



13. Click Finish

Finally, you can apply the Trigger to the Pipeline:

1. Edit the Pipeline you want to run when the blob file shows up
2. Click on Trigger
3. Click on New/Edit
4. Choose the trigger you just created above
5. Click Next
6. Click Finish

Now you can test the process by uploading a data.csv file and viewing the Pipeline Activity in Data Factory.