Microsoft

Operationalizing Analytics: Building Predictive Pipelines with Azure Data Factory

Contents

Overview

Summary

This lab articulates how Azure Data Factory (ADF) can be utilized to create an end-to-end production grade solution that uses:

- Hive (Hadoop) on-demand for data partitioning/aggregation/joining big datasets
- Azure ML web services as the analytical engine to do batch scoring (probability of churn)
- Azure SQL DB to store the cleaned data with probability of churn
- Power BI to present the analytical results

Prerequisites

- Completed an Introduction to Azure Machine Learning

Business Case

In this lab we have three datasets from a fictitious mobile telecommunications company:

1. Call Detail Record (CDR) – this is produced by a telephone exchange that documents the details of each telephone call that passes though the exchange. The record contains details on the number calling, the number called, time, date, duration, etc.
2. Customer Information – Details pertinent to the customer that are refreshed each month e.g. penalty to switch, unpaid balance, whether or not the customer uses the internet service, whether the customer churned or not (1=churned, 0=not churned) in the month.
3. Sample of Aggregated/Merged Data – We have a sample of data where the customer information data has been joined onto an “aggregated dataset” containing rolled up information on the customers calling behavior e.g. average number of minutes per call, total number of calls per minute. We will use this dataset to produce a predictive model in Azure Machine Learning.

Each month the telecommunications company would like to predict which customers have a high probability of churning. We will use Azure Data Factory to orchestrate a predictive pipeline.

Please take a few moments to look at these input datasets, which can be found at the following location in this lab folder in the **Data** sub-folder.

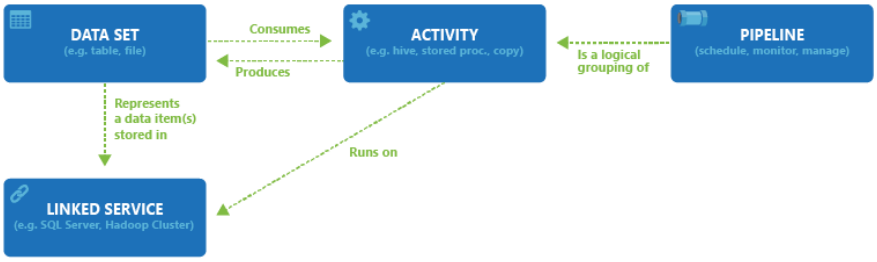
Azure Data Factory

Azure Data Factory is a cloud-based data integration service that orchestrates and automates the movement and transformation of data. Just like a manufacturing factory that runs equipment to take raw materials and transform them into finished goods, Data Factory orchestrates existing services that collect raw data and transform it into ready-to-use information.

Data Factory works across on-premises and cloud data sources and SaaS to ingest, prepare, transform, analyse, and publish your data. Use Data Factory to compose services into managed data flow pipelines to transform your data using services like Azure HDInsight (Hadoop) and Azure Batch for your big data computing needs, and with Azure Machine Learning to operationalize your analytics solutions.

Azure Data Factory has a few key entities that work together to define the input and output data, processing events, and the schedule and resources required to execute the desired data flow.

The 4 entities in Data Factory are outlined below:



- **Activities** define the actions to perform on your data. Each activity takes zero or more datasets as inputs and produces one or more datasets as outputs. An activity is a unit of orchestration in Azure Data Factory. For example, you may use a Copy activity to orchestrate copying data from one dataset to another. Similarly, you may use a Hive activity which will run a Hive query on an Azure HDInsight cluster to transform or analyse your data. Azure Data Factory provides a wide range of data transformation, analysis, and data movement activities.

- **Pipelines** are a logical grouping of Activities. They are used to group activities into a unit that together perform a task. For example, a sequence of several transformation Activities might be needed to cleanse log file data. This sequence could have a complex schedule and dependencies that need to be orchestrated and automated. All of these activities could be grouped into a single Pipeline named “CleanLogFiles”. “CleanLogFiles” could then be deployed, scheduled, or deleted as one single unit instead of managing each individual activity independently.

- **Datasets** are named references/pointers to the data you want to use as an input or an output of an Activity. Datasets identify data structures within different data stores including tables, files, folders, and documents.

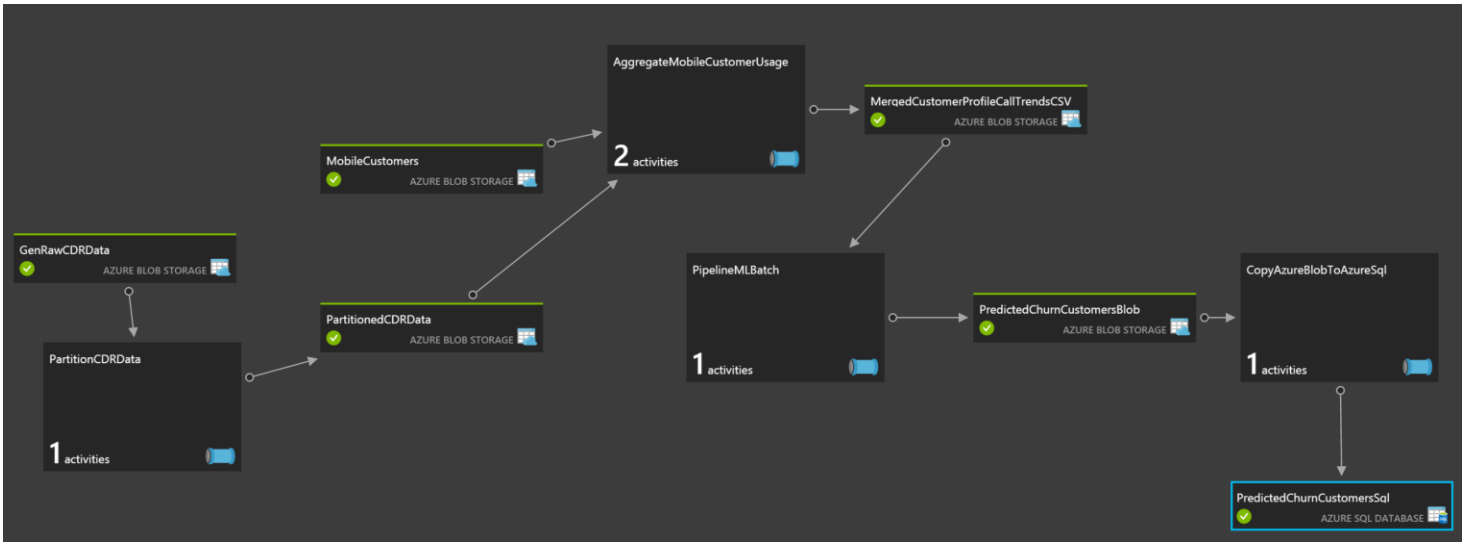
- **Linked Services** define the information needed for Data Factory to connect to external resources. Linked services are used for two purposes in Data Factory:

    1. To represent a data store including, but not limited to, an on-premises SQL Server, Oracle DB, File share or an Azure Blob Storage account. As discussed above, Datasets represent the structures within the data stores connected to Data Factory through a Linked service.

    2. To represent a compute resource that can host the execution of an Activity. For example, the “HDInsightHive Activity” executes on an HDInsight Hadoop cluster.

With the four simple concepts of datasets, activities, pipelines and linked services, you are ready to get started!

Our finished Azure Data Factory Pipeline will look like the following



Looking at this pipeline left-to-right and following the arrows we have:

- **GenRawCDRData** – this is a csv **table** in blob storage containing the Call Detail Record data

- **PartitionCDRData** – This is a **pipeline** containing 1 **activity**, which is to partition the data using a Hive Query into year/month. This pipeline **consumes** 1 dataset (**GenRawCDRData**) and **produces** 1 dataset (**PartitionedCDRData** i.e. the partitioned data from the Hive query)

- **MobileCustomers** – This is a csv “table” in blob storage containing the customer information.

- **AggregateMobileCustomerUsage** – This is a pipeline containing two activities:

ACTIVITY	TYPE
AggregateMobileCustomerUsage	HDInsightHiveActivity
RenameHiveOutput2CSV	Copy Actvity

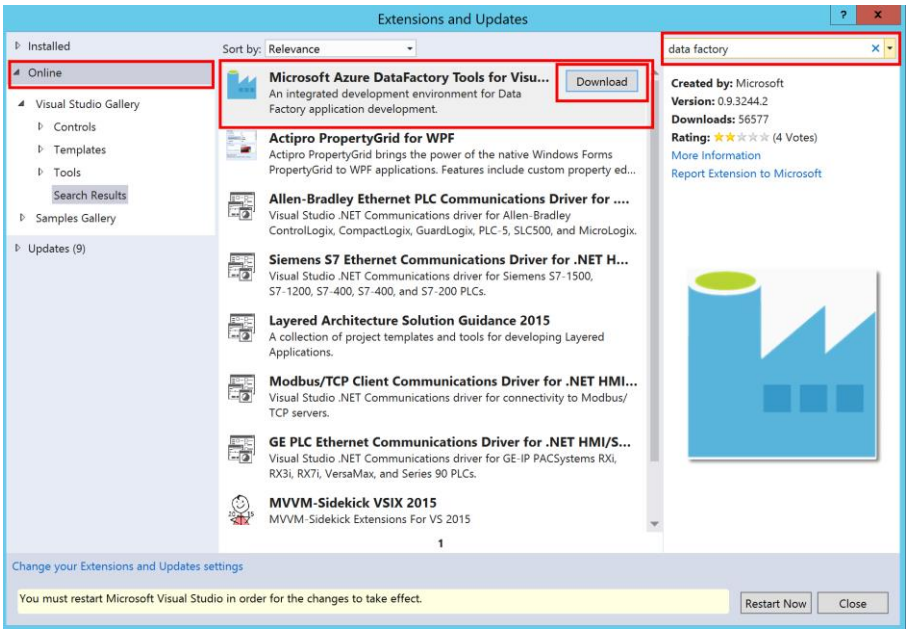
  The first activity consumes **PartitionedCDRData** and **MobileCustomers** and produces **MergedCustomerProfileCallTrends** by aggregating the **PartionedCDRData** to find the total number of consumed minutes per month and then joining on to **MobileCustomers**. The second activity (copy activity) simply renames the table as a CSV file.

- **PipelineMLBatch** – this pipeline containing 1 activity that is the linked service to an Azure Machine Learning web-service. This consumes the aggregated and merged dataset produced by the Hive query and produces a csv “table” in blob storage containing the scored data.

- **CopyAzureBlobToAzureSql** – This simply copies the output from Azure Machine Learning (i.e. csv in blob storage) into a table in an Azure SQL database.

Install Data Factory Tools for Visual Studio

1. Open Visual Studio
2. On the menu bar go to **Tools** > **Extensions and Updates**
3. Click on **Online** and then search for **data factory** > Download **Microsoft Azure DataFactory Tools for Visual Studio** > Once the download has finished you will be prompted to Install > Click **Install**.



4. Click on **Restart Now**.

Create the Azure ML Experiment & Web Service

1. Log in to Azure ML - https://studio.azureml.net/

2. For this lab you have two options:

   OPTION A: You can load in a pre-built experiment from the Cortana Intelligence Gallery or

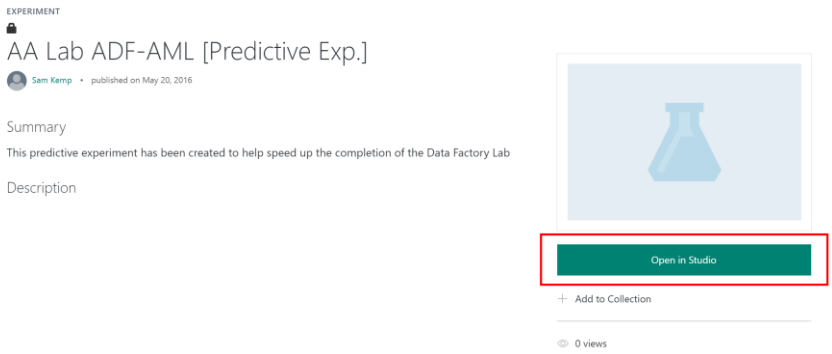
   OPTION B: Build the Azure ML experiment from scratch

OPTION A

3. Please go to the following web address:

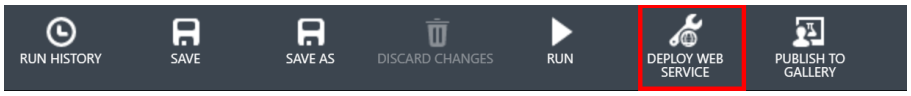
   http://gallery.cortanaintelligence.com/Experiment/AA-Lab-ADF-AML-Predictive-Exp-1

   and click **Open in Studio**



4. Once the experiment has opened in your workspace, click **Run**:

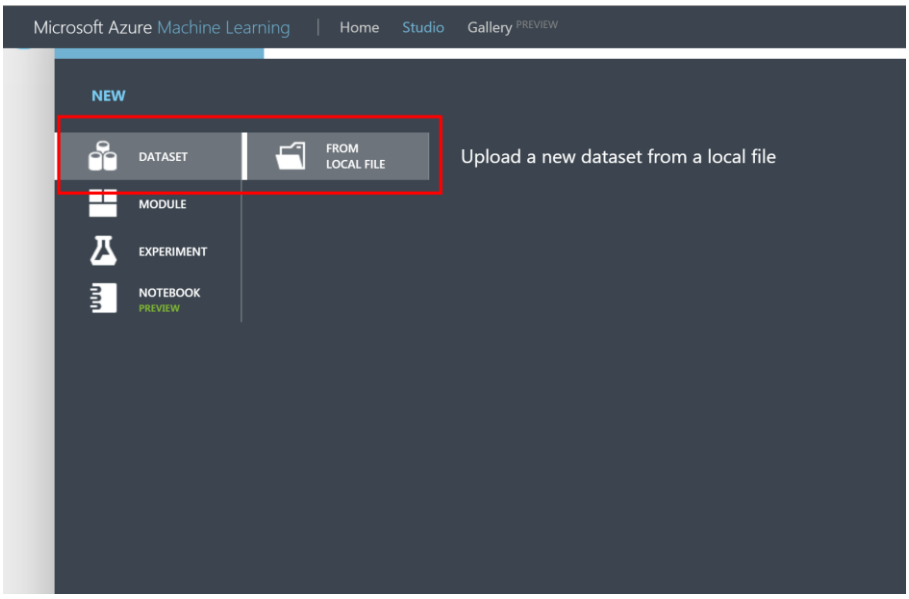
5. Once the predictive experiment has finished running you will now be enabled to **Deploy Web Service**.



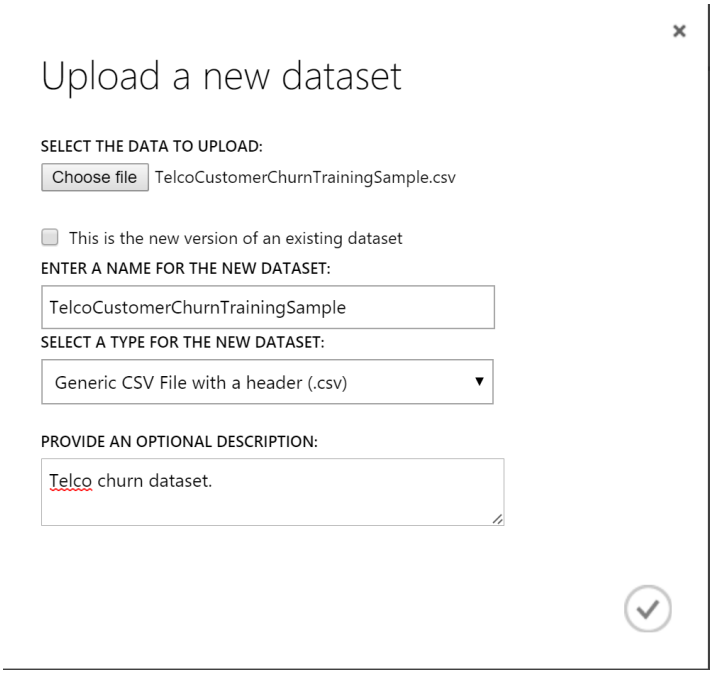
6. With the predictive experiment now deployed as a web service you can move to the next section in this document entitled **Create & Setup the Azure SQL Database.**

OPTION B

1. Upload the training data (**TelcoCustomerChurnTrainingSample.csv**) to Azure ML by clicking **New** in the bottom left corner, **DATASET** and **FROM LOCAL FILE**:

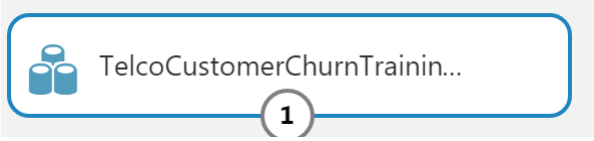


Click on **Choose File** and select the csv file, give the dataset a name, set the type to **Generic CSV File with a header (.csv).**



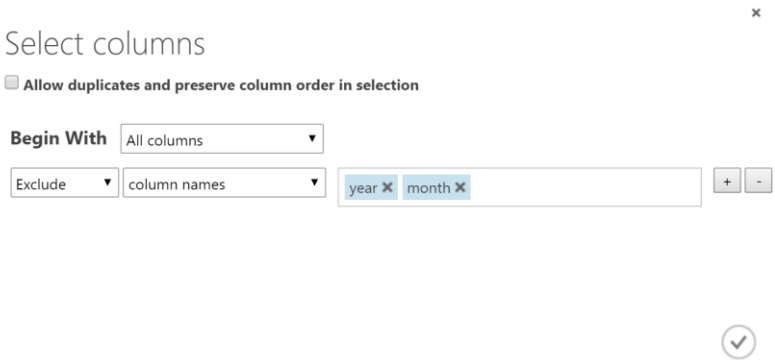
2. On the **Experiments** page click New (bottom left corner) and then **Blank Experiment**. Rename the experiment by clicking on the title and entering a meaningful name (recommend adding a date, for example Telco Customer Churn - 20151201).

3. Click on **Saved Datasets** and then **My Datasets**. You should see the file you just uploaded – drag this module onto the canvas.

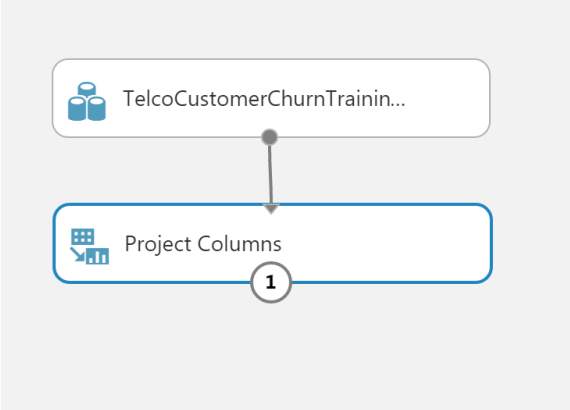


Right-click and **Visualize** the data. We have two unnecessary data columns: year and month.

4. Add a **Project Columns** module (from Data Transformation > Manipulation > Project Columns) onto the canvas connect with the **Dataset** module. On the **Project Columns** module properties blade click **Launch column selector**. On the **Begin with** dropdown select **All columns**, then **Exclude Column Names** and add **year** and **month**, i.e.



Click **Run**. We recommend adding a comment to the Projection Task 'remove Year and Month from Projection'.



5. Add a **Metadata Editor** module to the canvas - we want to change the annualincome and age columns to Integer. On the module properties **Launch Column Selector** and then Begin With **No columns**, include annualincome and age then on click on the tick. On the **Data Type** drop-down select **Integer**. Add a comment to the module my right-clicking and selecting **Edit**

Comment. Click **Run**. Your experiment should now look as follows:

6. We now want to do some data cleaning. Start by dragging a **Clean Missing Data** module onto the canvas connecting with the prior **Metadata Editor** module. Confirm the task has the following default values:

    a. Selected Columns: All Columns

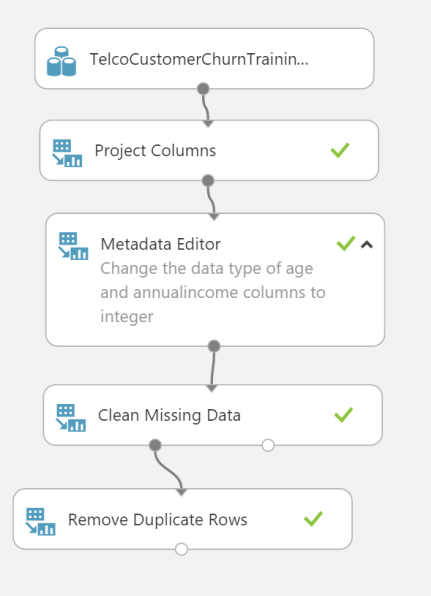
    b. Minimum missing value ratio: 0

    c. Maximum missing value ratio: 1

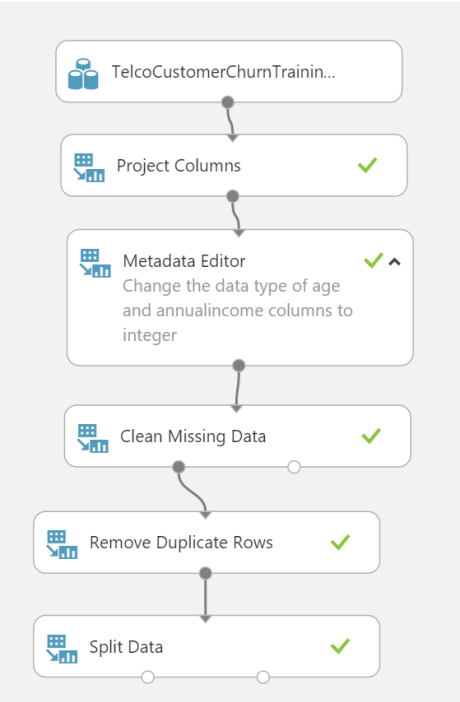
    d. Cleaning Mode: Custom Substitution value

7. Click **Run**.

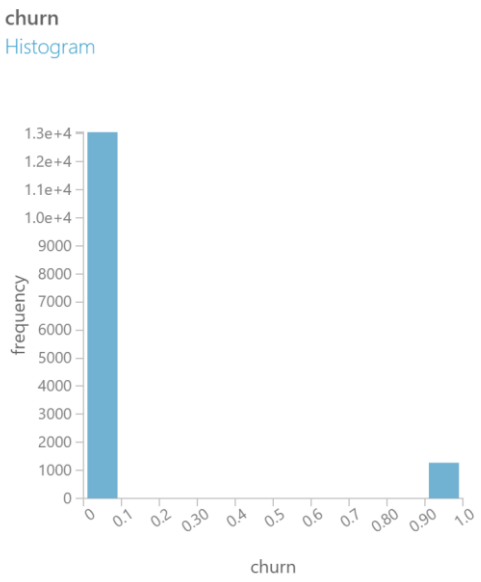
8. Add a **Remove Duplicate Rows** module and in the properties of that module select **All Columns** using the **Launch column selector**. Ensure **Retain first duplicate row** is selected. Connect to the prior **Clean Missing Values** module and click **Run**. Your experiment should now look as follows:



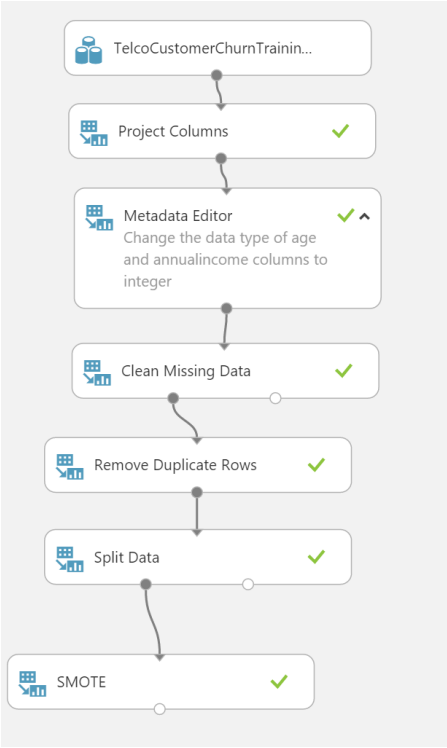
9. Now we want to split the data into a training/test set. To achieve this drag onto the canvas a **Split Data** module (Data Transformation > Sample and Split > Split Data). In the module properties set the **Splitting mode** to **Split Rows** and the **Fraction of rows in the first output dataset** to 0.7. Connect the input port to the prior **Remove Duplicate Rows** module and click **Run**. Your experiment will now look as follows:



10. Visualizing the left-hand (training) dataset we see there is a class imbalance on churn.

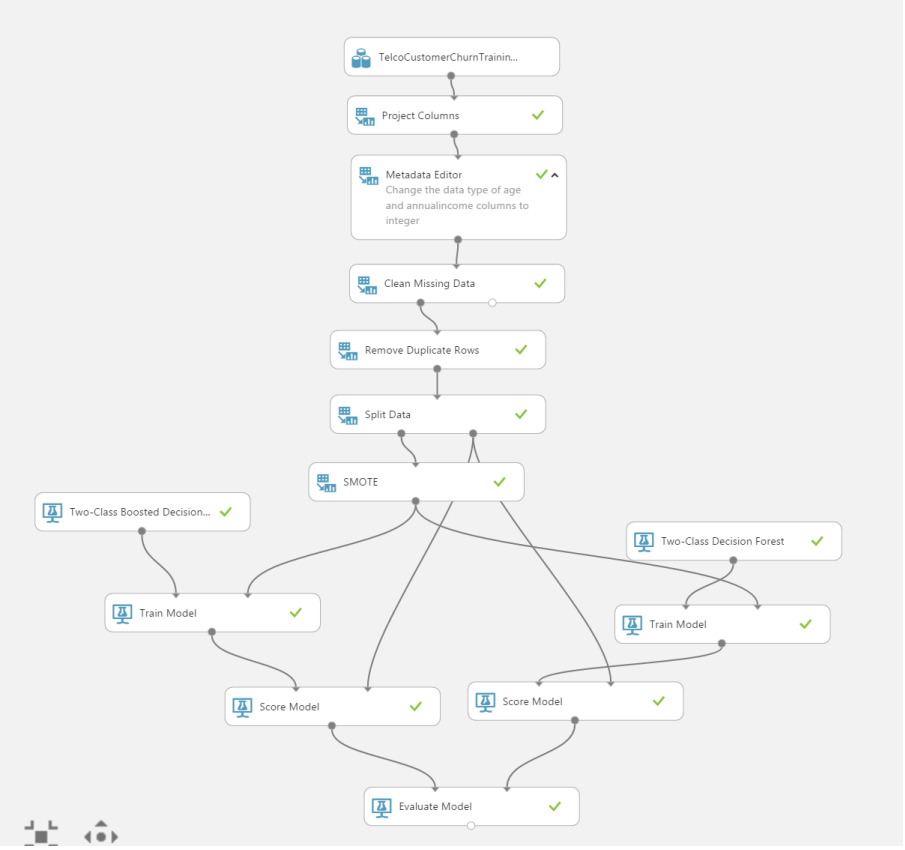


This class imbalance can affect the model's recall performance and therefore we can use the **SMOTE** module to alleviate this problem (Synthetic Minority Oversampling Technique). Drag onto the canvas the **SMOTE** module and connect the input port to the left-hand output port of the **Split Data** module. In the SMOTE properties select the **churn** column using the **Launch column selector**, set **SMOTE percentage** to **200** and the number of **nearest neighbours** to **3**. Click **Run**. Your experiment should now look as follows:



11. Set up **Train Model** and **Score Model** modules for a **Two-Class Boosted Decision Tree** and a **Two-Class Decision Forest**

(these modules can be found in Initialise model > Classification). For the **Train Model** modules ensure that the **churn** column is selected from the module properties and the data input is from the SMOTE output port (see below). Connect the **Score Model** modules from each model type to an **Evaluate Model** module (see below). Once all the connections are in-place click **Run**. Your experiment should now look as follows:



Visualizing the output of the **Evaluate Model** module shows that the Decision Forest has better recall and F1 scores than the Decision Tree.

> **Note:** we elect the Decision Forest to be our production model.

12. Click on **Publish Web Service** and select **Predictive Experiment Web Service**. Azure ML will ask you to select a model to publish, therefore select the **Train Model** Module for the **Two-Class Decision Forest**.

    In the **Predictive Experiment** we need to make a few tweaks because the Web service input will not have the churn indicator (we are trying to predict that!).

    Add a **Project Columns** module to the canvas and connect it underneath the **Remove Duplicate Rows** module (see graphic

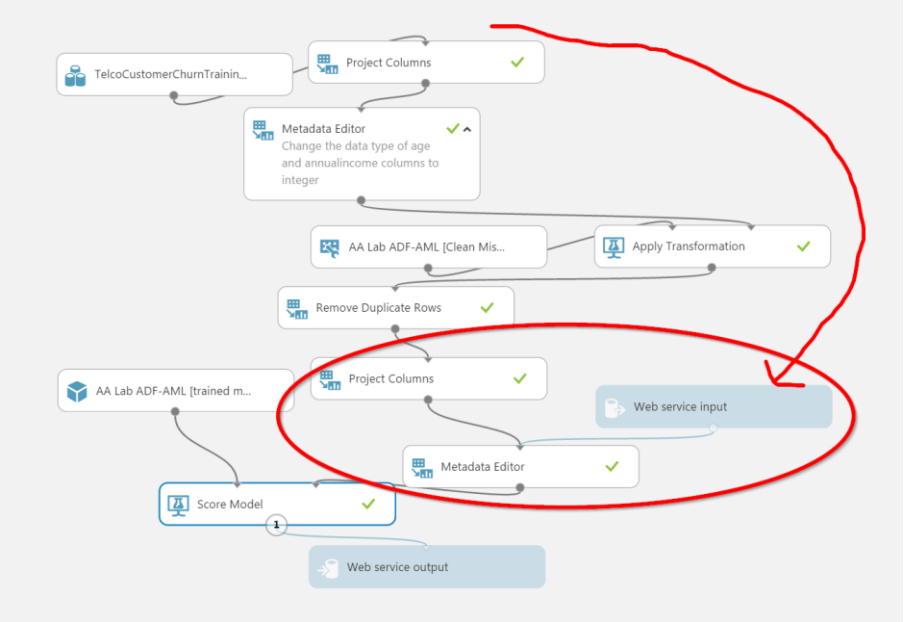
below). In the **Project Columns** properties click the **Launch Column Selector** and **Begin with including all columns** and then **exclude churn**.

Connect to the **Project Columns** module a **Metadata Editor** (see graphic below) and select **All columns** using the **Launch Column Selector**. On new column names copy-and-paste the following (**n.b. remove extra CR/LR interpreted as spaces**)

```
age,annualincome,calldroprate,callfailurerate,callingnum,customerid,customersuspended,education,gender,homeowner,maritalstatus,monthlybilledamount,noadditionallines,numberofcomplaints,numberofmonthunpaid,numdayscontractequipmentplanexpiring,occupation,penaltytoswitch,state,totalminsusedinlastmonth,unpaidbalance,usesinternetservice,usesvoiceservice,percentagecalloutsidenetwork,totalcallduration,avgcallduration
```

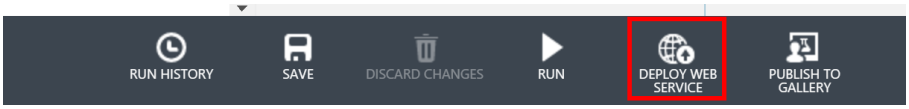
This ensures that columns names match the underlying model.

Move the **Web service input** module by selecting its connection arrow and pressing the **Delete** key – then move the **Web Service input** near to the **Metadata Editor** module just created (see below) and connect it to the **Metadata Editor** module. The Predictive Experiment should look like:



13. Click **Run**

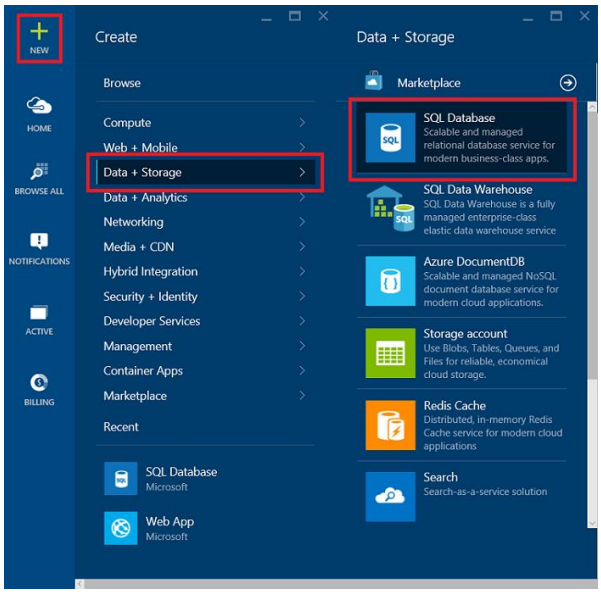
14. Once the Predictive experiment has finished running, click **DEPLOY WEB SERVICE**


RUN HISTORY
SAVE
DISCARD CHANGES
RUN
DEPLOY WEB SERVICE
PUBLISH TO GALLERY

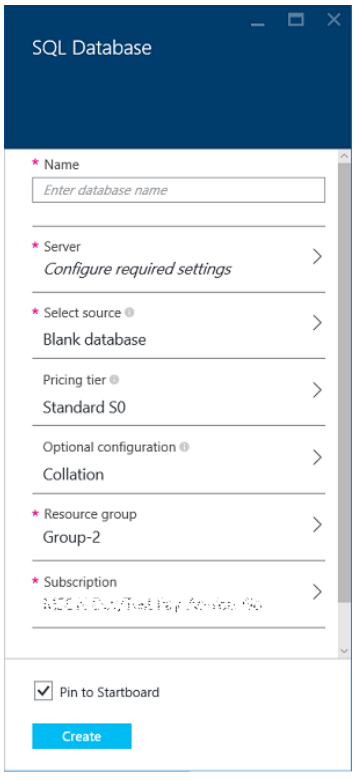
Create & Setup the Azure SQL Database

We want to store the results of our predictive engine (Azure Machine Learning) into a SQL Database hosted in Azure. This section details how to create an Azure SQL database instance.

1. Sign in to the Azure preview portal.
2. Click **New** > **Data + Storage** > **SQL Database**.



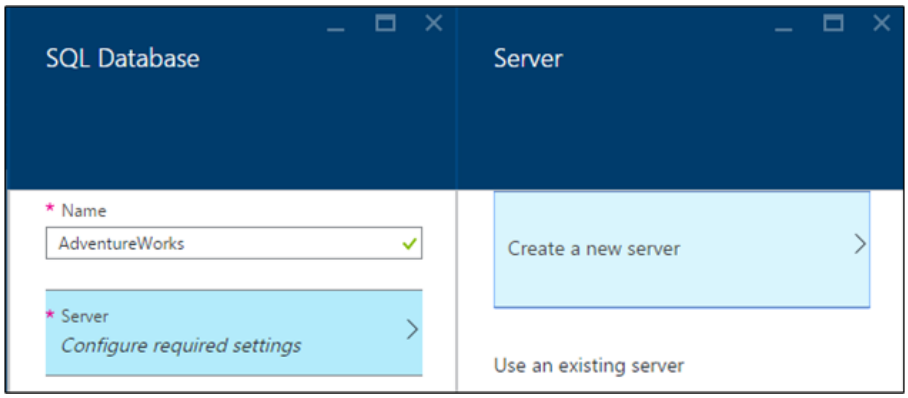
The SQL Database settings blade that appears is where you'll set up the server and database details.



A SQL database in Azure lives on a database server. A server can host multiple databases. As you set up a database, you can also create and set up the server that will host it, or you can use one that was created earlier. We'll set up a new one.

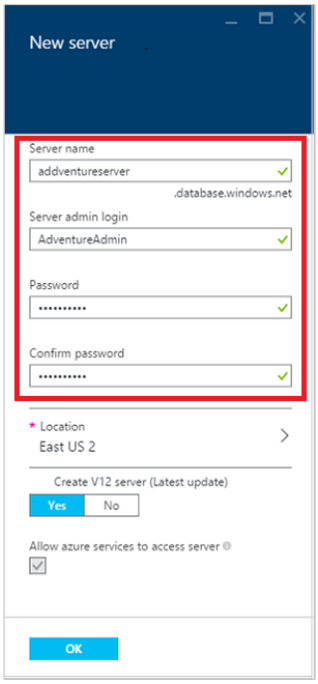
3. Type a **Name** for your database (we use **AdventureWorks**). We'll come back to cover other database settings later.

4. Under **Server** click **Configure required settings**, and then click **Create a new server**.



5. In the New server blade, type a **Server Name** that's unique throughout Azure and easy to remember. You'll need this name later when you connect and work with your database.

6. Type a **Server admin login** that's easy to remember (we use AdventureAdmin). Then type a secure **Password** and type it again in Confirm password.



Leave **Create V12 Server** (latest update) set to **Yes** to use the latest features. The Location determines the data center region where your server is created.

Leave **Allow azure services to access server** checked i.e. we want to allow Azure services like Data Factory to connect to the server.

7. Click **OK** to go back to SQL Database blade.

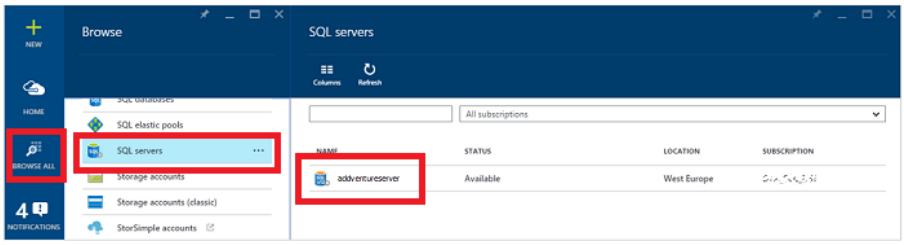
8. In the SQL Database blade, click **Select source** and then click **Blank database**.

9. You go back to the SQL Database blade, click **Create** to kick off creation of the server and database.

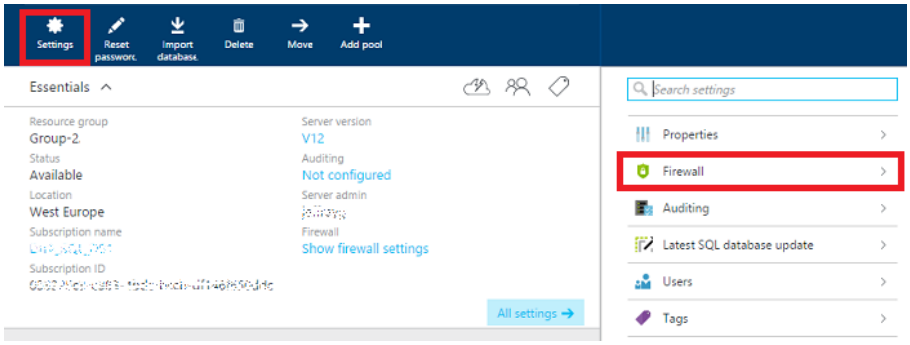
   You jump back to the Azure Startboard, where a tile shows progress until the database is created and it's online. You can also click Browse all and then click SQL Databases to confirm the database is online.

   Congratulations! You now have a database running in the cloud. You're almost finished. There's one key step left. **You need to create a rule on the database server so you can connect to the database.**

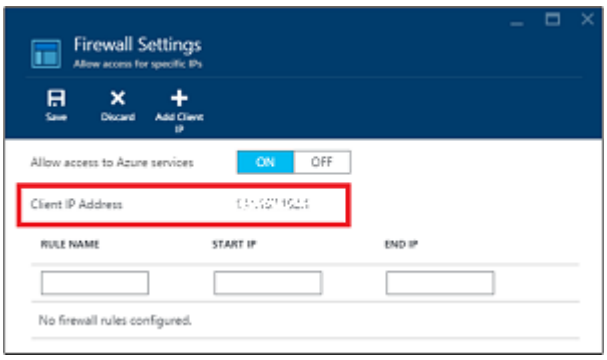
10. Click **Browse all**, scroll down and then click **SQL servers**, and then click the name of the server you created earlier from the list of SQL servers.



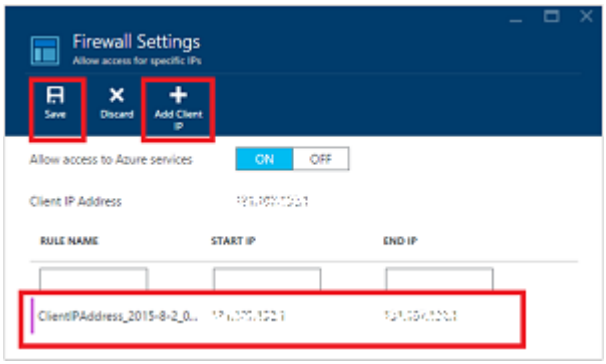
11. In the database properties blade that appears to the right, click Settings and then click Firewall from the list.



The Firewall settings show your current Client IP address.



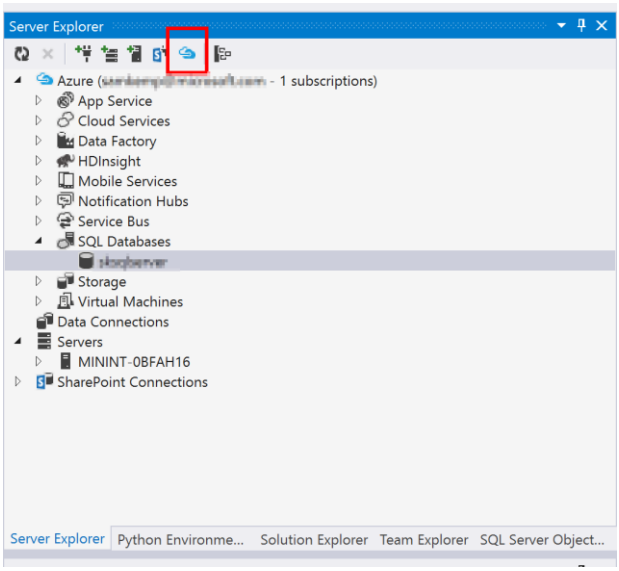
Click Add Client IP to have Azure create a rule for that IP address, and then click Save.



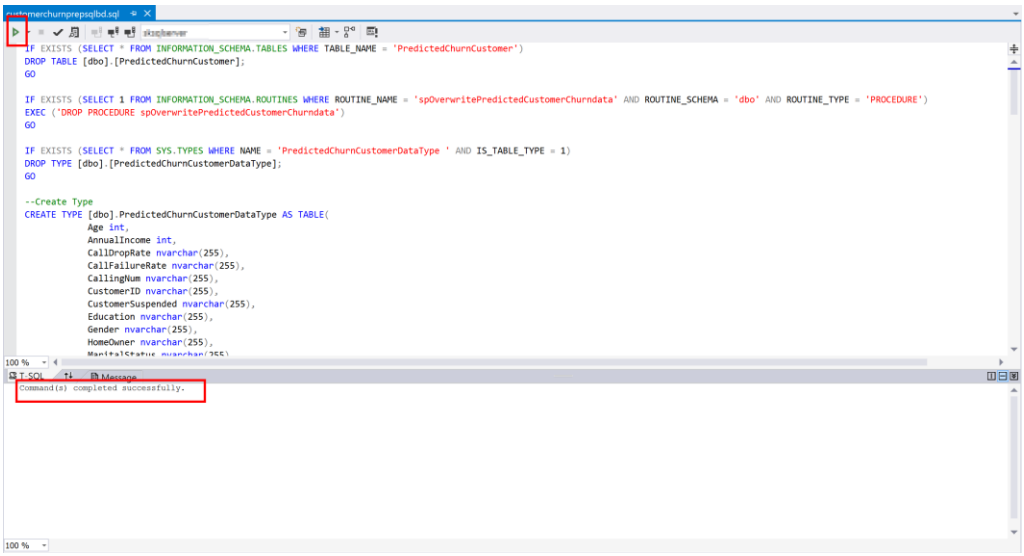
Important
Your Client IP address is likely to change from time to time, and you may not be able to access your server until you create a new firewall rule. You can check your IP address using Bing, and then add a single IP address or a range of IP addresses. See How to configure firewall settings for details.

Create SQL Table for Predictions

1. Open Visual Studio.
2. In the menu bar at the top, click **View** > **Server Explorer**
3. In **Server Explorer** connect to your **Azure subscription** > Enter your credentials.

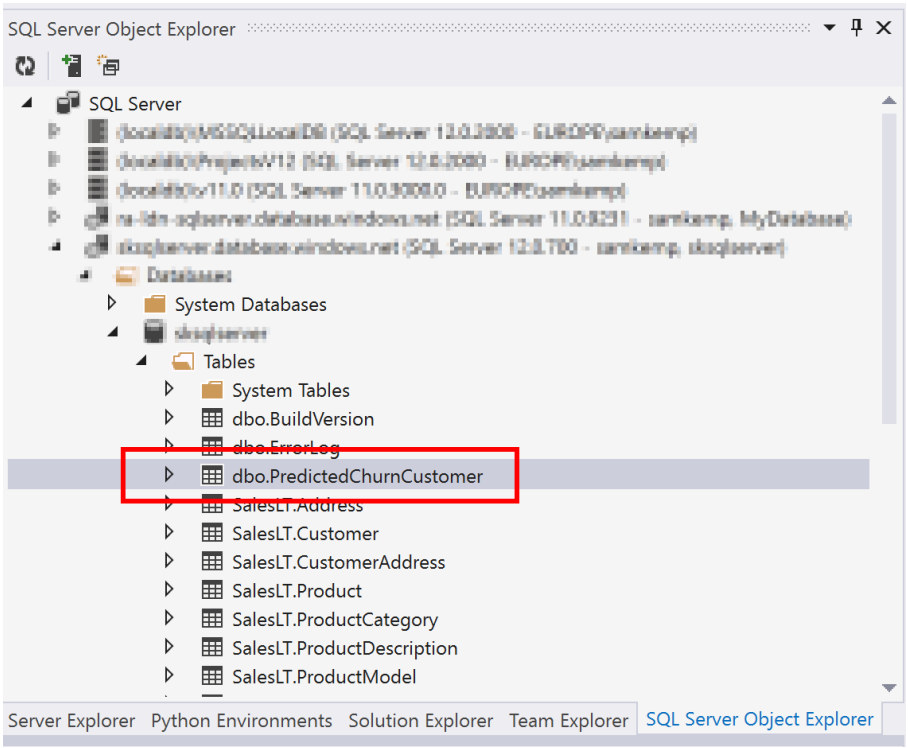


4. Right click on your Azure SQL Database and select **Open in SQL Server Object Explorer**.
5. Enter you're the SQL Server credentials you created in the prior section.
6. In Visual Studio open the **AALAB09 - ADF and AML\SQL Table Prep\customerchurnprepsqlbd.sql** file in your lab pack and click the **Green Play** button as indicated below:



This SQL script uploads a stored procedure to the SQL server. This stored procedure overwrites the table with new predicted scores.

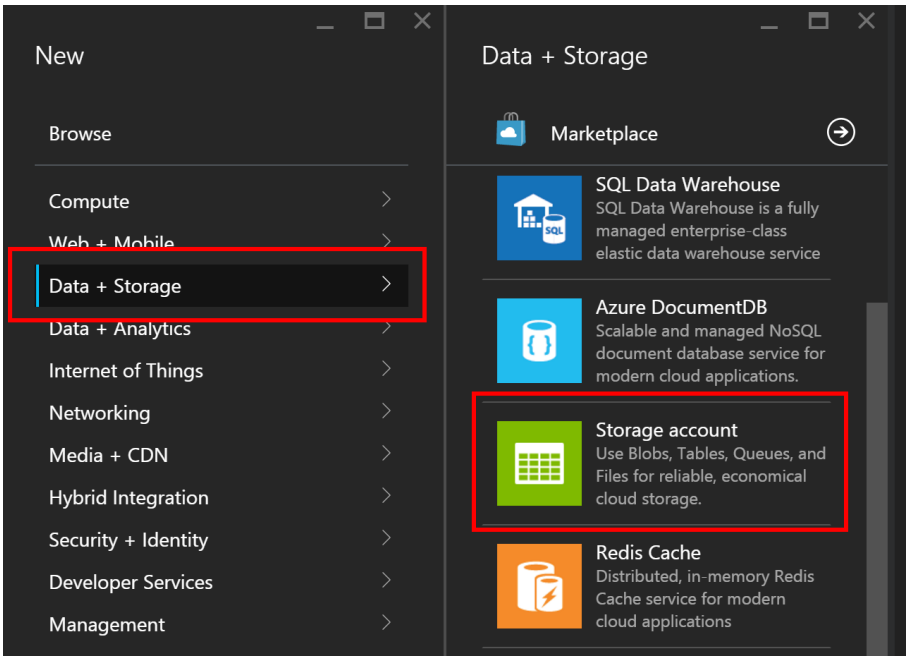
In the **SQL Server Object Explorer** in Visual Studio you should see than empty dbo.PredictedChurnCustomer table has been created:



Uploading data into blob storage

Azure Data Factory has the capability to create a **Data Gateway** to your own local file store (on-premise). Incidentally, it also supports on-premise SQL servers to allow hybrid cloud solutions. However, these topics are beyond the focus of this lab and therefore we will manually place the CDR Data and Customer information data into Azure Blob storage.

1. Create a new storage account in the Azure Portal by clicking on **New (+)** > **Data + Storage** > **Storage Account**



2. Enter in **Storage Name**, the **Pricing tier**, **Resource Group** and **location**. Click **Create**.

3. Once the storage account has been provisioned, we need to create two containers (think of this as a parent folder) in Azure Blob storage:

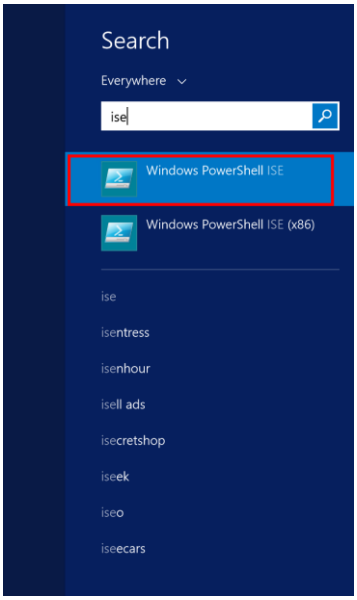
    - churndata – this stores the raw data, Hive output tables and the results from Azure Machine learning.
    - churnhql – this stores the HQL queries that we use to preprocess the data.

    To do this you can either:

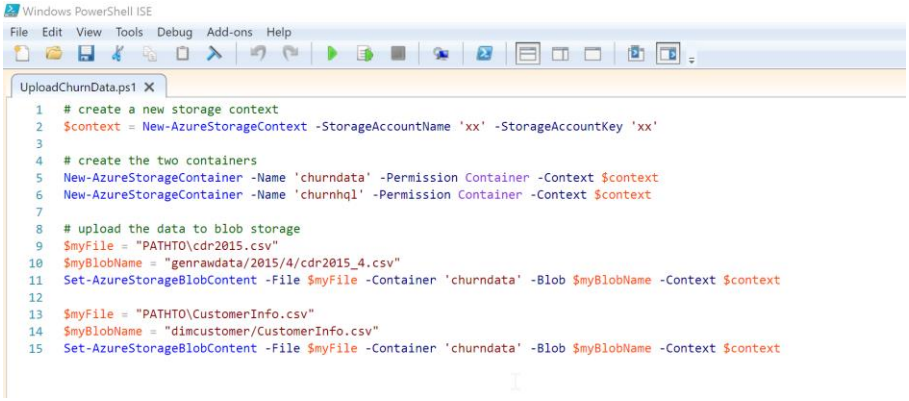
    - issue the following PowerShell commands
    - download the Azure Storage Explorer
    - use the Azure Portal

    For this lab we will use the PowerShell.

4. Open PowerShell ISE by hitting the Windows Key and searching ‘ISE’ and clicking on **Windows PowerShell ISE**



5. In Windows PowerShell ISE click **File** > **Open** and select the **Data/UploadChurnData.ps1** in this lab pack.

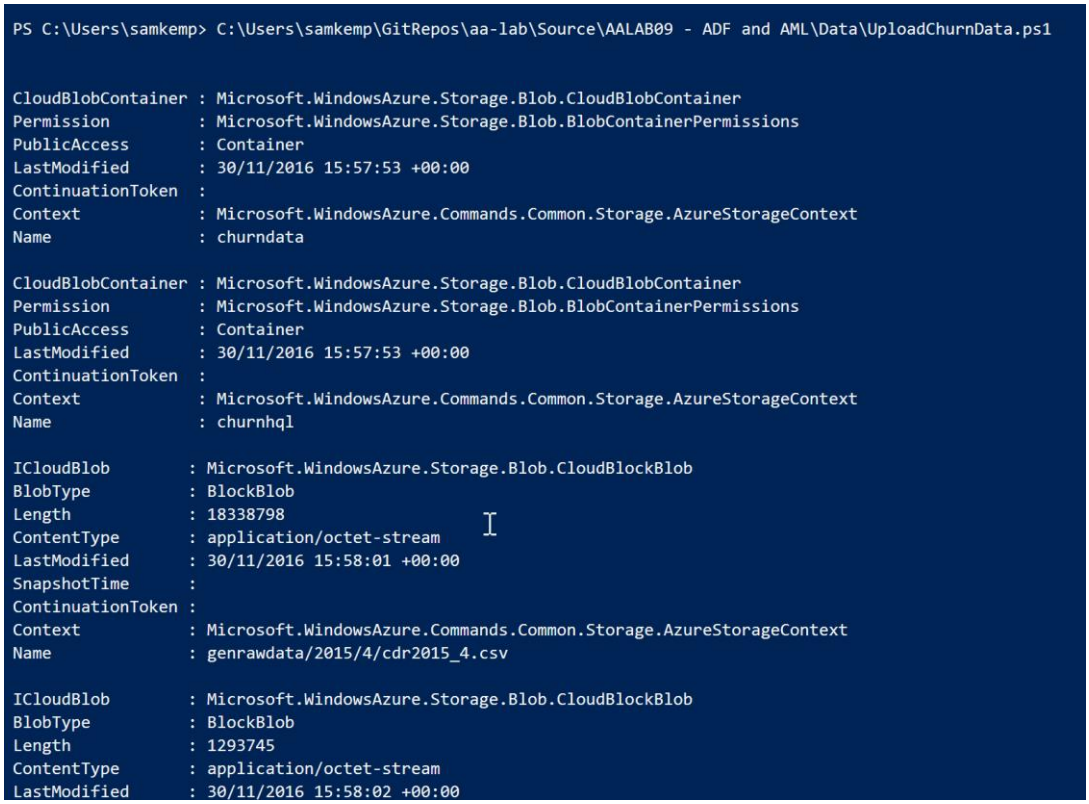


6. Updating your storage account name and key, also update the directory path to the data given out with this lab (see yellow highlights in script below).

   The storage account key can be found from the Azure Portal. By Selecting the **storage account** > **All Settings** > **Access keys** > key1.

7. Hit Save and then green play button to execute the PowerShell

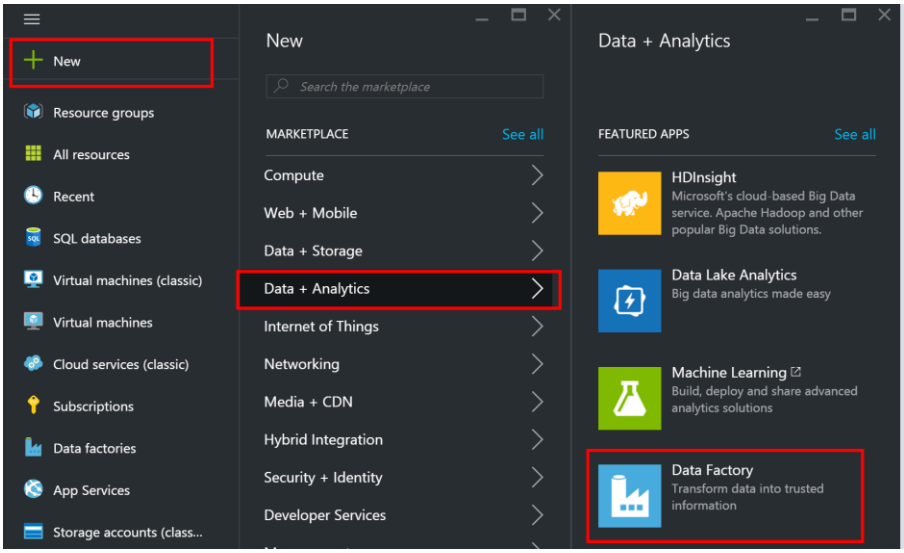
```
# create a new storage context
$context = New-AzureStorageContext -StorageAccountName 'xx' -StorageAccountKey 'xx'

# create the two containers
New-AzureStorageContainer -Name 'churndata' -Permission Container -Context $context
New-AzureStorageContainer -Name 'churnhql' -Permission Container -Context $context
# upload the data to blob storage
$myFile = "PATHTO\cdr2015.csv"
$myBlobName = "genrawdata/2015/4/cdr2015_4.csv"
Set-AzureStorageBlobContent -File $myFile -Container 'churndata' -Blob $myBlobName -Context $context

$myFile = "PATHTO\CustomerInfo.csv"
$myBlobName = "dimcustomer/CustomerInfo.csv"
Set-AzureStorageBlobContent -File $myFile -Container 'churndata' -Blob $myBlobName -Context $context
```



Output from PowerShell when files are successfully uploaded.

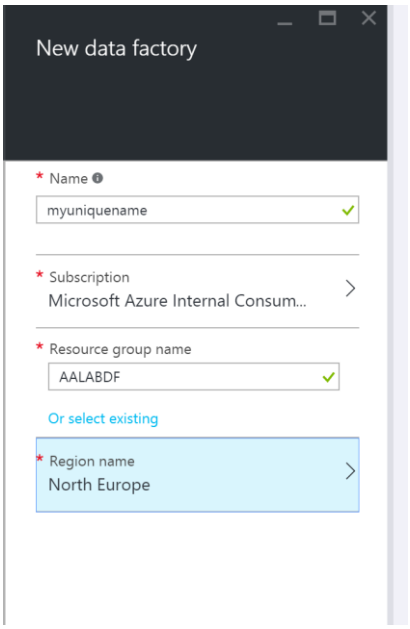
Creating the Azure Data Factory

1. Log in to the Azure Portal - https://ms.portal.azure.com/
2. Click **New** > **Data + Analytics** > **Data Factory**

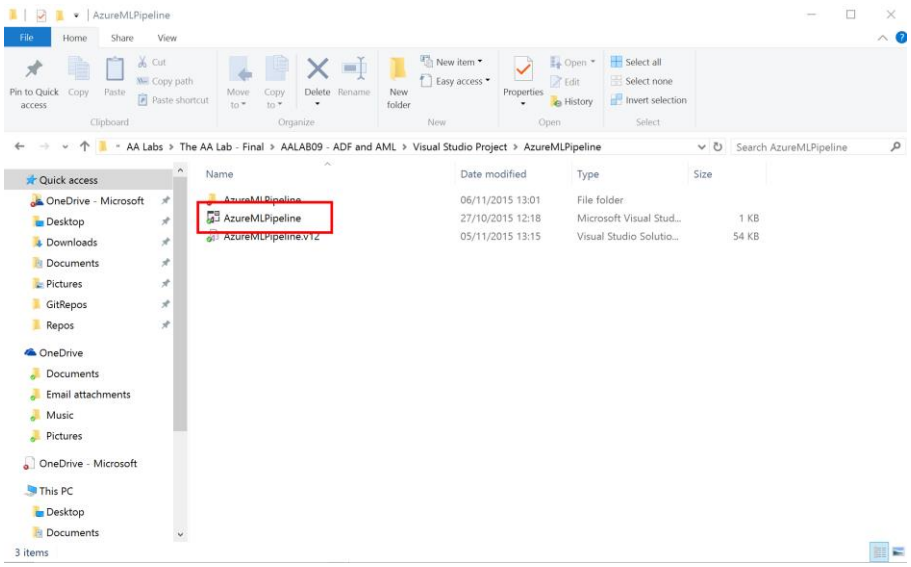


3. Give your Data Factory a **Name** (it needs to be unique to the whole of Azure), create a new **resource group** with appropriate name and select a Region. Click **Create**.

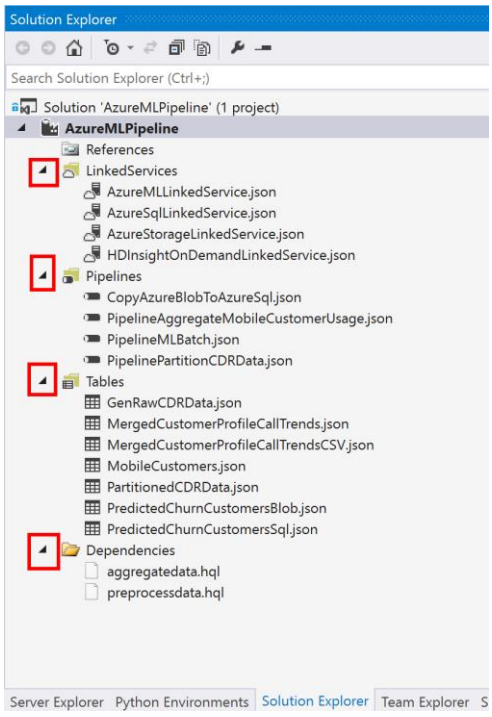
   N.B. If you are selecting an existing Resource Group, make sure it exists in the same region as the data factory.



4. Open the **AzureMLPipeline** Visual Studio project contained in the **AALAB09 – ADF and AML** directory



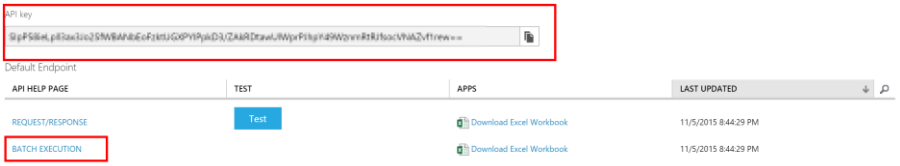
5. In the Solution Explorer expand LinkedServices, Pipelines, Tables and Dependencies.



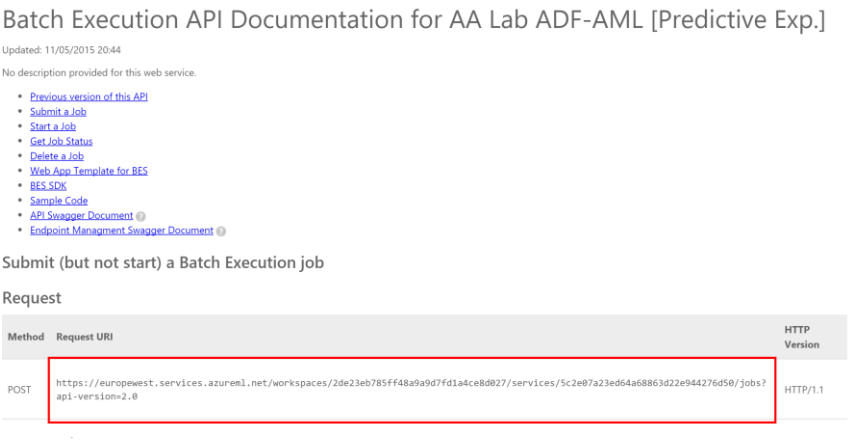
6. Double click on the **AzureMLLinkedService.json** file and you should see the following:

```
{
    "$schema": "http://datafactories.schema.management.azure.com/schemas/2015-08-01/Microsoft.DataFactory.LinkedService.json",
    "name": "AzureMLLinkedService",
    "properties": {
        "type": "AzureML",
        "typeProperties": {
            "mlEndpoint": "ENTER THE ENDPOINT",
            "apiKey": "ENTER THE API KEY"
        }
    }
}
```

Log in to Azure ML and click on **Web Services** > click on the **Web Service** created earlier in the lab > Copy the **API Key** and paste into the **apiKey** property in the JSON file. Then click on **Batch Execution**.



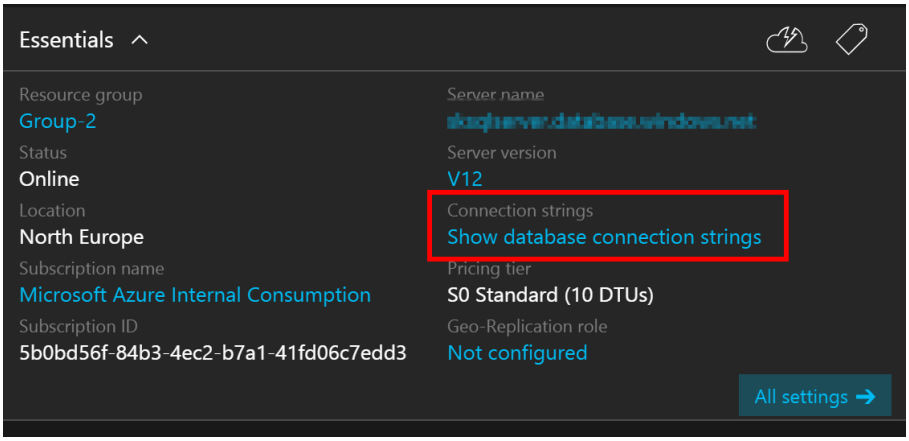
Clicking on Batch Execution takes you to the API documentation where you will find the URI



Copy-and-paste the URI into the **mlEndpoint** property in the JSON file.

7. Next, in Visual Studio double click on the **AzureSqlLinkedService.json** file

Update the connectionString property with that from your Azure SQL Database. The connection string can be found on the **Azure portal** by clicking on SQL Databases > Select your database > **Show database connection strings**:



From the list of database connection strings select **ADO.NET** to copy-and paste into the **connectionString** property of the JSON file. **Update your user ID and password in the connection string**.

8. In Visual Studio now open the **AzureStorageLinkedService.json** file. Update the **AccountName** and **AccountKey**.

9. In the file **PipelineAggregateMobileCustomerUsage.json** you will need to update the table locations for Hive. Replace **ACCOUNTNAME** with your storage account name.

```
"type": "HDInsightHive",
"typeProperties": {
    "scriptPath": "churnhql/aggregatedata.hql",
    "scriptLinkedService": "AzureStorageLinkedService",
    "storageLinkedServices": [
        "AzureStorageLinkedService"
    ],
    "defines": {
        "CallsPartitioned": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/partitioneddata",
        "AggregatedData": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/aggregateddata",
        "CustomerData": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/dimcustomer",
        "Customer360": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/mlinput",
        "Year": "$$Text.Format('{0:yyyy}',SliceStart)",
        "Month": "$$Text.Format('{0:%M}',SliceStart)",
        "Day": "$$Text.Format('{0:%d}',SliceStart)"
    }
```

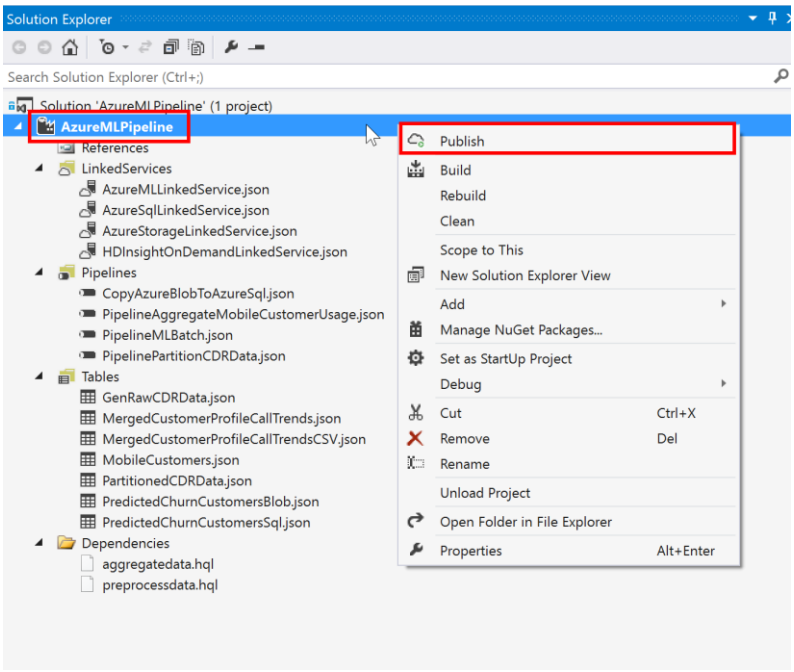
10. You need to repeat this in the **PipelinePartitionCDRData.json** file i.e. replace **ACCOUNTNAME** for your storage account name.

```
"type": "HDInsightHive",
"typeProperties": {
    "scriptPath": "churnhql/preprocessdata.hql",
    "scriptLinkedService": "AzureStorageLinkedService",
    "storageLinkedServices": [
        "AzureStorageLinkedService"
    ],
    "defines": {
        "RawData": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/rawdata",
        "GenRawData": "$$Text.Format('wasb://churndata@ACCOUNTNAME.blob.core.windows.net/genrawdata/{0:yyyy}
        "CallsPartitioned": "wasb://churndata@ACCOUNTNAME.blob.core.windows.net/partitioneddata",
        "AggregatedData": "$$Text.Format('wasb://churndata@ACCOUNTNAME.blob.core.windows.net/aggregateddata/
        "Year": "$$Text.Format('{0:yyyy}',SliceStart)",
        "Month": "$$Text.Format('{0:%M}',SliceStart)",
        "Day": "$$Text.Format('{0:%d}',SliceStart)"
    }
},
```

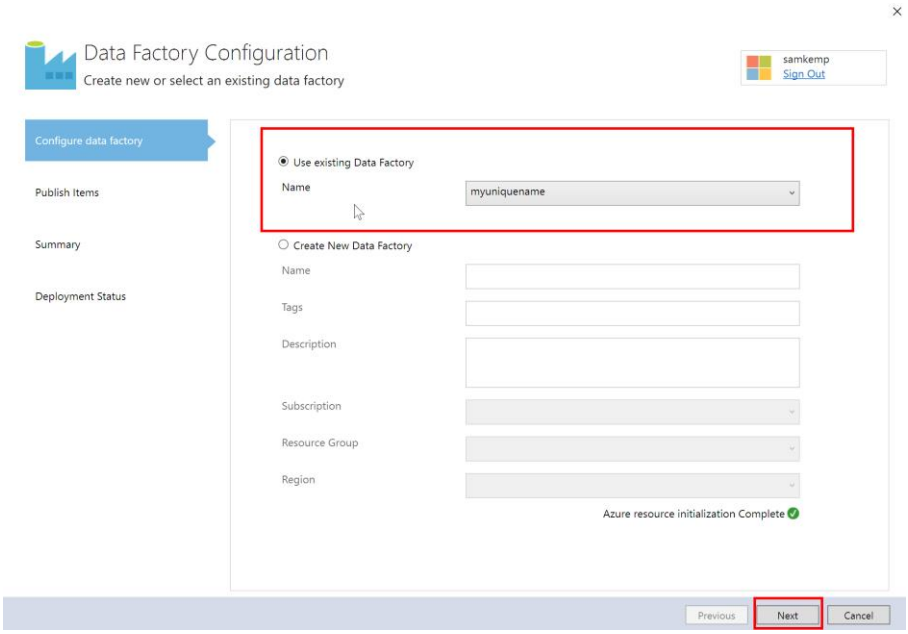
11. You are now ready to validate your JSON files. Click **Build** > **Rebuild Solution**.

    Once validated that the build is successful. You can now Publish to Azure.

12. In the Solution Explorer, Right click on the **AzureMLPipeline** Data Factory > click **Publish**.



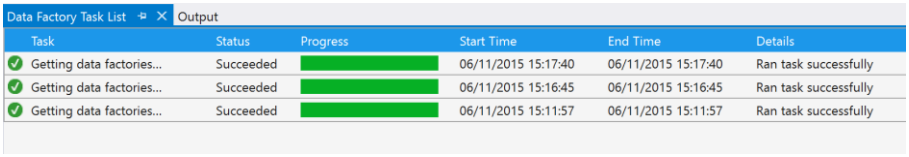
13. In the Data Factory Configuration window, select **Use existing Data Factory** and select the data factory you created earlier. Click **Next**.



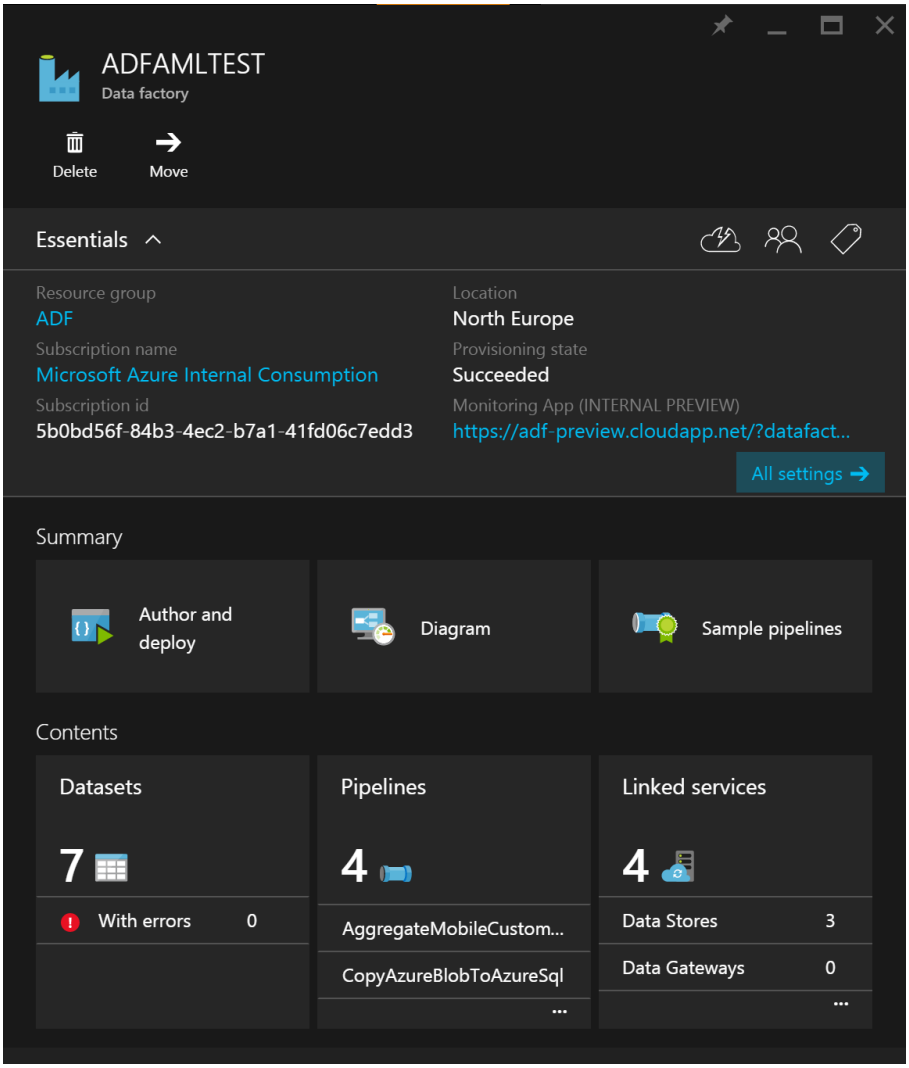
14. On the Publish Items Tab > Click Next.

15. On the Summary Tab > Click Next.

16. The Data Factory will start deploying to Azure, once it has finished you will be able to click the **Finished** button.



17. In the Azure Portal > click on you Data Factory and you should have the following Blade:

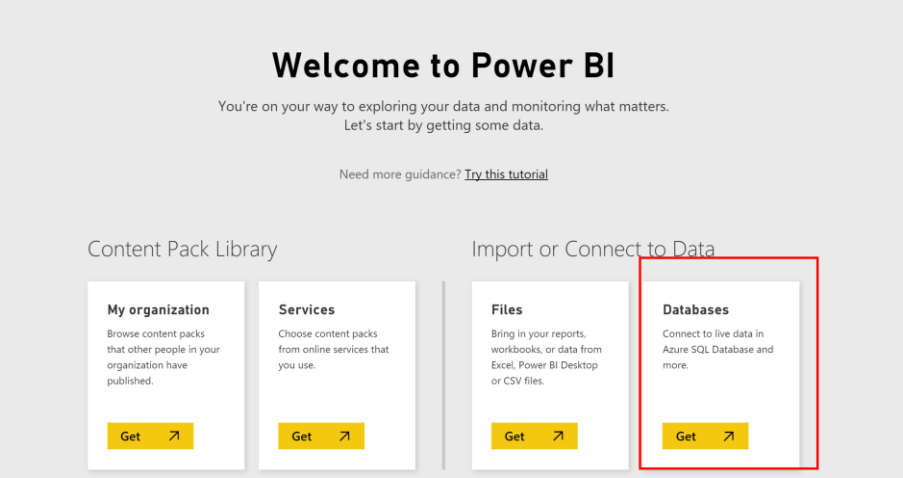


18. Click through the datasets and you should see that the Data Factory is in progress.

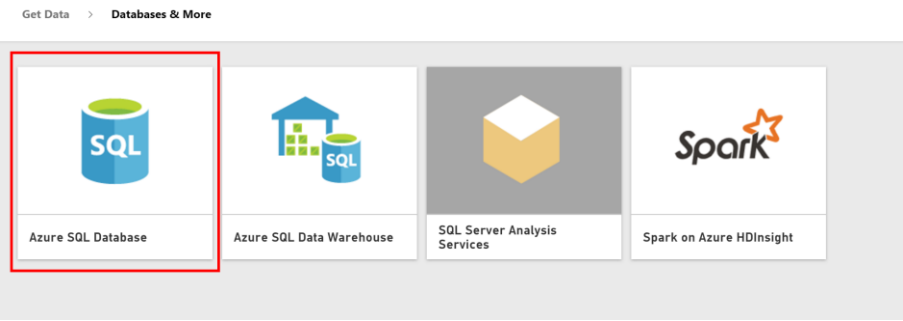
NOTE: It will take 20mins for the factory to finish because it is provisioning an OnDemand (temporary) Hadoop cluster.

Connecting to Power BI

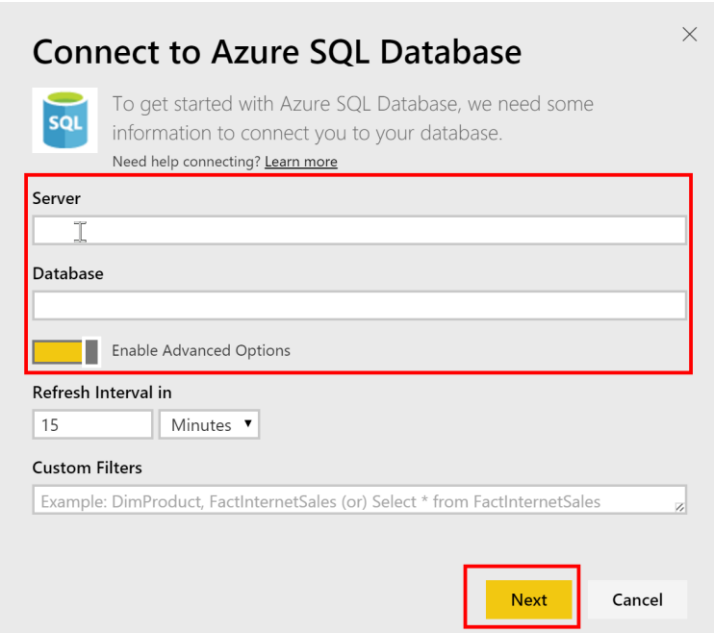
1. Log in to the Power BI Portal.
2. On the Welcome page click on **Databases**



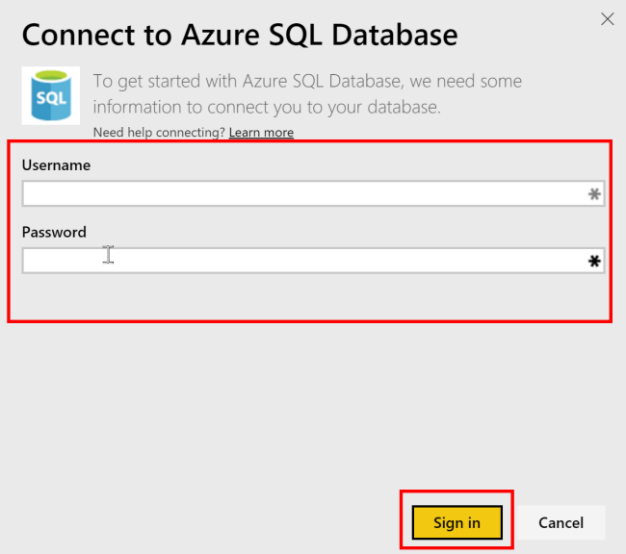
3. Click on **Azure SQL Database** > **Connect**.



4. Enter **Server** and **Database** names. Also slide the **Enable Advanced Options** to the right. Click **Next**.



5. Enter **Username** and **Password** credentials for your Server and click **Sign In**.



6. You will now see the **PredictedChurnCustomer** table and you can freely explore the data.

Terms of Use

By using this Hands-on Lab, you agree to the following terms:

The technology/functionality described in this Hands-on Lab is provided by Microsoft Corporation in a "sandbox" testing environment for purposes of obtaining your feedback and to provide you with a learning experience. You may only use the Hands-on Lab to evaluate such technology features and functionality and provide feedback to Microsoft. You may not use it for any other purpose. You may not modify copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, or sell this Hands-on Lab or any portion thereof.

COPYING OR REPRODUCTION OF THE HANDS-ON LAB (OR ANY PORTION OF IT) TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

THIS HANDS-ON LAB PROVIDES CERTAIN SOFTWARE TECHNOLOGY/PRODUCT FEATURES AND FUNCTIONALITY, INCLUDING POTENTIAL NEW FEATURES AND CONCEPTS, IN A SIMULATED ENVIRONMENT WITHOUT COMPLEX SET-UP OR INSTALLATION FOR THE PURPOSE DESCRIBED ABOVE. THE TECHNOLOGY/CONCEPTS REPRESENTED IN THIS HANDS-ON LAB MAY NOT REPRESENT FULL FEATURE FUNCTIONALITY AND MAY NOT WORK THE WAY A FINAL VERSION MAY WORK. WE ALSO MAY NOT RELEASE A FINAL VERSION OF SUCH FEATURES OR CONCEPTS. YOUR EXPERIENCE WITH USING SUCH FEATURES AND FUNCITONALITY IN A PHYSICAL ENVIRONMENT MAY ALSO BE DIFFERENT.

FEEDBACK. If you give feedback about the technology features, functionality and/or concepts described in this Hands-on Lab to Microsoft, you give to Microsoft, without charge, the right to use, share and commercialize your feedback in any way and for any purpose. You also give to third parties, without charge, any patent rights needed for their products, technologies and services to use or interface with any specific parts of a Microsoft software or service that includes the feedback. You will not give feedback that is subject to a license that requires Microsoft to license its software or documentation to third parties because we include your feedback in them. These rights survive this agreement.

MICROSOFT CORPORATION HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH REGARD TO THE HANDS-ON LAB, INCLUDING ALL WARRANTIES AND CONDITIONS OF MERCHANTABILITY, WHETHER EXPRESS, IMPLIED OR STATUTORY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. MICROSOFT DOES NOT MAKE ANY ASSURANCES OR REPRESENTATIONS WITH REGARD TO THE ACCURACY OF THE RESULTS, OUTPUT THAT DERIVES FROM USE OF THE VIRTUAL LAB, OR SUITABILITY OF THE INFORMATION CONTAINED IN THE VIRTUAL LAB FOR ANY PURPOSE.

DISCLAIMER

This lab contains only a portion of the features and enhancements in Microsoft Azure. Some of the features might change in future releases of the product.