Microsoft Cloud Workshop

Big data and visualization

Hands-on lab step-by-step

October 2017

Contents

Big data and visualization hands-on lab step-by-step

Abstract and learning objectives

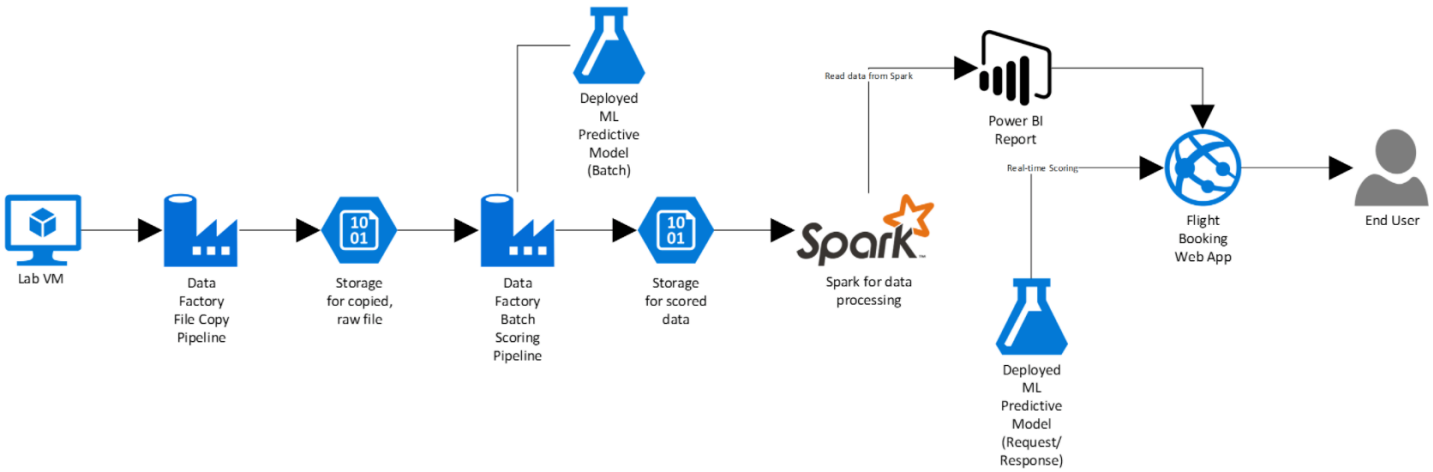
In this workshop, you will complete a web app using Machine Learning to predict travel delays given flight delay data and weather conditions, plan the bulk data import operation, followed by preparation tasks, such as cleaning and manipulating the data for testing, and training your Machine Learning model.

By attending this workshop, you will be better able to build a complete Azure Machine Learning (ML) model for predicting if an upcoming flight will experience delays. In addition, you will learn to:

- Integrate the Azure ML web service in a Web App for both one at a time and batch predictions
- Analyze batch data with SQL Data Warehouse
- Visualize batch predictions on a map using Power BI

This hands-on lab is designed to provide exposure to many of Microsoft's transformative line of business applications built using Microsoft big data and advanced analytics. The goal is to show an end-to-end solution, leveraging many of these technologies, but not necessarily doing work in every component possible. The lab architecture is below and includes:

- Azure Machine Learning (Azure ML)
- Azure Data Factory (ADF)
- Azure Storage
- HDInsight Spark
- Power BI Desktop
- Azure App Service



Overview

AdventureWorks Travel (AWT) provides concierge services for business travelers. In an increasingly crowded market, they are always looking for ways to differentiate themselves, and provide added value to their corporate customers.

They are looking to pilot a web app that their internal customer service agents can use to provide additional information useful to the traveler during the flight booking process. They want to enable their agents to enter in the flight information and produce a prediction as to whether the departing flight will encounter a 15-minute or longer delay, taking into account the weather forecasted for the departure hour.

In this hands-on lab, attendees will build an end-to-end solution to predict flight delays, accounting for the weather forecast.

Requirements

1. Microsoft Azure subscription must be pay-as-you-go or MSDN
    a. Trial subscriptions will not work

Before the hands-on lab

Duration: 45 mins

In this exercise, you will set up your environment for use in the rest of the hands-on lab. You should follow all the steps provided in the Before the Hands-on Lab section to prepare your environment *before* attending the hackathon.

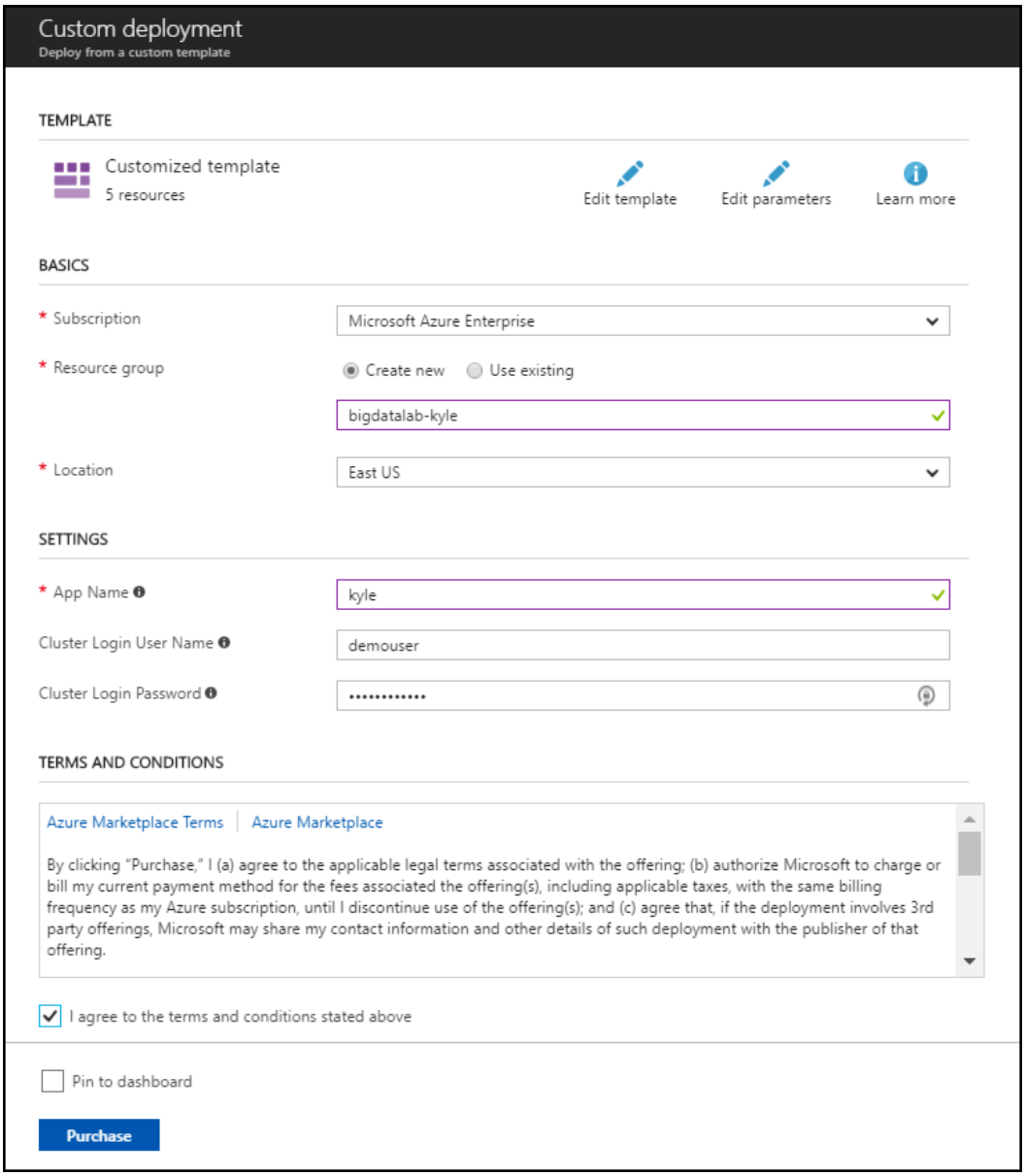
Task 1: Deploy HDInsight cluster, Azure ML, and storage accounts to Azure

1. CTRL+Click the **Deploy to Azure** button below, and you will be taken to the Azure portal, and presented with a form for a new custom deployment (which uses an Azure Resource Management (ARM) template from a GitHub repository). You will be presented with a blade to provide some custom parameters as show in the screenshot below.

2. In the Custom deployment blade that appears, enter the following values:
   a. Subscription: Select your subscription
   b. Resource group: Use and existing Resource group, or create a new one by entering a unique name, such as "bigdatalab-[your initials or first name]".
   c. Location: Select a location for the Resource group. Recommend using East US, East US 2, West Central US, or West US 2, as some resources, such as Data Factory, are only available in those regions.
   d. App name: Enter a unique name, such as your initials or first name. This value must be between 3 and 10 characters long and should not contain any special characters. Note the name, as you will need to use it in your Lab VM deployment in Task 3 as well.
   e. Cluster Login User Name: Enter a name, or accept the default. Note all references to this in the lab use the default user name, demouser, so if you change it, please note it for future reference throughout the lab.
   f. Cluster Login Password: Enter a password, or accept the default. Note all references to this in the lab use the default password, **Password.1!!**, so if you change it, please note it for future reference throughout the lab.
   g. Check the box to agree to the terms.

h. Select **Purchase**.



3. The deployment will take about 15 minutes to complete.
4. Wait for the deployment to complete before attempting to deploy the Lab Virtual Machine in Task 3, as it depends on the Virtual Network created by this deployment. In the meantime, you can move on to the next task, Task 2, while this deployment is ongoing.
5. See Appendix A for detailed steps on creating these components without using an ARM template.

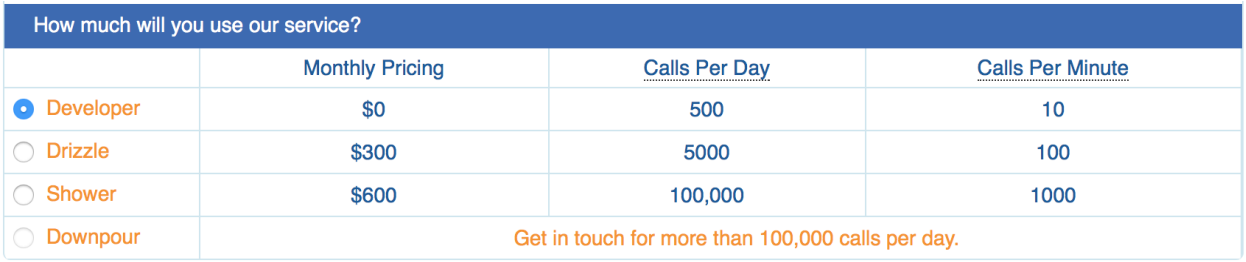
Task 2: Register for a trial API account at WeatherUnderground.com

To retrieve the 10-day hourly weather forecast, you will use an API from WeatherUnderground.com. There is a free developer version that provides you access to the API you need for this hands-on lab.

1. Navigate to http://www.wunderground.com/weather/api/.
2. Select **Explore My Options**.

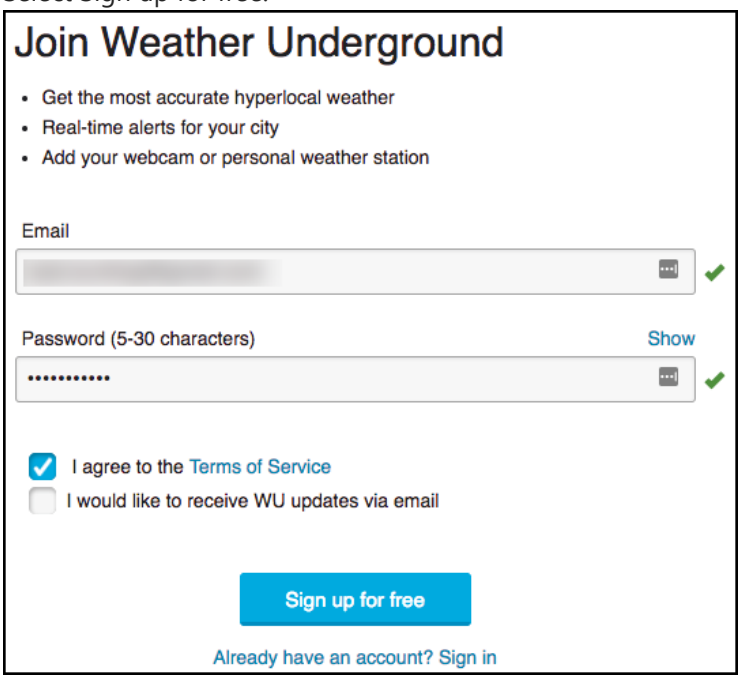
3. On the Get Your API Key page, select **Anvil Plan**.

4. Scroll down until you see the area titled How much will you use our service? Ensure **Developer** is selected.

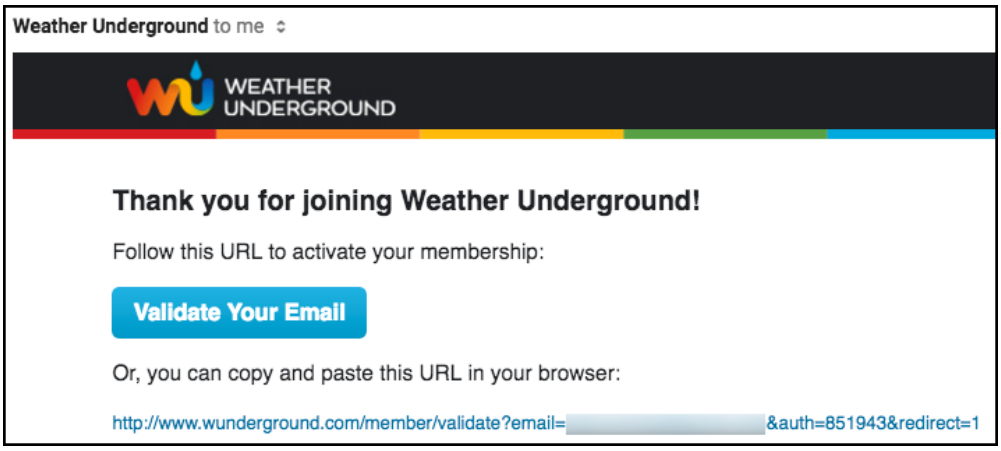


5. Select **Purchase Key**.

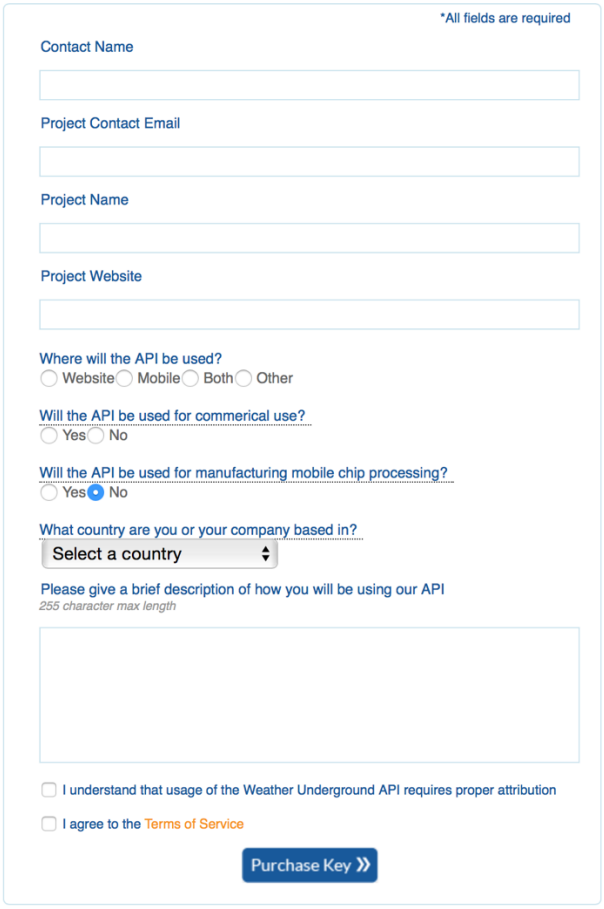
6. Complete the Create an Account form by providing your email address and a password, and agreeing to the terms.
7. Select Sign up for free.



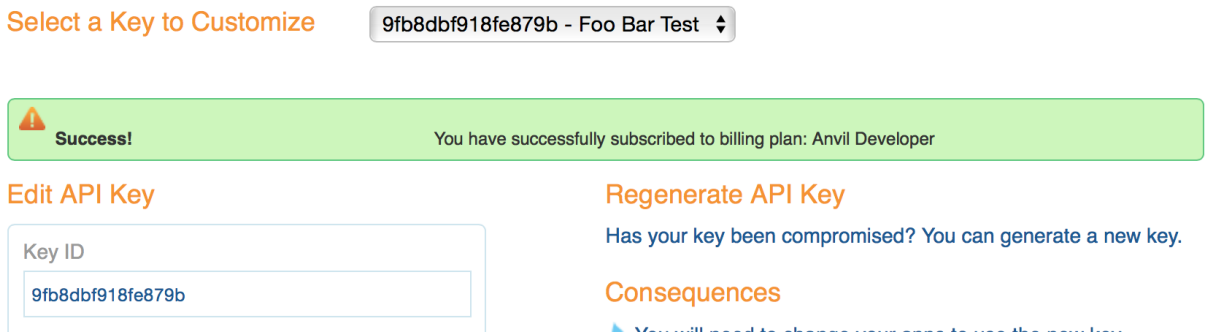
8. In a few moments you should receive a confirmation email at the email address you provided. Select the Validate Your Email link found within the email.



9. Once you have validated your email, go back to the Get Your API Key page, re-select Anvil and select Purchase Key.
10. Complete the brief contact form. When answering where will the API be used, select **Website**. For Will the API be used for commercial use, select **No**. Select **Purchase Key**.



11. You should be taken to a page that displays your key, similar to the following:

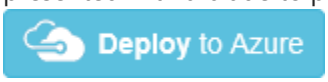


12. Take note of your API Key. It is available from the text box labeled **Key ID**.
13. To verify that your API Key is working, modify the following URL to include your API Key: http://api.wunderground.com/api/<YOURAPIKEY>/hourly10day/q/SEATAC.json.
14. Open your modified link in a browser, you should get a JSON result showing the 10-day, hourly weather forecast for the Seattle-Tacoma International Airport.


```
{
  "response": {
    "version": "0.1",
    "termsofService": "http://www.wunderground.com/weather/api/d/terms.html",
    "features": {
      "hourly10day": 1
    }
  },
  "hourly_forecast": [
    {
      "FCTTIME": {
        "hour": "15",
        "hour_padded": "15",
        "min": "00",
        "min_unpadded": "0",
        "sec": "0",
        "year": "2016",
        "mon": "12",
        "mon_padded": "12",
        "mon_abbrev": "Dec",
```

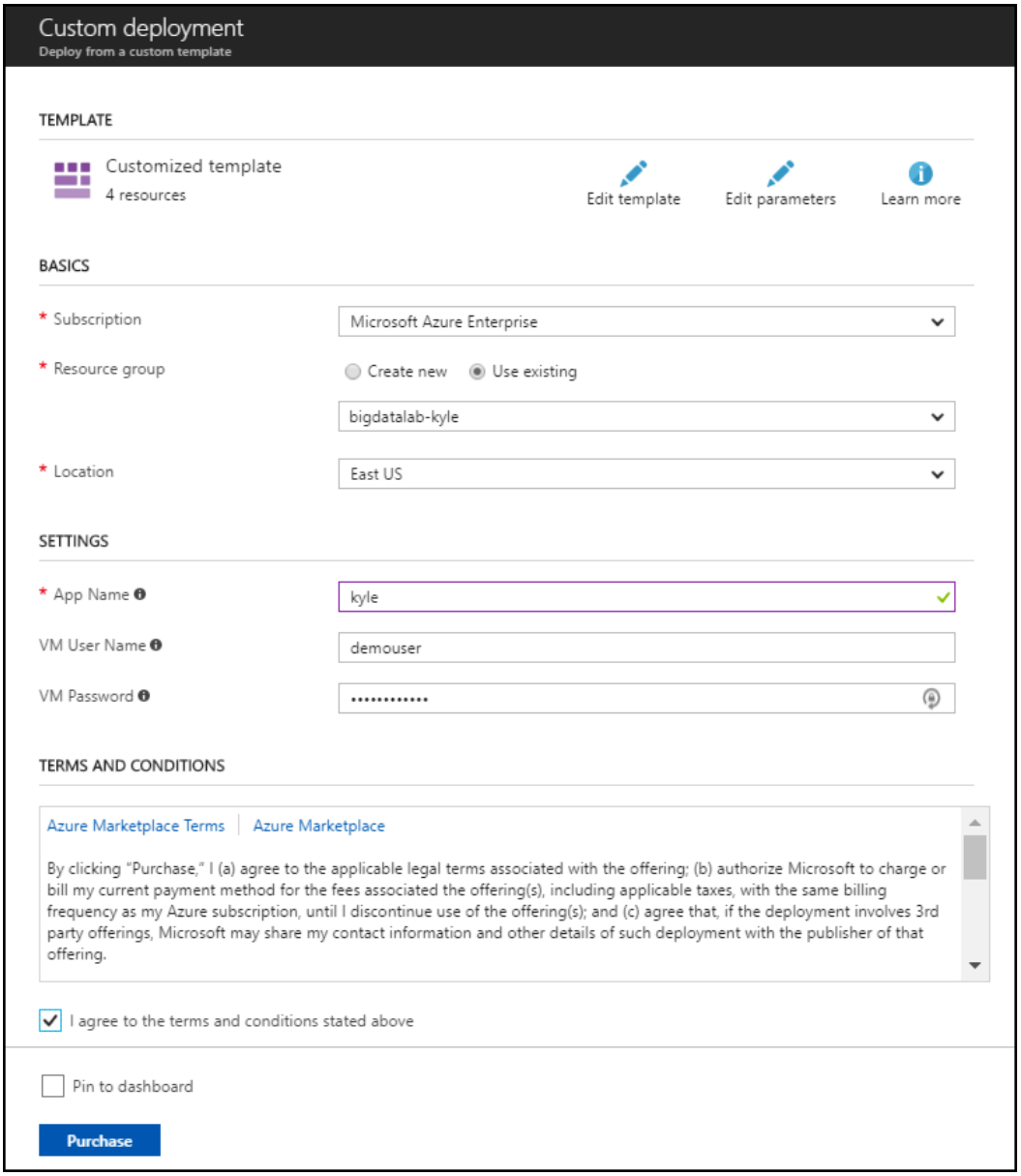

Task 3: Deploy lab virtual machine (lab VM) to Azure

1. CTRL+Click the **Deploy to Azure** button below, and you will be taken to the Azure portal, and presented with a form for a new custom deployment (which uses an ARM template from a GitHub repository). You will be presented with a blade to provide some custom parameters as show in the screenshot below.



2. In the Custom deployment blade that appears, enter the following values:
    a. Subscription: Select your subscription
    b. Resource group: Choose Use Existing, and select the same resource group you used when deploying your HDInsight cluster and Azure ML workspace, above.
    c. Location: The location should be automatically selected to be the same as your Resource Group.
    d. App name: IMPORTANT: You must enter the same App name you used in the deployment above in Task 1.
    e. VM User Name: Enter a name, or accept the default. Note all references to this in the lab use the default user name, demouser, so if you change it, please note it for future reference throughout the lab.

f. VM Password: Enter a password, or accept the default. Note all references to this in the lab use the default password, Password.1!!, so if you change it, please not it for future reference throughout the lab.
g. Check the box to agree to the terms.
h. Select **Purchase**.

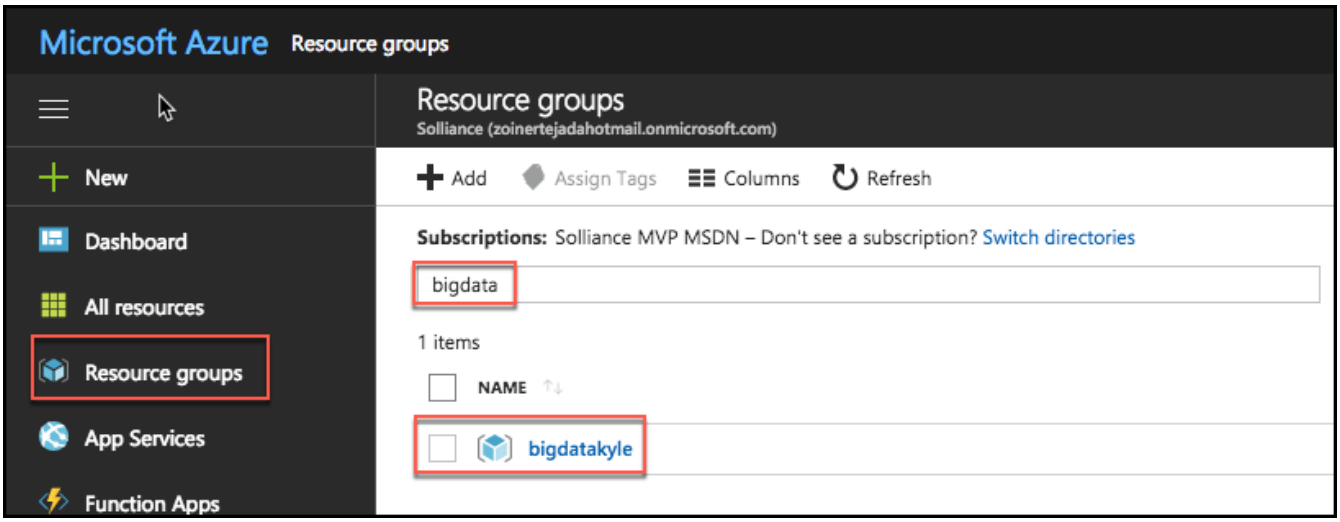


3. The deployment will take about 10 minutes to complete.
4. See Appendix B for detailed steps on creating the lab virtual machine without using an ARM template.

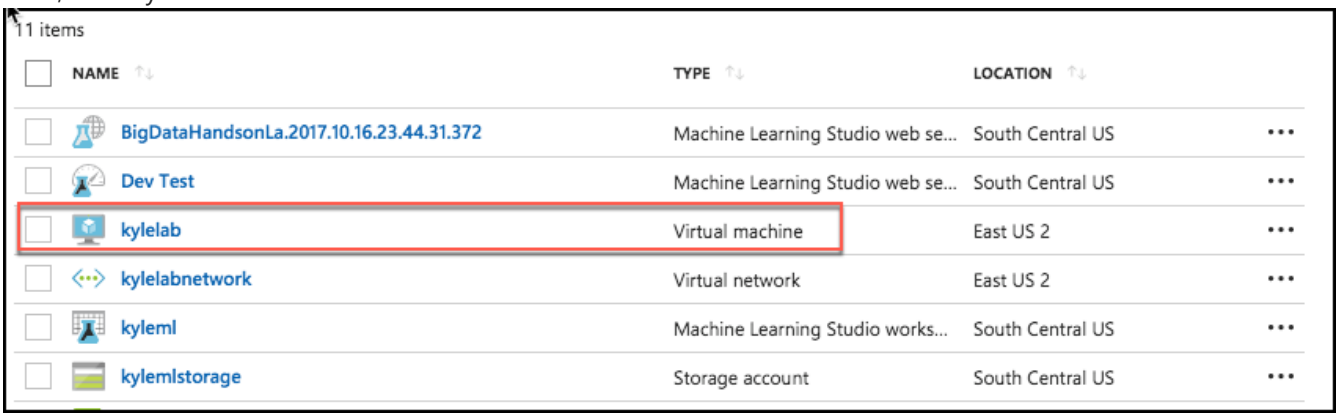
Task 4: Install Power BI Desktop on the lab VM

1. Connect to the Lab VM. (If you are already, connected to your Lab VM, skip to Step 7.

2. From the left side menu in the Azure portal, click on **Resource groups**, then enter your resource group name into the filter box, and select it from the list.



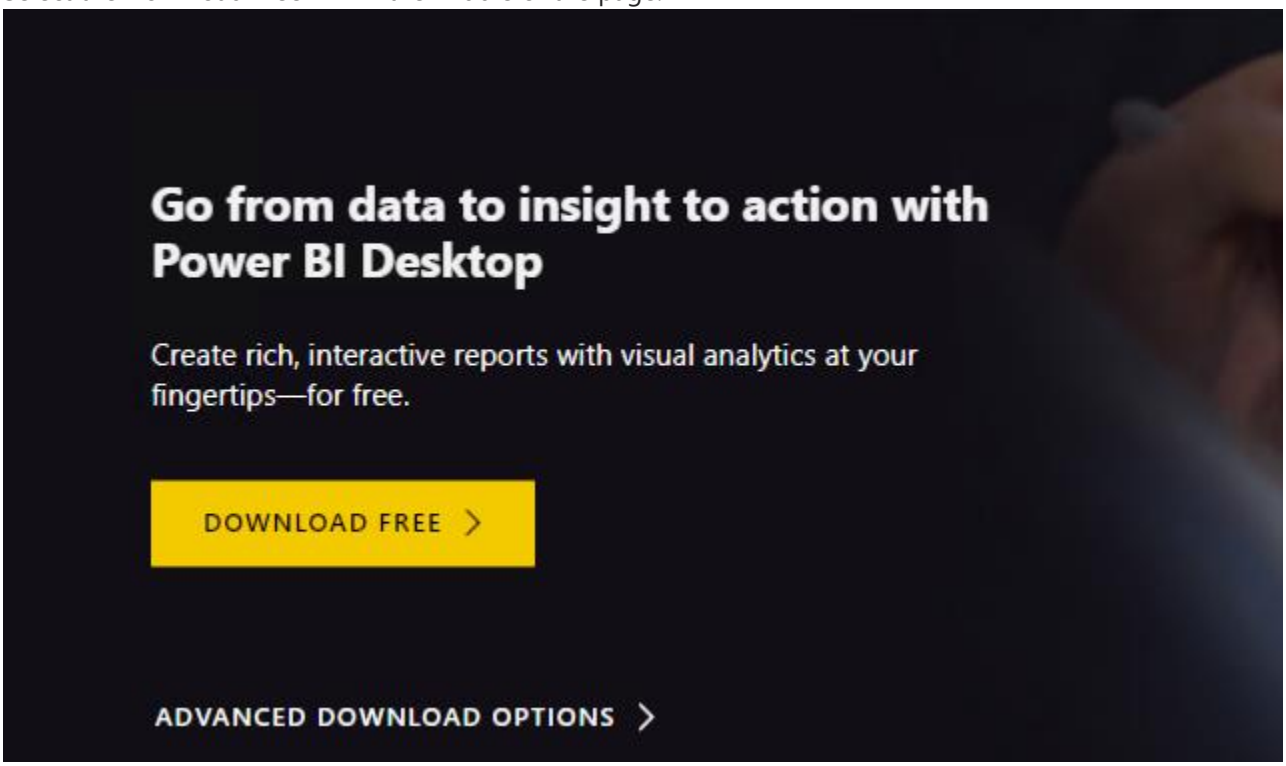
3. Next, select your lab virtual machine from the list.



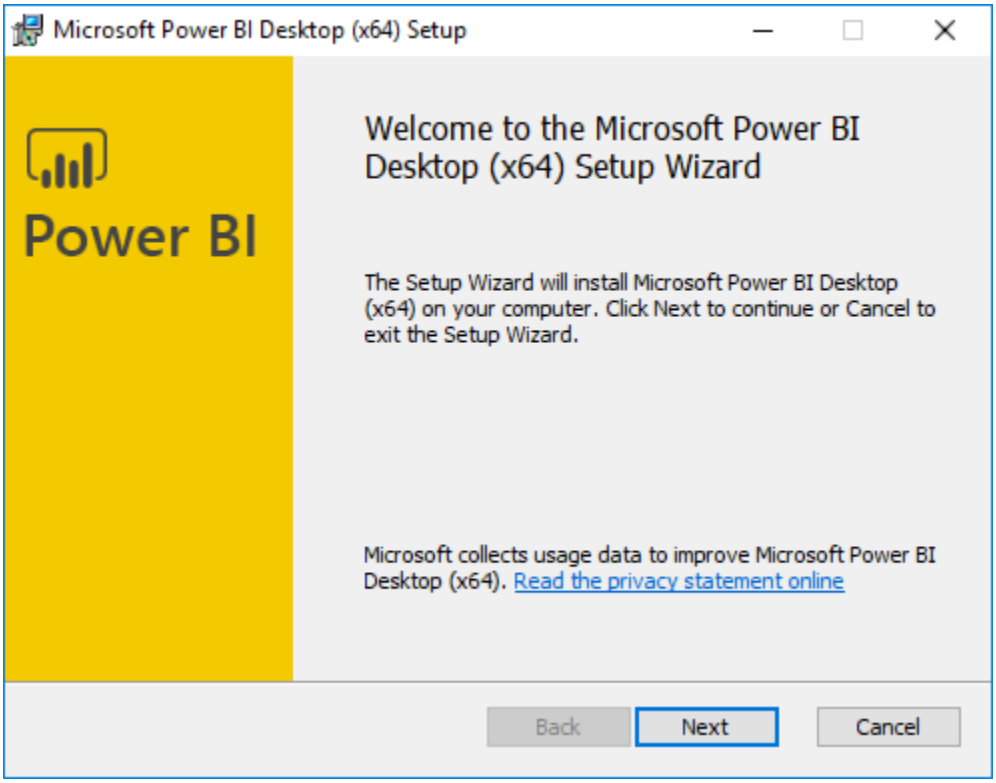
4. On your Lab VM blade, select **Connect** from the top menu.

5. Download and open the RDP file.
6. Select Connect, and enter the following credentials (or the non-default credentials if you changed them):
   - User name: demouser
   - Password: Password.1!!
7. In a web browser on the Lab VM navigate to the Power BI Desktop download page (https://powerbi.microsoft.com/en-us/desktop/).

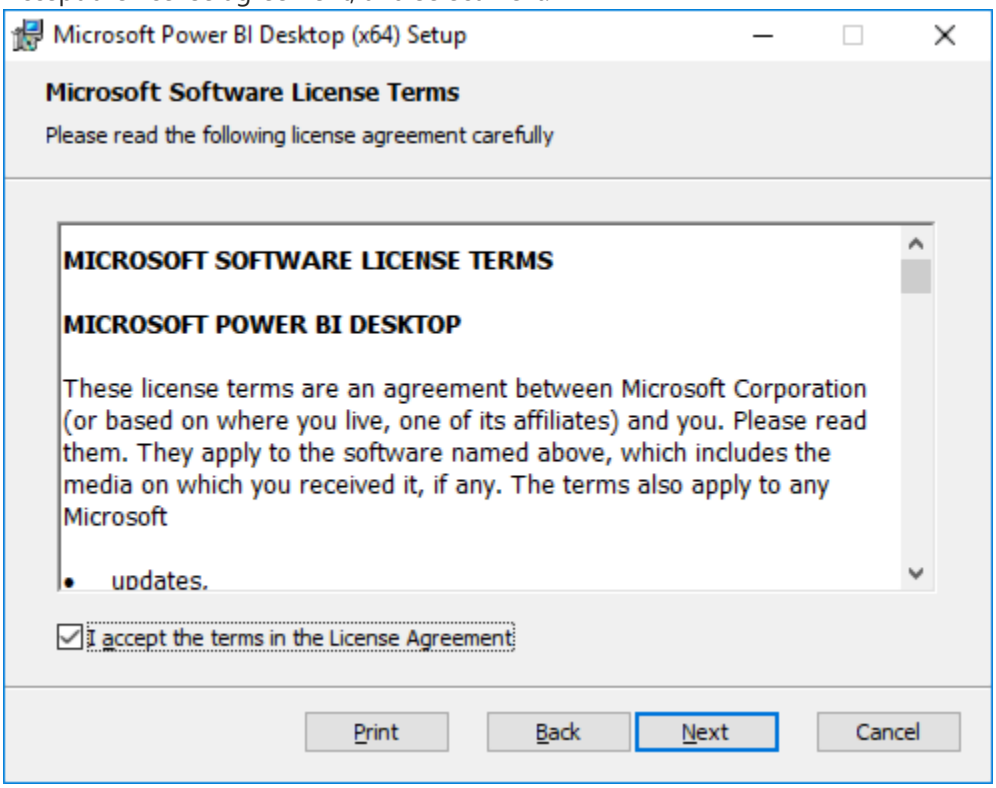
8. Select the Download Free link in the middle of the page.



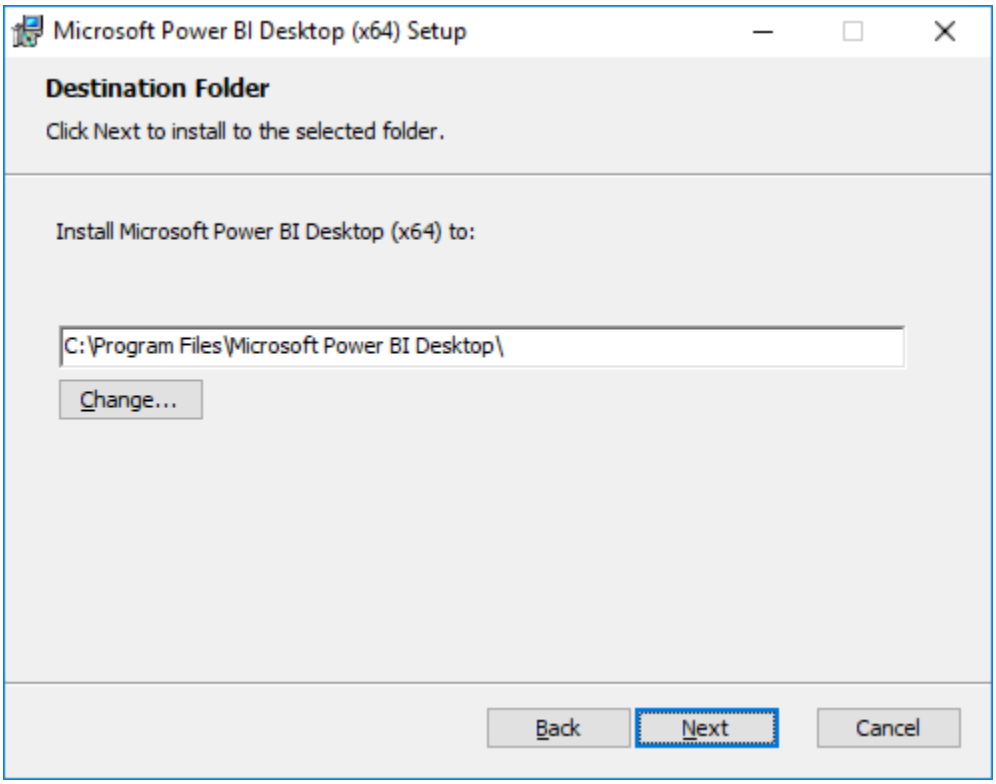
9. Run the installer.
10. Select Next on the welcome screen.



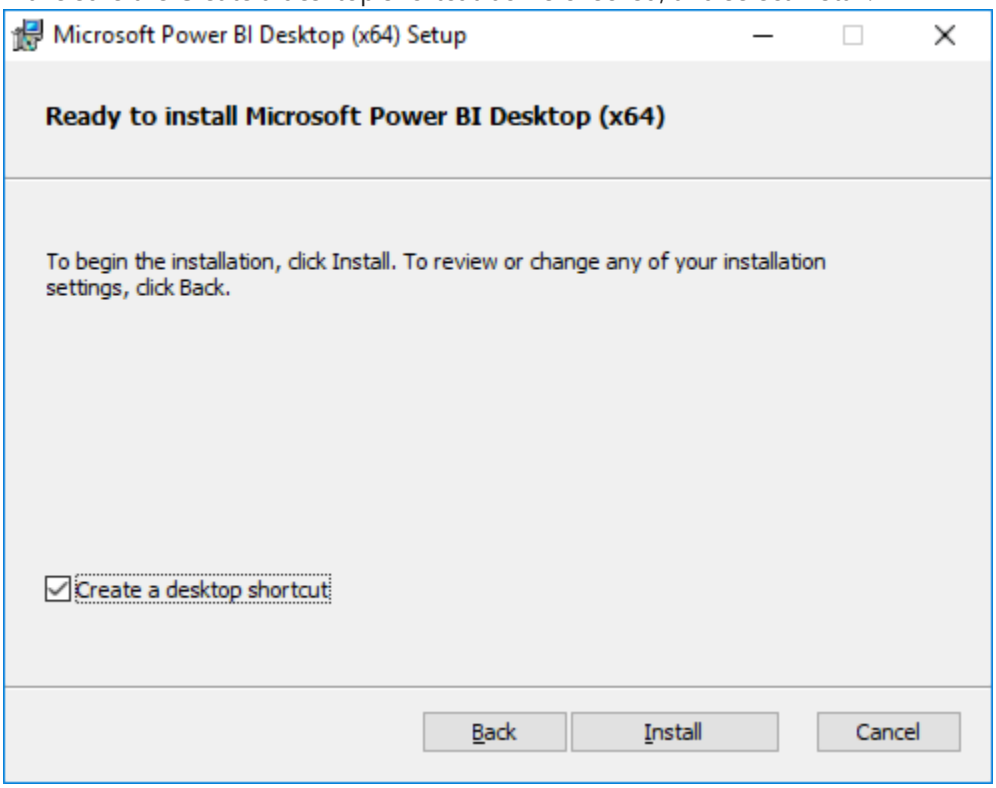
11. Accept the license agreement, and select Next.



12. Leave the default destination folder, and select Next.



13. Make sure the Create a desktop shortcut box is checked, and select Install.



14. Uncheck Launch Microsoft Power BI Desktop, and select Finish.

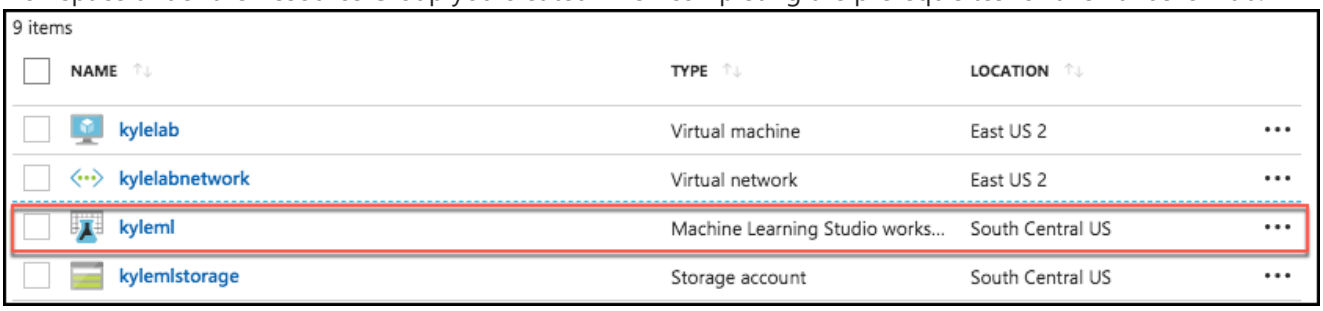
Exercise 1: Build a machine learning model

Duration: 60 mins

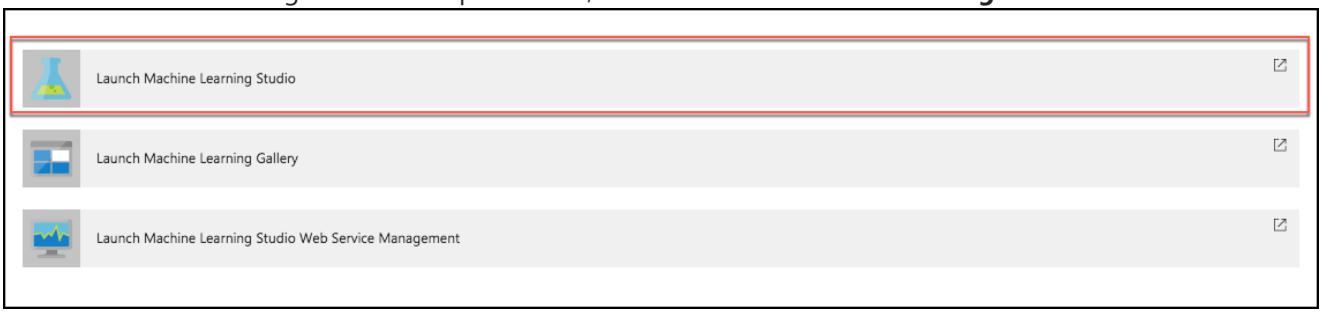
In this exercise, attendees will implement a classification experiment. They will load the training data from their local machine into a dataset. Then, they will explore the data to identify the primary components they should use for prediction and use two different algorithms for predicting the classification. They will evaluate the performance of both and algorithms choose the algorithm that performs best. The model selected will be exposed as a web service that is integrated with the sample web app.

Task 1: Navigate to Machine Learning Studio

1. In a browser, go to the Azure portal (https://portal.azure.com), and navigate to your Machine Learning Studio workspace under the Resource Group you created when completing the prerequisites for this hands-on lab.



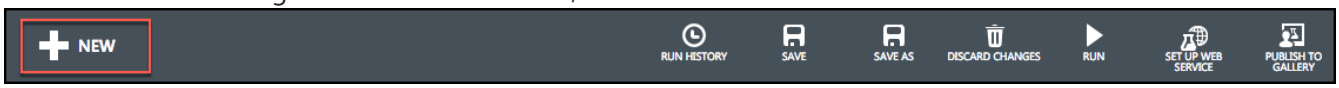
2. On the Machine Learning Studio workspace blade, select **Launch Machine Learning Studio**.



1. Sign in, if prompted.
2. If you have multiple Azure ML workspaces, choose the one you created for this hands-on lab from the drop-down menu near the top right of Azure Machine Learning Studio.

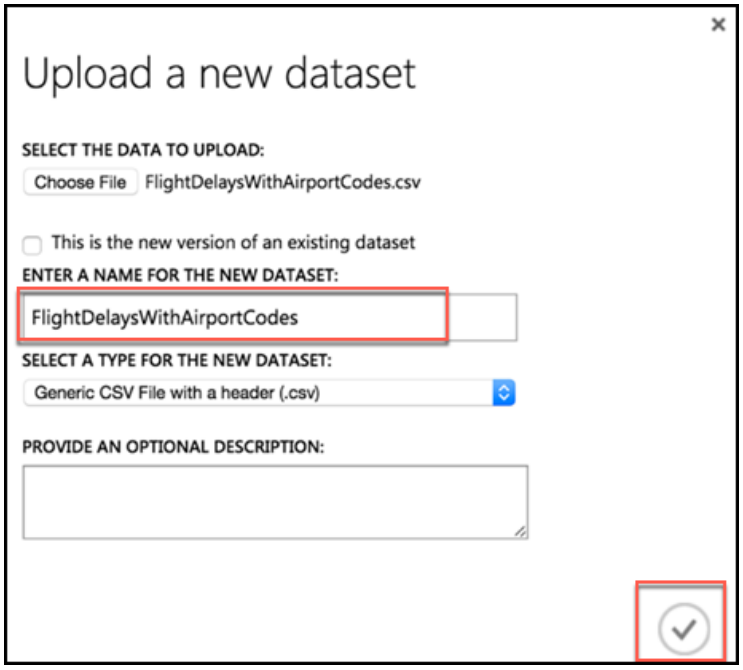
Task 2: Upload the sample datasets

1. Before you begin creating a machine learning experiment, there are three datasets you need to load.
2. Download the three CSV sample datasets from here: http://bit.ly/2wGAqrl (If you get an error, or the page won't open, try pasting the URL into a new browser window and verify the case sensitive URL is exactly as shown).
3. Extract the ZIP and verify you have the following files:
    - FlightDelaysWithAirportCodes.csv
    - FlightWeatherWithAirportCodes.csv
    - AirportCodeLocationLookupClean.csv
4. In the Machine Learning Studio browser window, select **+ NEW** at the bottom left.



5. Select **Dataset** under New, and then select **From Local File**.

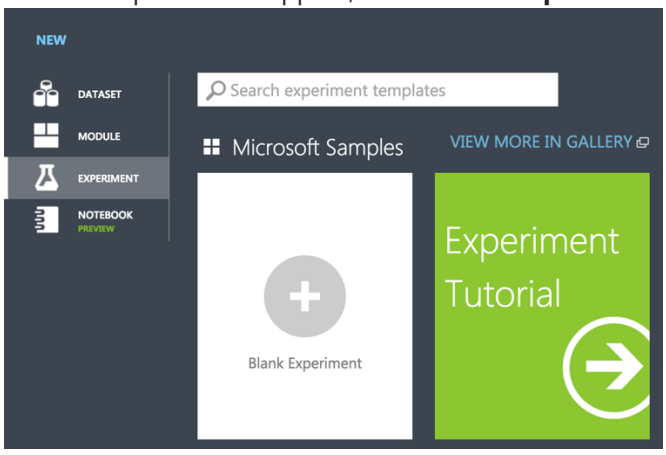
6. In the dialog that appears, select **Choose File**, browse to the FlightDelaysWithAirportCodes.csv file you downloaded in the previous step, and select **Open**.
7. Change the name of the dataset to "FlightDelaysWithAirportCodes," and select the checkmark to upload the data into a new dataset.



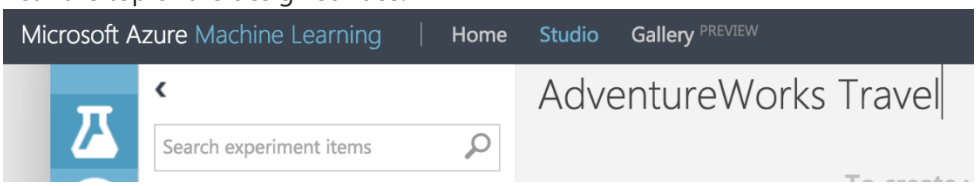
8. Repeat the previous step for the FlightWeatherWithAirportCode.csv and AirportCodeLocationsClean.csv files, setting the name for each dataset in a similar fashion.

Task 3: Start a new experiment

1. Select **+ NEW** in the command bar at the bottom left of the page, and select **Experiment**.
2. From the options that appear, select **Blank Experiment**.

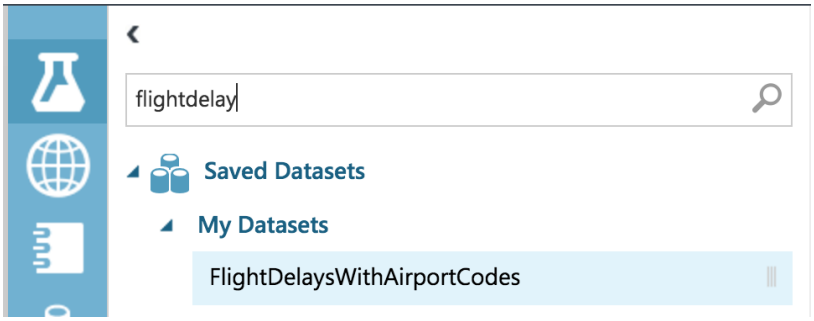


3. Give your new experiment a name, such as AdventureWorks Travel by editing the "Experiment created on ..." label near the top of the design surface.

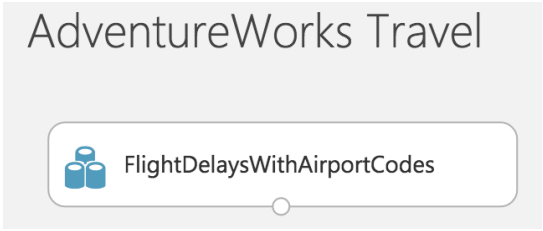


Task 4: Prepare flight delay data

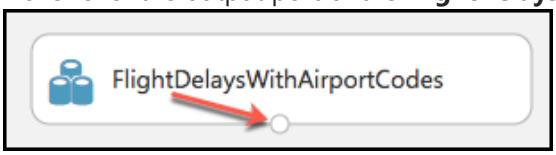
4. In the toolbar on the left, in the **Search experiment items** box, type the name of the dataset you created with flight delay data (FlightDelaysWithAirportCodes). You should see a component for it listed under Saved Datasets, My Datasets.



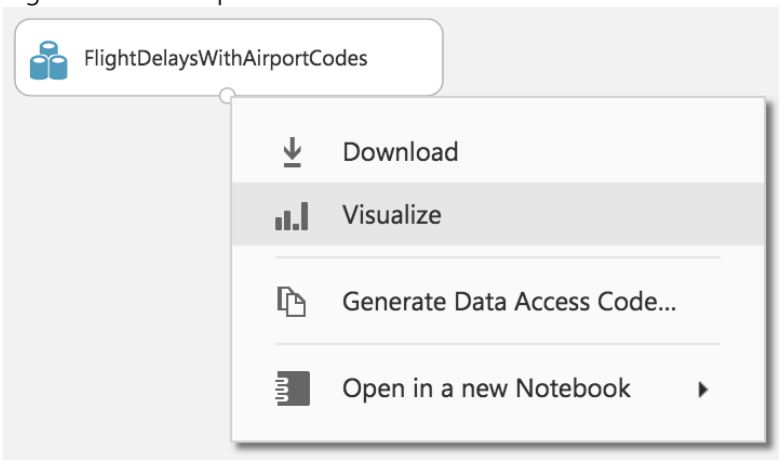
5. Select and drag the FlightDelaysWithAirportCodes module onto the design surface.



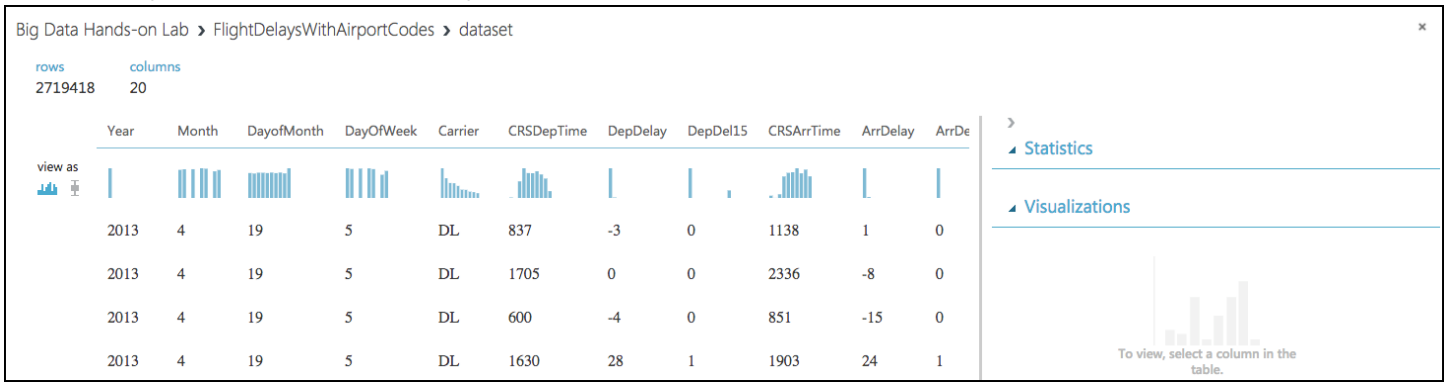
6. Next, you will explore the Flight delays datasets to understand what kind of cleanup (e.g., data munging) will be necessary.
7. Hover over the output port of the **FlightDelaysWithAirportCodes** module.



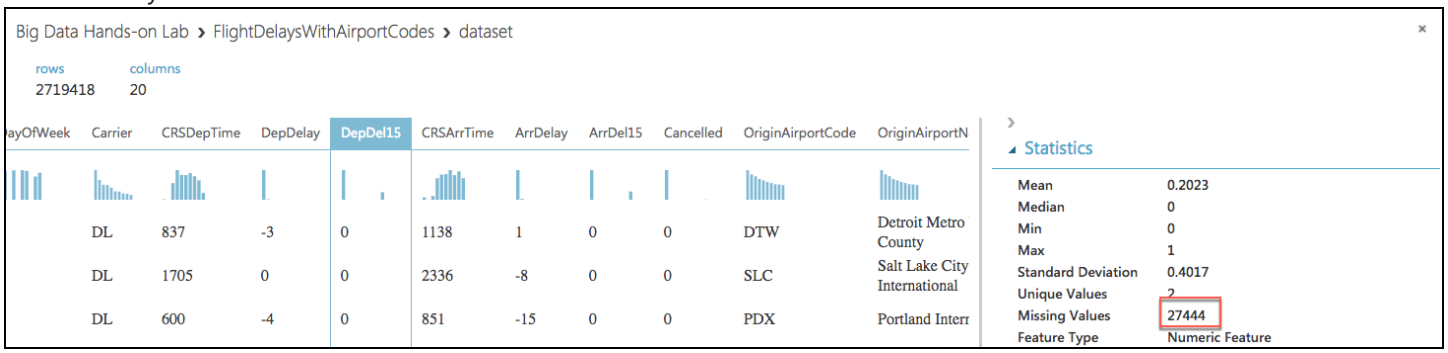
8. Right-click on the port and select **Visualize**.



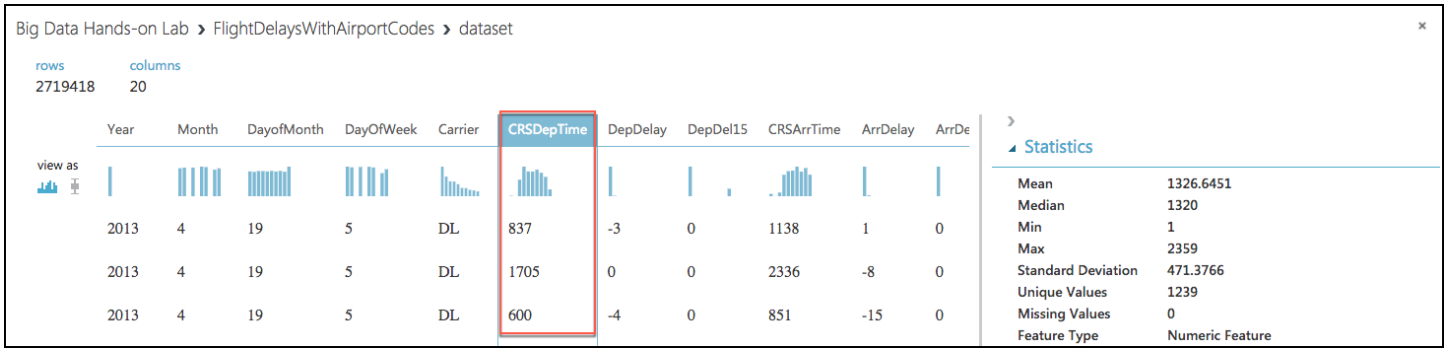
9. A new dialog will appear showing a maximum of 100 rows by 100 columns sample of the dataset. You can see at the top that the dataset has a total of 2,719,418 rows (also referred to as examples in Machine Learning literature) and has 20 columns (also referred to as features).



10. Because all 20 columns are displayed, you can scroll the grid horizontally. Scroll until you see the **DepDel15** column, and select it to view statistics about the column. The DepDel15 column displays a 1 when the flight was delayed at least 15 minutes and 0 if there was no such delay. In the model you will construct, you will try to predict the value of this column for future data. Notice in the Statistics panel that a value of 27444 appears for Missing Values. This means that 27,444 rows do not have a value in this column. Since this value is very important to our model, we will need to eliminate any rows that do not have a value for this column.



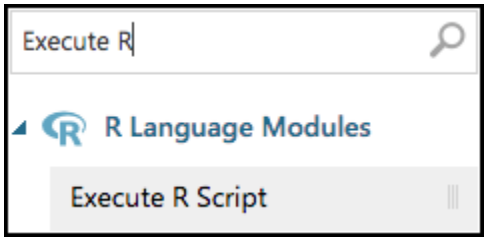
11. Next, select the **CRSDepTime** column. Our model will approximate departure times to the nearest hour, but departure time is captured as an integer. For example, 8:37 am is captured as 837. Therefore, we will need to process the CRSDepTime column, and round it down to the nearest hour. To perform this rounding will require two steps, first you will need to divide the value by 100 (so that 837 becomes 8.37). Second, you will round this value down to the nearest hour (so that 8.37 becomes 8).



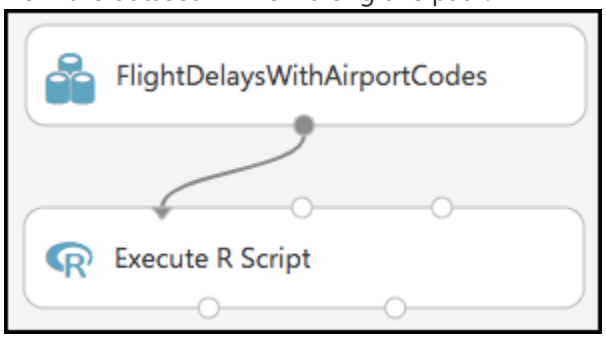
12. Finally, we do not need all 20 columns present in the FlightDelaysWithAirportCodes dataset, so we will need to pare down the columns in the dataset to the 12.
13. Close the Visualize dialog, and go back to the design surface.
14. To perform our data munging, we have multiple options, but in this case, we've chosen to use an **Execute R Script** module, which will perform the following tasks:

a. Remove rows with missing values
b. Generate a new column, named "CRSDepHour," which contains the rounded down value from CRSDepTime
c. Pare down columns to only those needed for our model

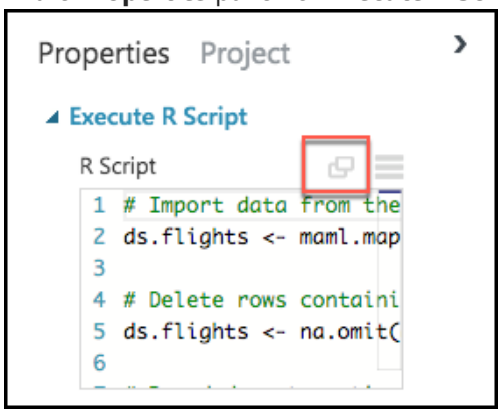
15. To add the module, search for **Execute R Script** by entering "Execute R" into the Search experiment items box.



16. Drag this module on to the design surface beneath your FlightDelaysWithAirportCodes dataset. Select the small circle at the bottom of the FlightDelaysWithAirportCodes dataset, drag and release when your mouse is over the circle found in the top left of the Execute R Script module. These circles are referred to as ports, and by taking this action you have connected the output port of the dataset with the input port of the Execute R Script module, meaning data from the dataset will flow along this path.



17. In the **Properties** panel for **Execute R Script** module, select the **Double Windows** icon to maximize the script editor.



18. Replace the script with the following (Press CTRL+A to select all then CTRL+V to paste)

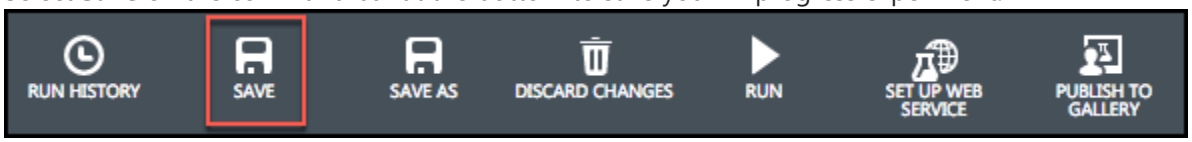
```
# Import data from the input port
ds.flights <- maml.mapInputPort(1)

# Delete rows containing missing values
ds.flights <- na.omit(ds.flights)

# Round departure times down to the nearest hour, and export the result as a new column
named "CRSDepHour"
ds.flights[, "CRSDepHour"] <- floor(ds.flights[, "CRSDepTime"] / 100)
```



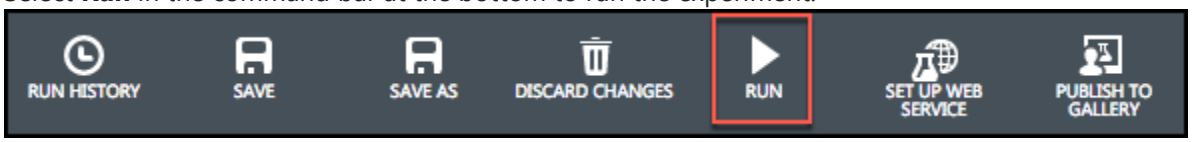
```
# Trim the columns to only those we will use for the predictive model
ds.flights = ds.flights[, c("OriginAirportCode","OriginLatitude", "OriginLongitude",
"Month", "DayofMonth", "CRSDepHour", "DayOfWeek", "Carrier", "DestAirportCode",
"DestLatitude", "DestLongitude", "DepDel15")]

# Export the cleaned up data set
maml.mapOutputPort("ds.flights");
```

19. Select the check mark in the bottom right to save the script (Do not worry if the formatting is off before hitting the check mark.)

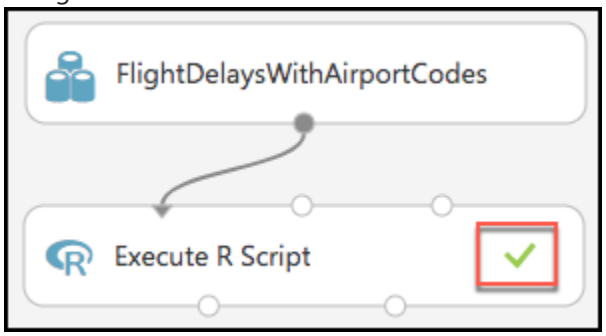
20. Select **Save** on the command bar at the bottom to save your in-progress experiment.



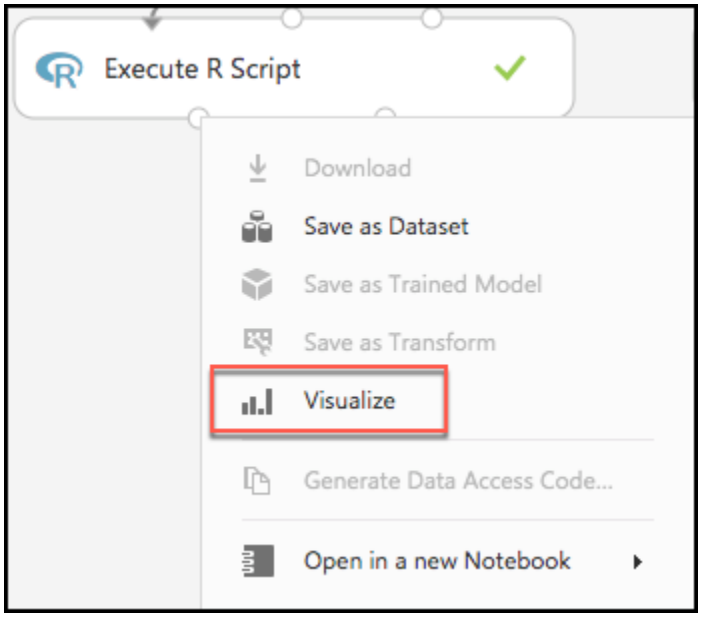
21. Select **Run** in the command bar at the bottom to run the experiment.



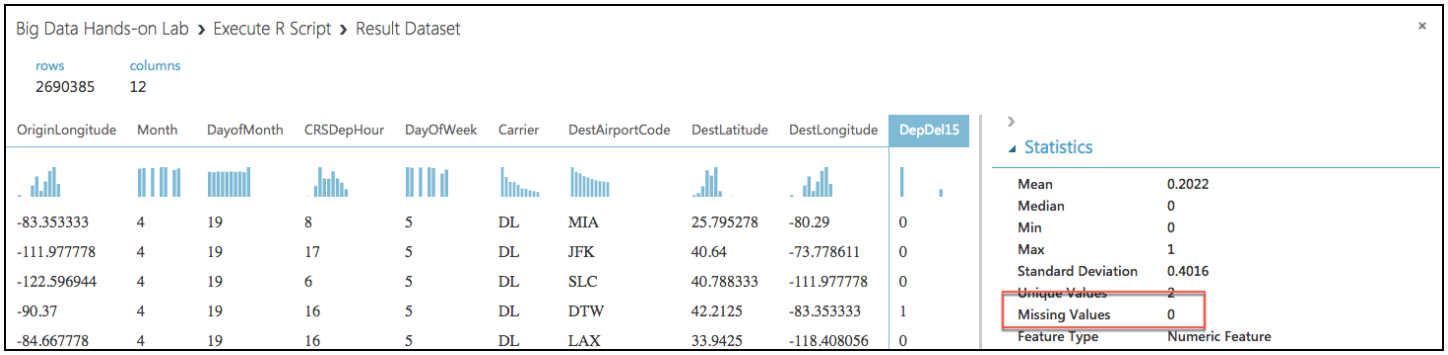
22. When the experiment is finished running, you will see a finished message in the top right corner of the design surface, and green check marks over all modules that ran.



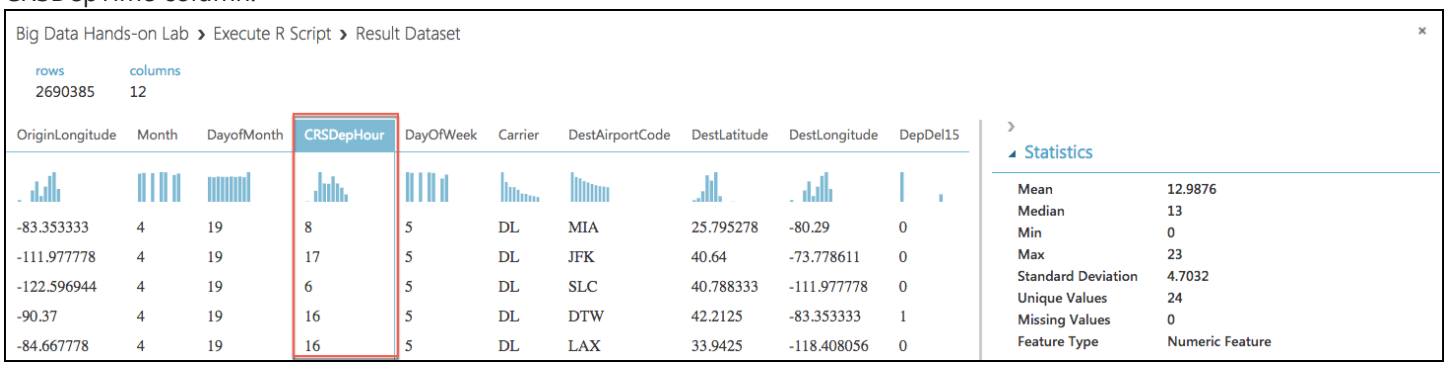
23. You should run your experiment whenever you need to update the metadata describing what data is flowing through the modules, so that newly added modules can be aware of the shape of your data (most modules have dialogs that can suggest columns, but before they can make suggestions you need to have run your experiment).
24. To verify the results of our R script, right-click the left output port (Result Dataset) of the Execute R Script module and select **Visualize**.



25. In the dialog that appears, scroll over to DepDel15 and select the column. In the statistics you should see that Missing Values reads 0.



26. Now, select the CRSDepHour column, and verify that our new column contains the rounded hour values from our CRSDepTime column.

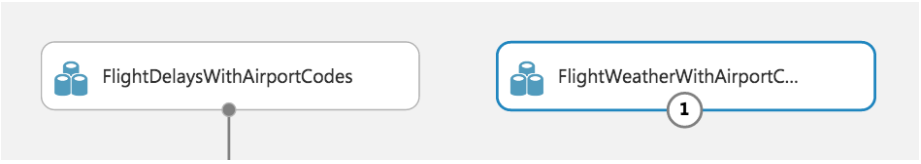


27. Finally, observe that we have reduced the number of columns from 20 to 12. Close the dialog.

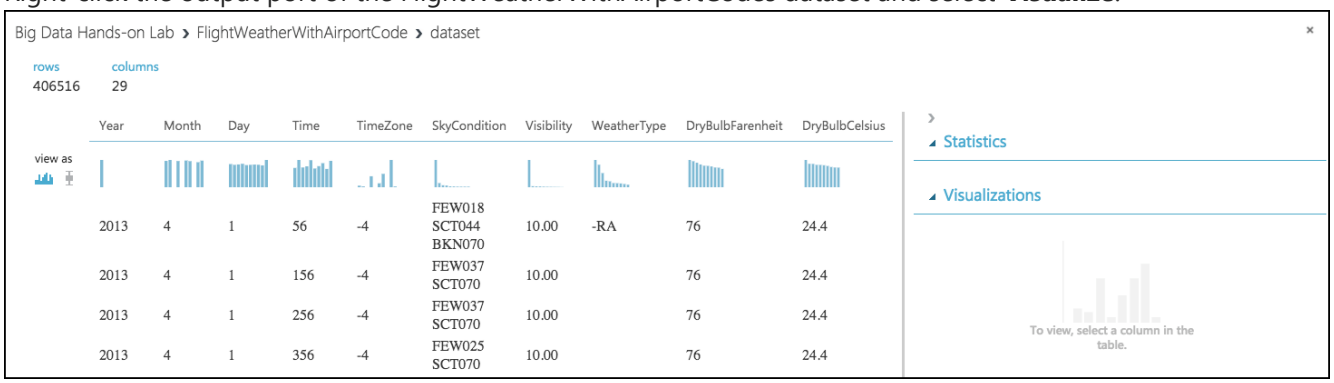
28. At this point the Flight Delay Data is prepared, and we turn to preparing the historical weather data.

Task 5: Prepare the weather data

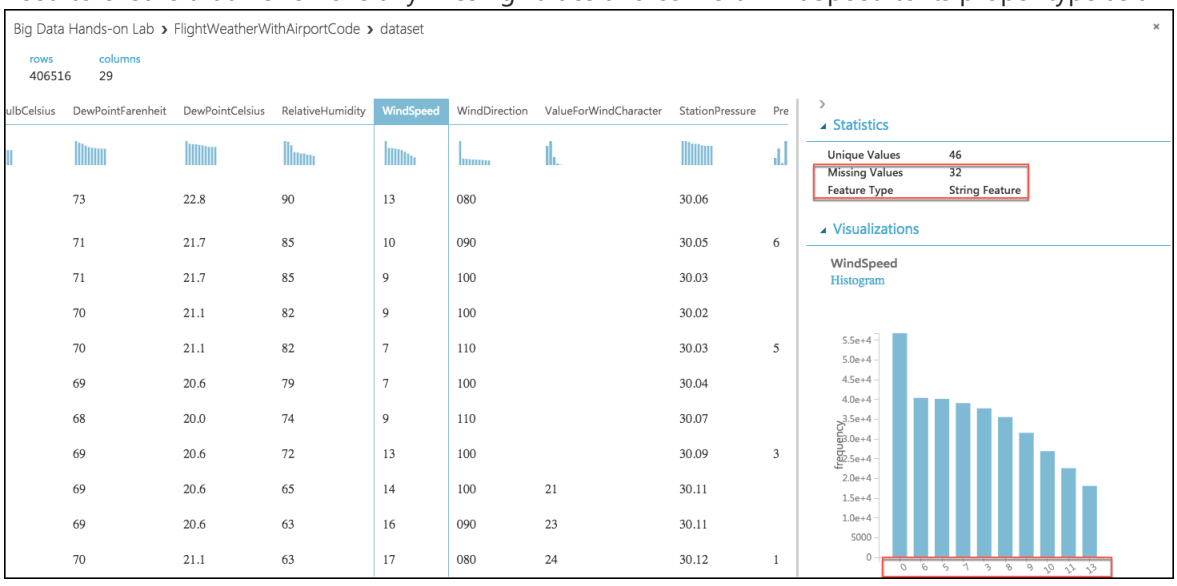
1. To the right of the FlightDelaysWithAirportCodes dataset, add the **FlightWeatherWithAirportCodes** dataset.



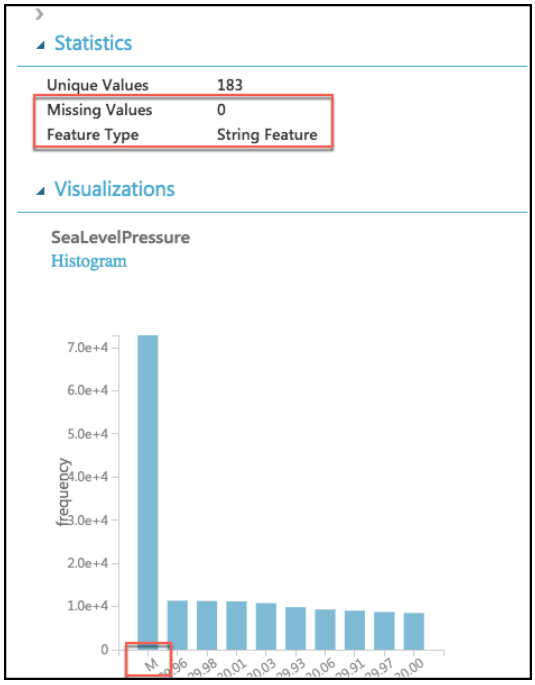
2. Right-click the output port of the FlightWeatherWithAirportCodes dataset and select **Visualize**.



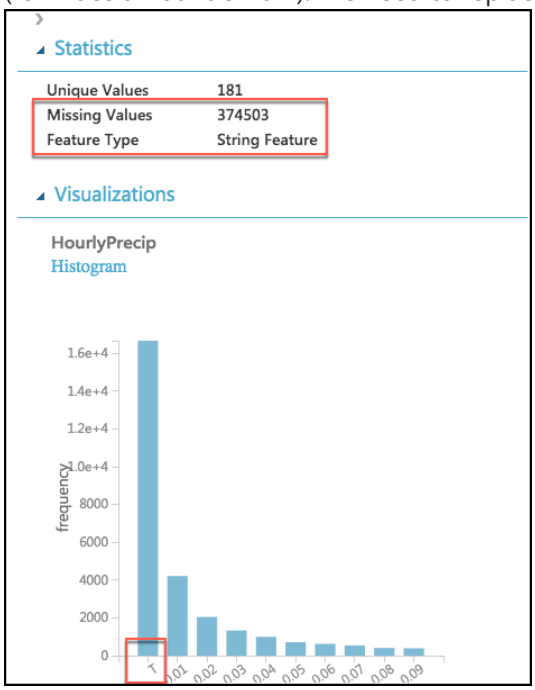
3. Observe that this data set has 406,516 rows and 29 columns. For this model, we are going to focus on predicting delays using WindSpeed (in MPH), SeaLevelPressure (in inches of Hg), and HourlyPrecip (in inches). We will focus on preparing the data for those features.
4. In the dialog, select the **WindSpeed** column, and review the statistics. Observe that the Feature Type was inferred as String and that there are 32 Missing Values. Below that, examine the histogram to see that, even though the type was inferred as string, the values are all actually numbers (e.g. the x-axis values are 0, 6, 5, 7, 3, 8, 9, 10, 11, 13). We will need to ensure that we remove any missing values and convert WindSpeed to its proper type as a numeric feature.



5. Next, select the **SeaLevelPressure** column. Observe that the Feature Type was inferred as String and there are 0 Missing Values. Scroll down to the histogram, and observe that many of the features are of a numeric value (e.g., 29.96, 30.01, etc.), but there are many features with the string value of M for Missing. We will need to replace this value of "M" with a suitable numeric value so that we can convert this feature to be a numeric feature.



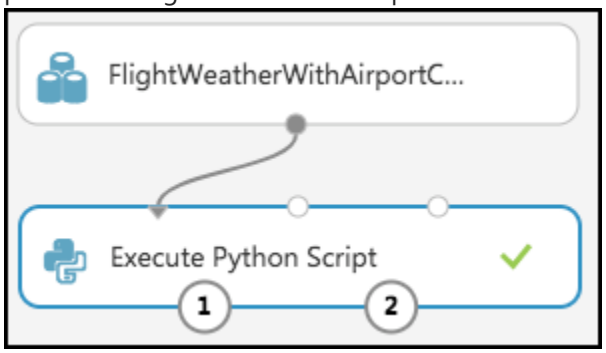
6. Finally, examine the **HourlyPrecip** feature. Observe that it too was inferred to have a Feature Type of String and is missing values for 374,503 rows. Looking at the histogram, observe that besides the numeric values, there is a value T (for Trace amount of rain). We need to replace T with a suitable numeric value and covert this to a numeric feature.



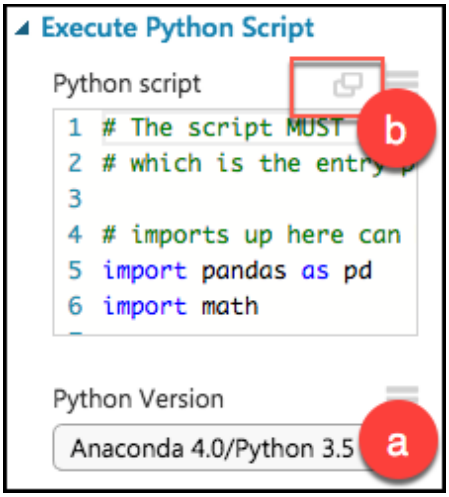
7. To preform our data cleanup, we will use a Python script, in which we will perform the following tasks:
    a. WindSpeed: Replace missing values with 0.0, and "M" values with 0.005
    b. HourlyPrecip: Replace missing values with 0.0, and "T" values with 0.005
    c. SeaLevelPressure: Replace "M" values with 29.92 (the average pressure)

d. Convert WindSpeed, HourlyPrecip, and SeaLevelPressure to numeric columns
e. Round "Time" column down to the nearest hour, and add value to a new column named "Hour"
f. Eliminate unneeded columns from the dataset

8. Add an **Execute Python Script** module below the FlightWeatherWithAirportCode module, and connect the output port of the FlightWeatherWithAirportCode module to the first input port of the Execute Python Script module.



9. In the **Properties** panel for the Execute Python Script:
   a. Set the Python Version to **Anaconda 4.0/Python 3.5**
   b. Select the **Double Windows** icon to open the script editor.

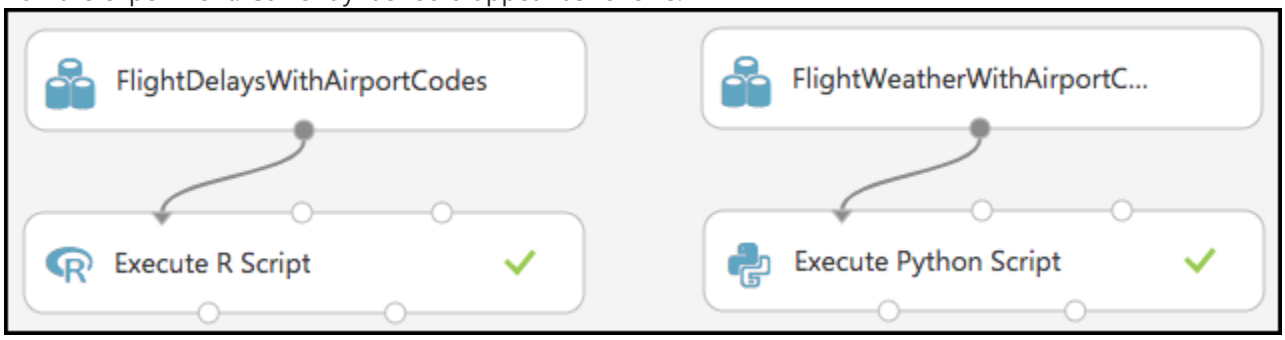


10. Paste in the following script into the Python script window, and select the checkmark at the bottom right of the dialog (press CTRL+A to select all then CTRL+V to paste and then immediately select the checkmark -- don't worry if the formatting is off before hitting the checkmark).

```
# imports
import pandas as pd
import math

# The entry point function can contain up to two input arguments:
#   Param<dataframe1>: a pandas.DataFrame
#   Param<dataframe2>: a pandas.DataFrame
def azureml_main(dataframe1 = None, dataframe2 = None):

    # Round weather Time down to the next hour, since that is the hour for which we want to
use flight dataframe1
    # Add the rounded Time to a new column named "Hour," and append that column to the
dataframe1
    dataframe1["Hour"] = dataframe1["Time"].apply(roundDown)
```

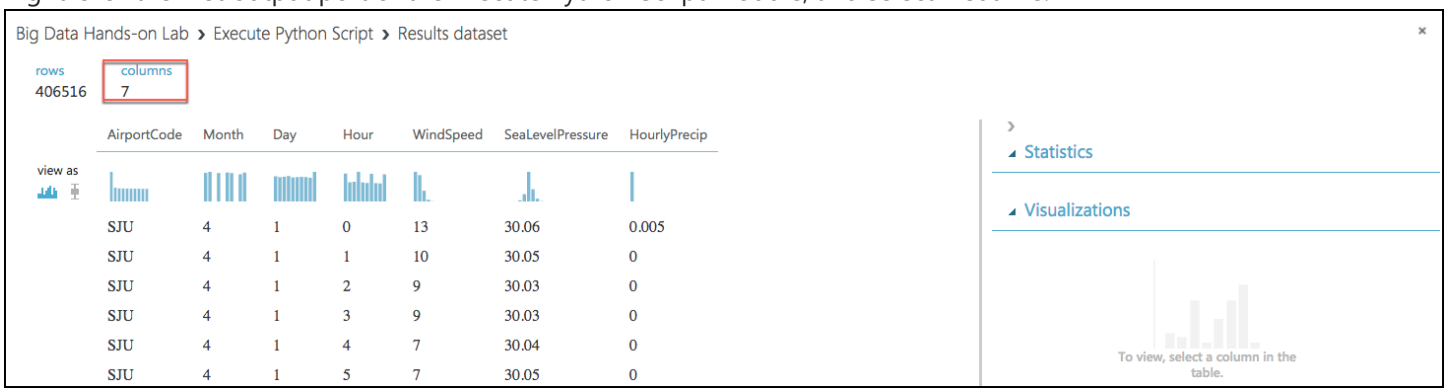
```
    # Replace any missing HourlyPrecip and WindSpeed values with 0.0
    dataframe1["HourlyPrecip"] = dataframe1["HourlyPrecip"].fillna('0.0')
    dataframe1["WindSpeed"] = dataframe1["WindSpeed"].fillna('0.0')

    # Replace any WindSpeed values of "M" with 0.005
    dataframe1["WindSpeed"] = dataframe1['WindSpeed'].replace(['M'], '0.005')

    # Replace any SeaLevelPressure values of "M" with 29.92 (the average pressure)
    dataframe1["SeaLevelPressure"] = dataframe1['SeaLevelPressure'].replace(['M'], '29.92')

    # Replace any HourlyPrecip values of "T" (trace) with 0.005
    dataframe1["HourlyPrecip"] = dataframe1['HourlyPrecip'].replace(['T'], '0.005')

    # Convert our WindSpeed, SeaLevelPressure, and HourlyPrecip columns to numeric
    dataframe1[['WindSpeed','SeaLevelPressure', 'HourlyPrecip']] =
dataframe1[['WindSpeed','SeaLevelPressure', 'HourlyPrecip']].apply(pd.to_numeric)

    # Pare down the variables in the Weather dataset to just the columns being used by the
model
    df_result = dataframe1[['AirportCode', 'Month', 'Day', 'Hour', 'WindSpeed',
'SeaLevelPressure', 'HourlyPrecip']]

    # Return value must be of a sequence of pandas.DataFrame
    return df_result

def roundDown(x):
    z = int(math.floor(x/100.0))
    return z
```

11. Run the experiment. Currently it should appear as follows:

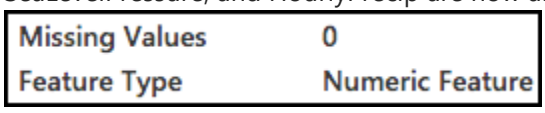


12. If you receive an error in the Python script that .to_numeric does not exist, go back and verify that you selected the proper Python version.

13. Right-click the first output port of the Execute Python Script module, and select Visualize.

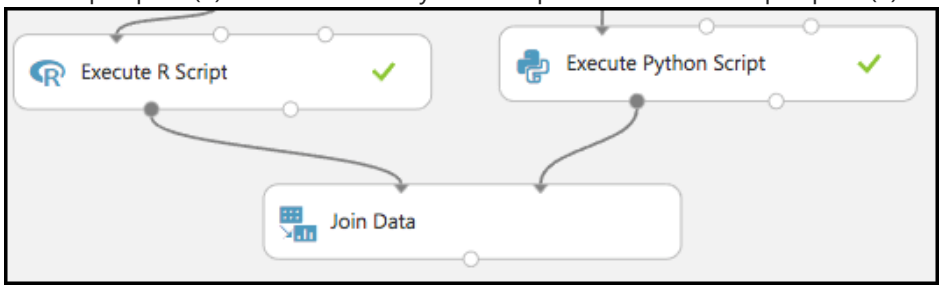


14. In the statistics, verify that there are now only the 7 columns we are interested in, and that WindSpeed, SeaLevelPressure, and HourlyPrecip are now all Numeric Feature types and that they have no missing values.



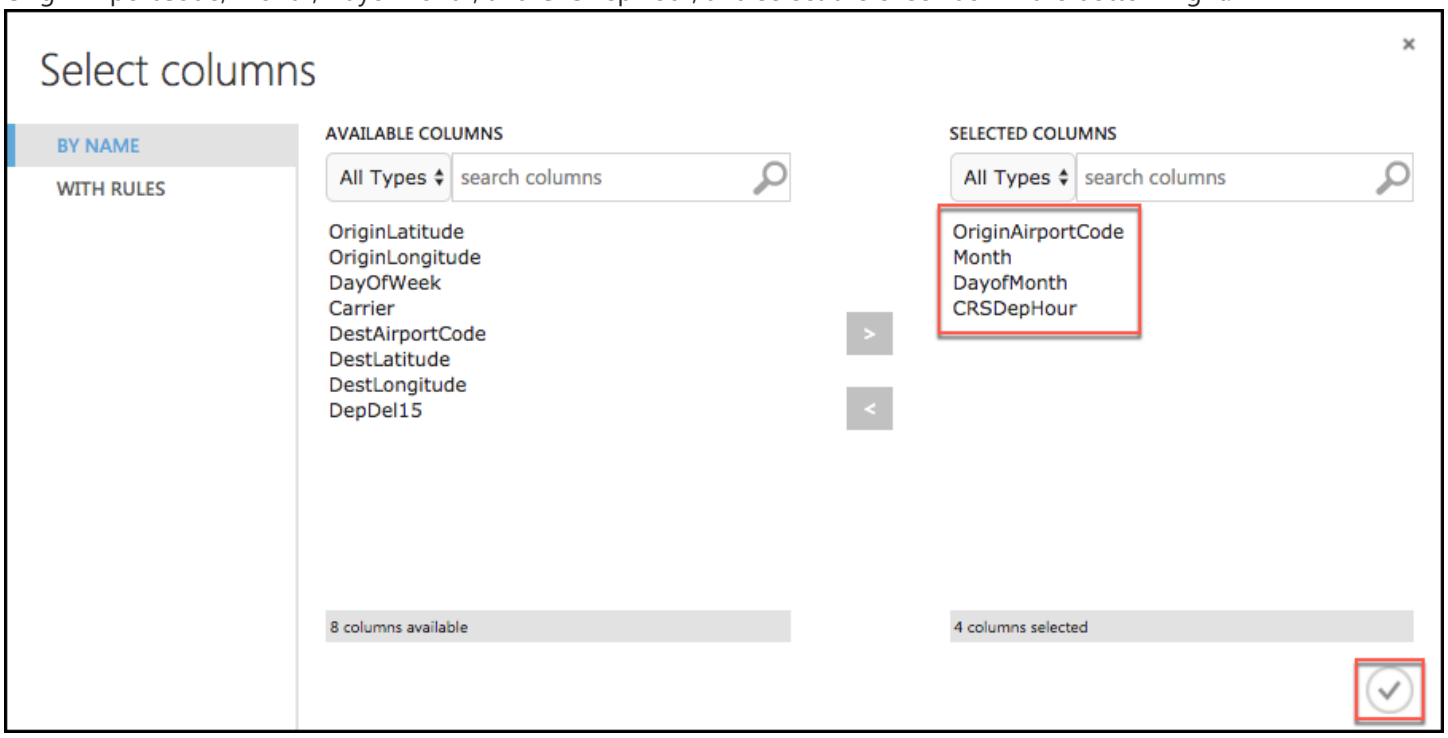
Task 6: Join the flight and weather datasets

1. With both datasets ready, we want to join them together so that we can associate historical flight delays with the weather data at departure time.
2. Drag a **Join Data** module onto the design surface, beneath and centered between both Execute R and Python Script modules. Connect the output port (1) of the Execute R Script module to input port (1) of the Join Data module, and the output port (1) of the Execute Python Script module to the input port (2) of the Join Data module.

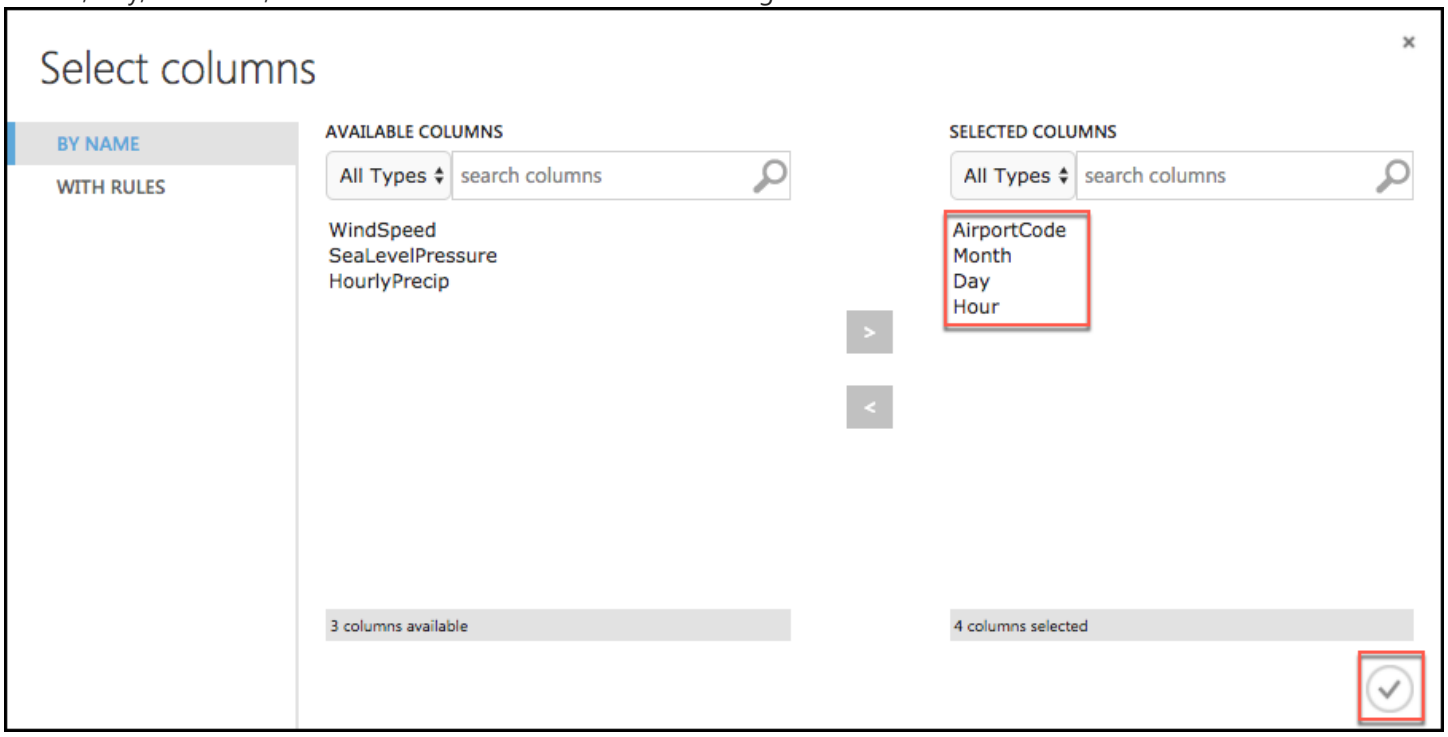


3. In the **Properties** panel for the Join Data module, relate the rows of data between the two sets L (the flight delays) and R (the weather).

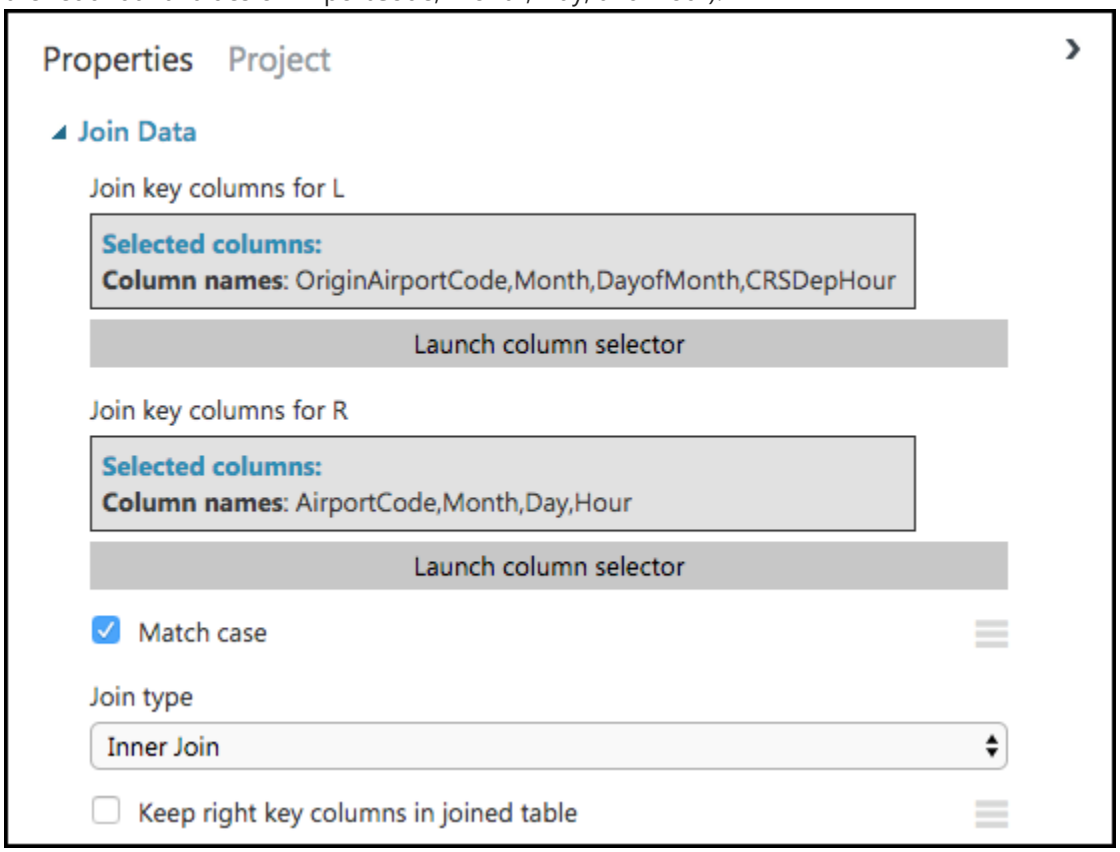
4. Select **Launch Column selector** under **Join key columns for L**. Set the Join key columns for L to include OriginAirportCode, Month, DayofMonth, and CRSDepHour, and select the check box in the bottom right.



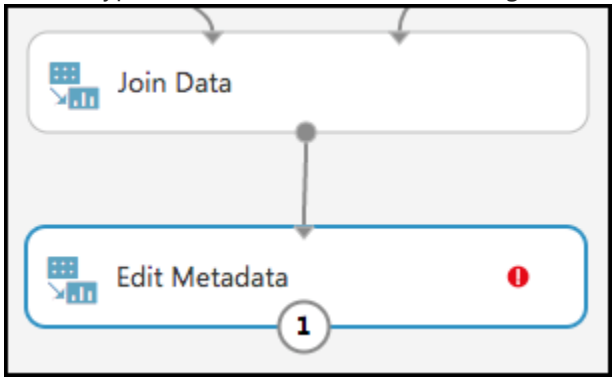
5. Select **Launch Column selector** under **Join key columns for R**. Set the join key columns for R to include AirportCode, Month, Day, and Hour, and select the check box in the bottom right.



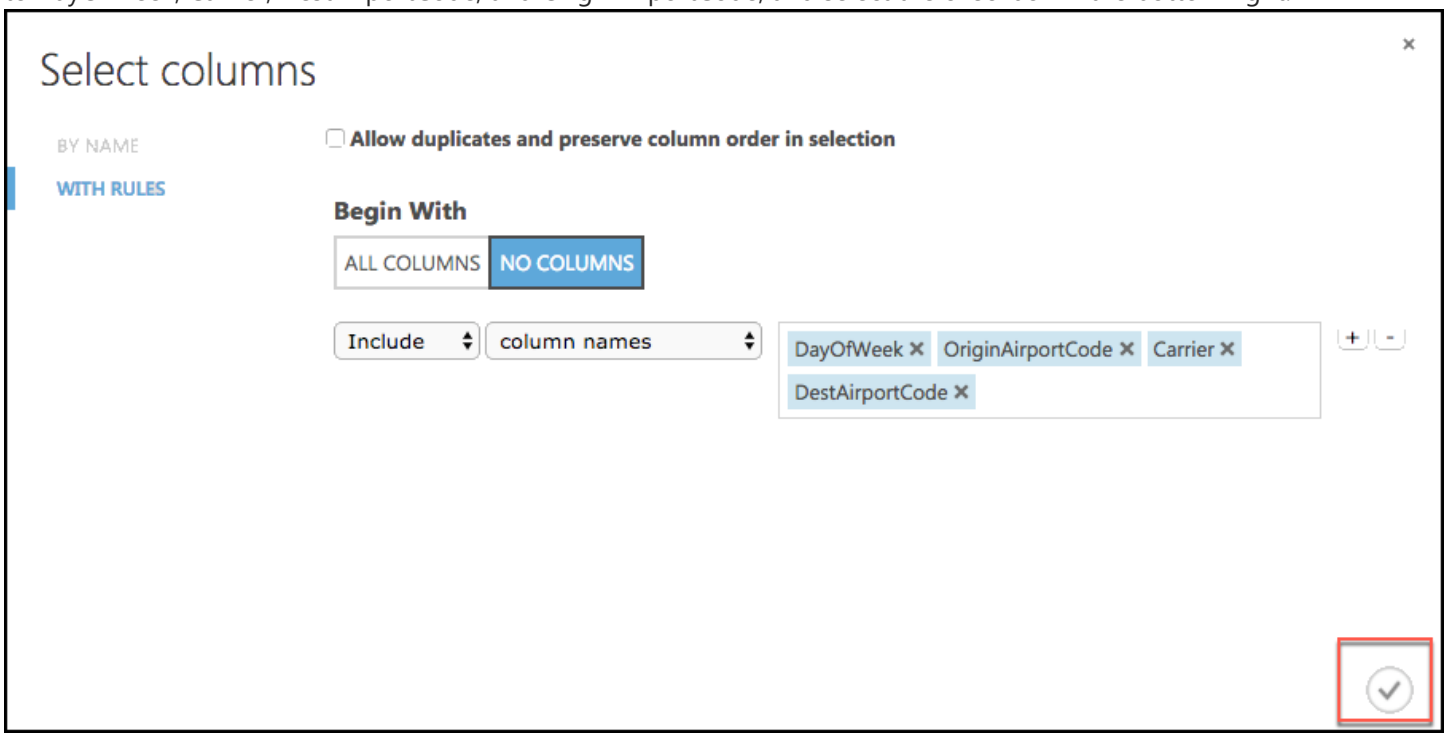
6. Leave the Join Type at Inner Join, and uncheck **Keep right key columns in joined table** (so that we do not include the redundant values of AirportCode, Month, Day, and Hour).



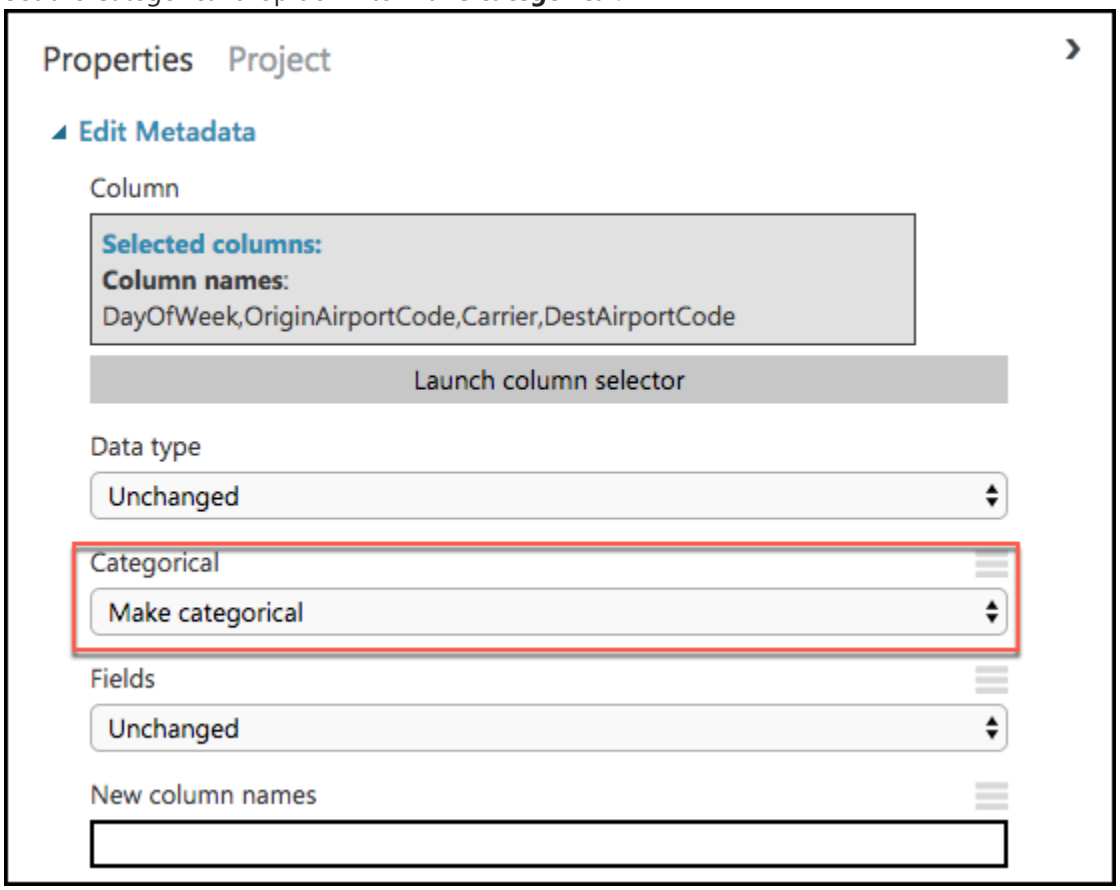
7. Next, drag an **Edit Metadata** module onto the design surface below the Join Data module, and connect its input port to the output port of the Join Data module. We will use this module to convert the fields that were unbounded String feature types, to the enumeration like Categorical feature.



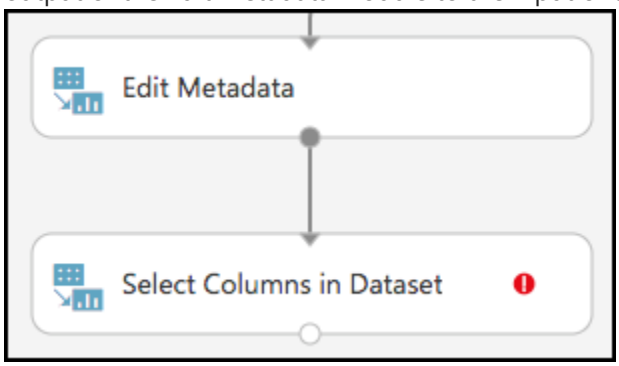
8. On the **Properties** panel of the Edit Metadata module, select **Launch column selector** and set the Selected columns to DayOfWeek, Carrier, DestAirportCode, and OriginAirportCode, and select the checkbox in the bottom right.



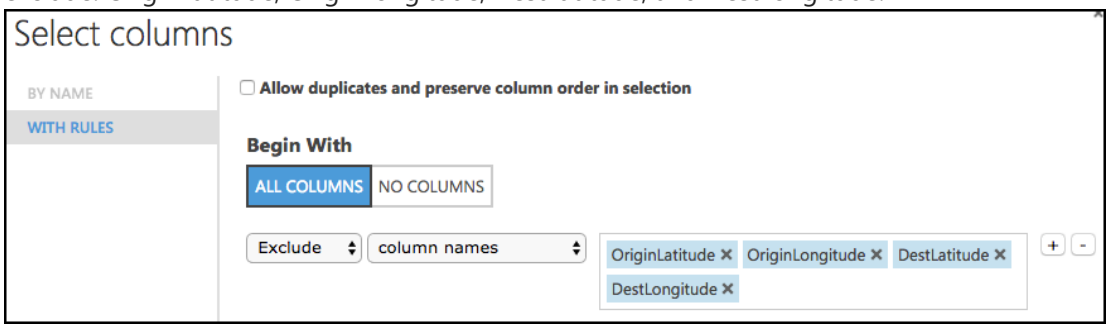
9. Set the Categorical drop down to **Make categorical**.



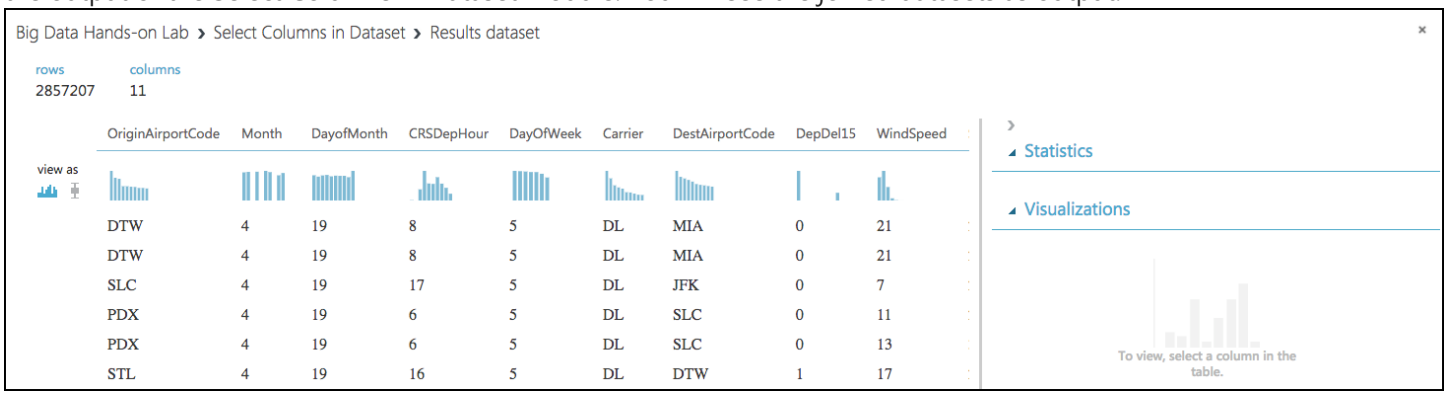
10. Drag a **Select Columns in Dataset** module onto the design surface, below the Edit Metadata module. Connect the output of the Edit Metadata module to the input of the Select Columns in Dataset module.



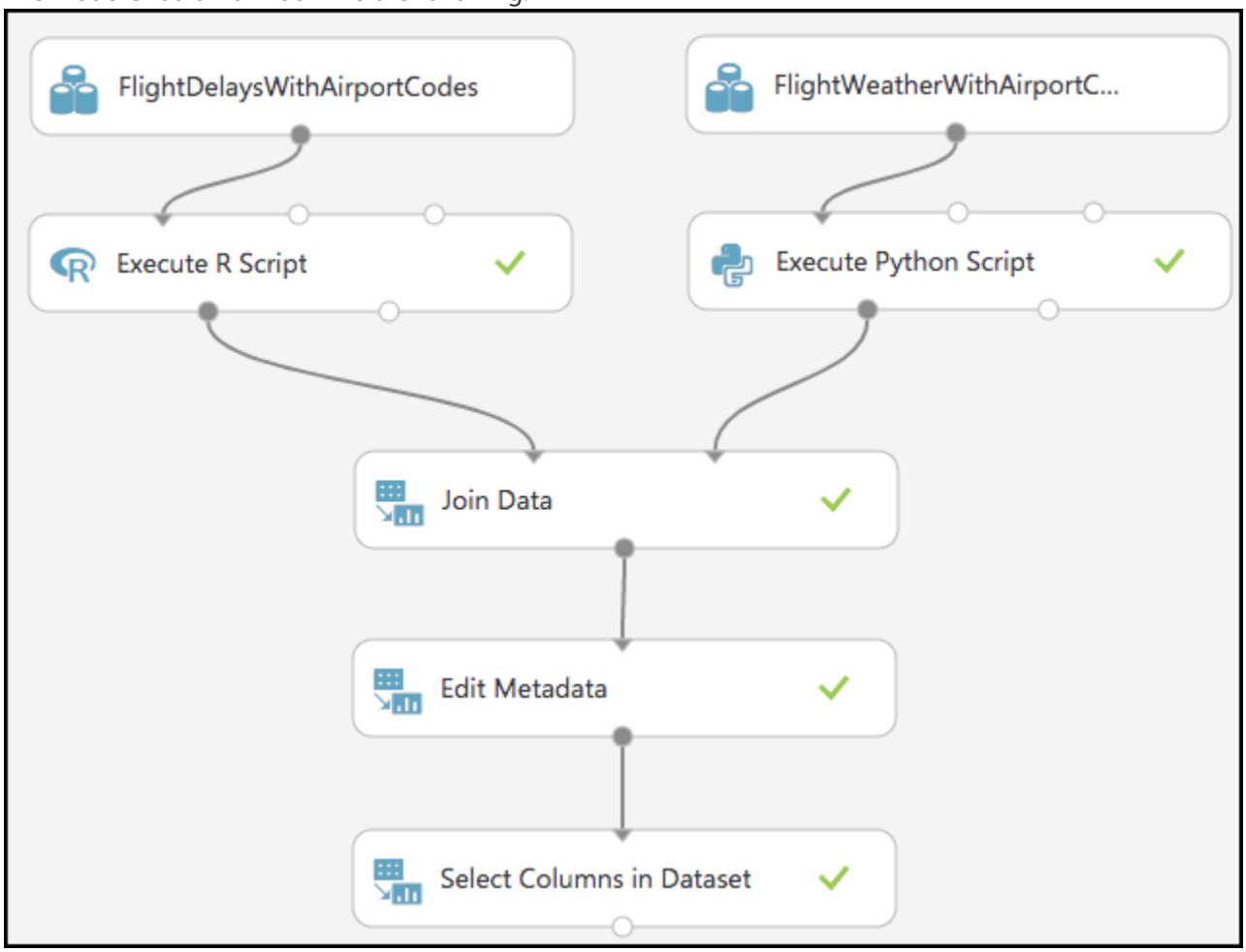
11. Launch the column selector, and choose Begin With **All Columns**, choose **Exclude** and set the selected columns to exclude: OriginLatitude, OriginLongitude, DestLatitude, and DestLongitude.



12. Save your experiment.
13. Run the experiment, to verify everything works as expected, and when completed select Visualize by right-clicking on the output of the Select Columns in Dataset module. You will see the joined datasets as output.



14. The model should now look like the following.

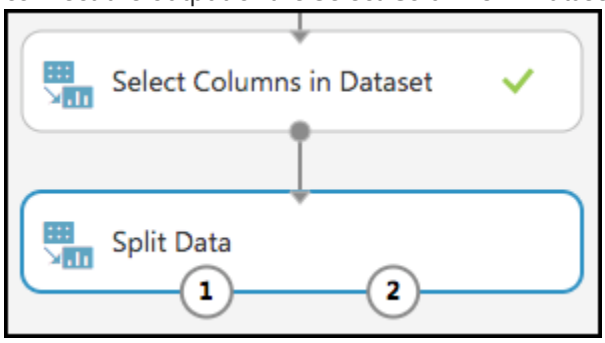


Task 7: Train the model

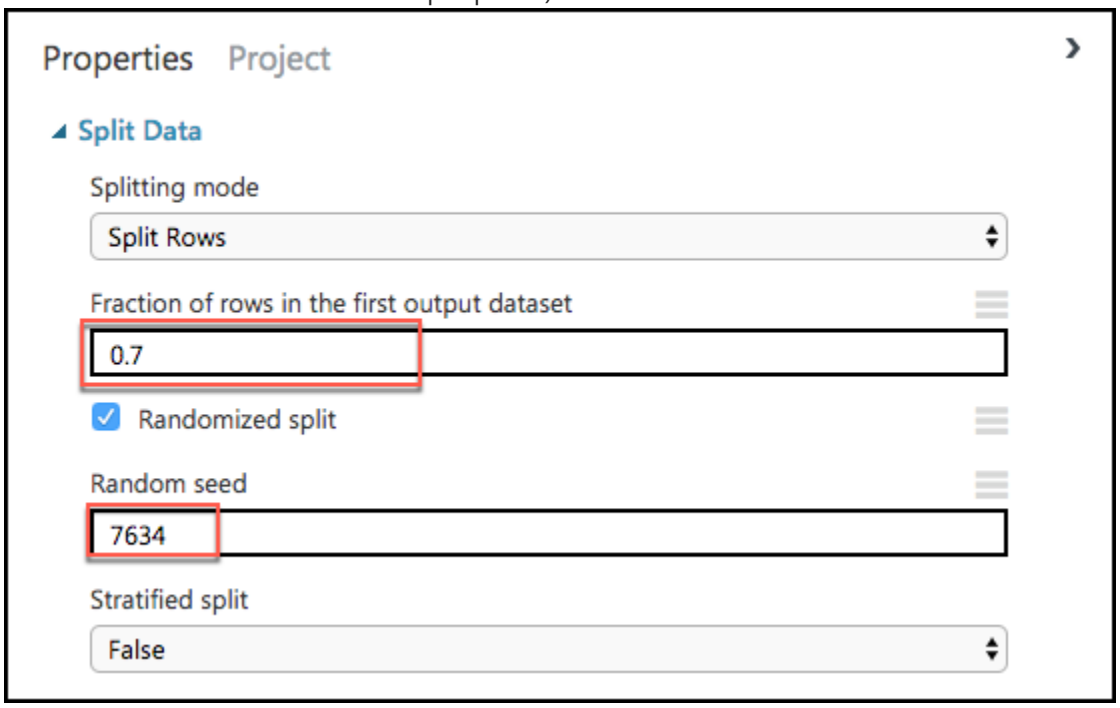
AdventureWorks Travel wants to build a model to predict if a departing flight will have a 15-minute or greater delay. In the historical data they have provided, the indicator for such a delay is found within the DepDel15 (where a value of 1 means delay, 0 means no delay). To create a model that predicts such a binary outcome, we can choose from the various Two-Class modules that Azure ML offers. For our purposes, we begin with a Two-Class Logistic Regression. This type of classification module needs to be first trained on sample data that includes the features important to making a prediction and must also include the actual historical outcome for those features.

The typical pattern is to split the historical data so a portion is shown to the model for training purposes, and another portion is reserved to test just how well the trained model performs against examples it has not seen before.

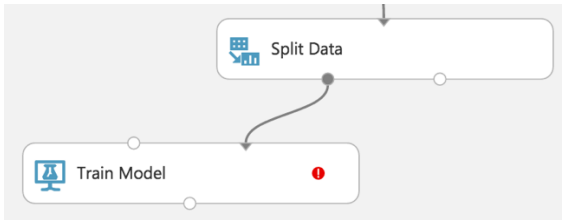
1. To create our training and validation datasets, drag a **Split Data** module beneath Select Columns in Dataset, and connect the output of the Select Columns in Dataset module to the input of the Split Data module.



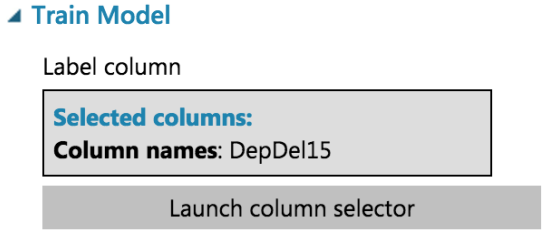
2. On the **Properties** panel for the Split Data module, set the Fraction of rows in the first output dataset to **0.7** (so 70% of the historical data will flow to output port 1). Set the Random seed to **7634**.



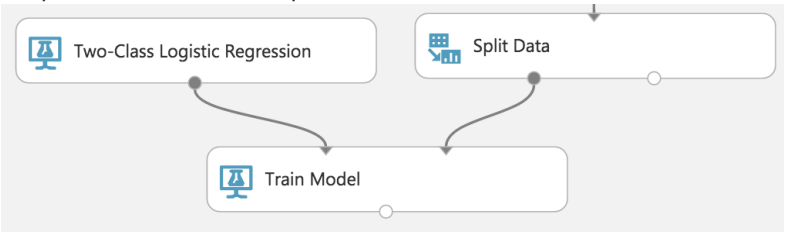
3. Next, add a Train Model module and connect it to output 1 of the Split Data module.



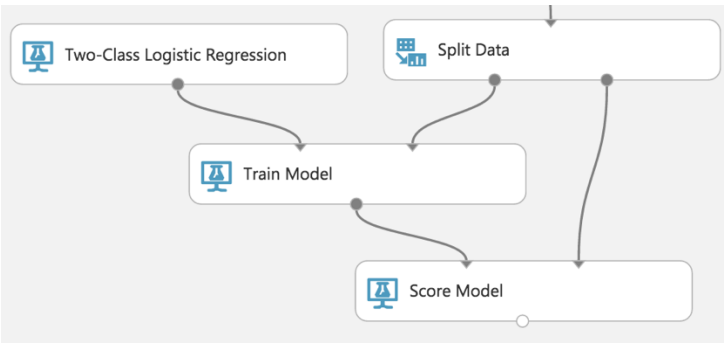
4. On the **Properties** panel for the Train Model module, set the Selected columns to **DepDel15**.



5. Drag a Two-Class Logistic Regression module above and to the left of the Train Model module and connect the output to the leftmost input of the Train Model module



6. Below the Train Model drop a Score Model module. Connect the output of the Train Model module to the leftmost input port of the Score Model and connect the rightmost output of the Split Data module to the rightmost input of the Score Model.



7. Save the experiment.
8. Run the experiment.
9. When the experiment is finished running (which takes a few minutes), right-click on the output port of the Score Model module and select **Visualize** to see the results of its predictions. **You should have a total of 13 columns**.

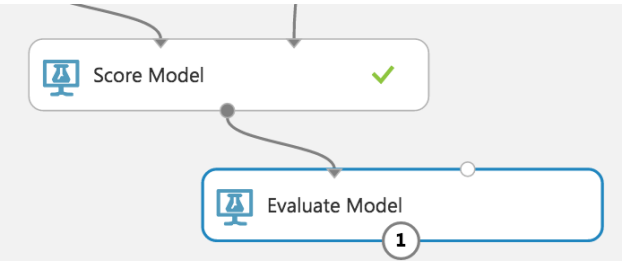
AdventureWorks Travel › Score Model › Scored dataset

rows	columns
861324	13

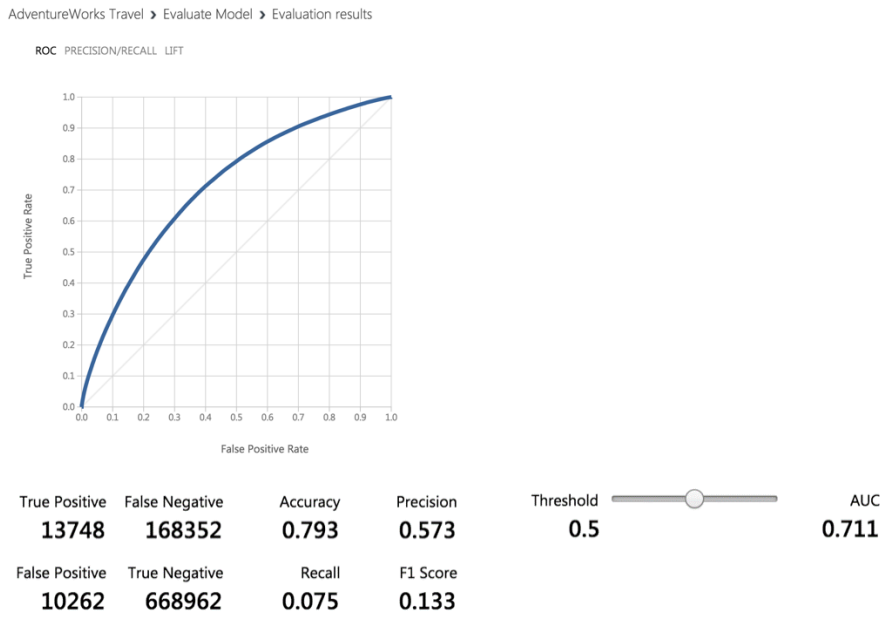
10. If you scroll to the right so that you can see the last two columns, observe there are Scored Labels and Scored Probabilities columns. The former is the prediction (1 for predicting delay, 0 for predicting no delay) and the latter is the probability of the prediction. In the following screenshot, for example, the last row shows a delay predication with a 53.1% probability.

Scored Labels	Scored Probabilities
0	0.224147
0	0.182701
0	0.073537
0	0.270637
0	0.267242
0	0.191029
0	0.095636
0	0.207545
0	0.22024
0	0.082833
1	0.531273

11. While this view enables you to see the prediction results for the first 100 rows, if you want to get more detailed statistics across the prediction results to evaluate your model's performance, you can use the Evaluate Model module.
12. Drag an Evaluate Model module on to the design surface beneath the Score Model module. Connect the output of the Score Model module to the leftmost input of the Evaluate Model module.

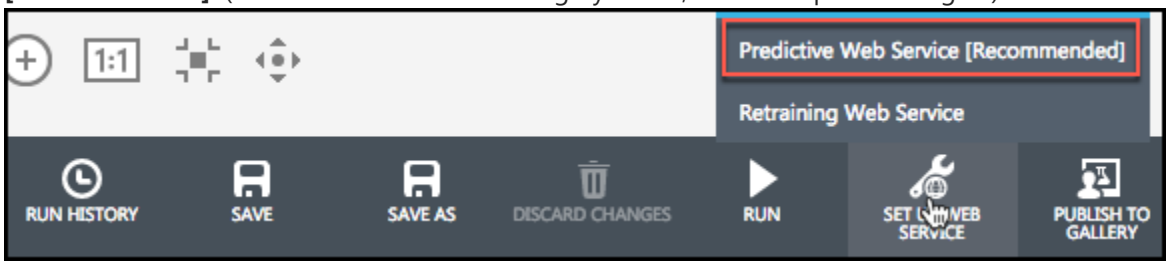


13. Run the experiment.
14. When the experiment is finished running, right-click the output of the Evaluate Model module and select **Visualize**. In this dialog box, you are presented with various ways to understand how your model is performing in the aggregate. While we will not cover how to interpret these results in detail, we can examine the ROC chart that tells us that at least our model (the blue curve) is performing better than random (the light gray straight line going from 0,0 to 1,1)—which is a good start for our first model!



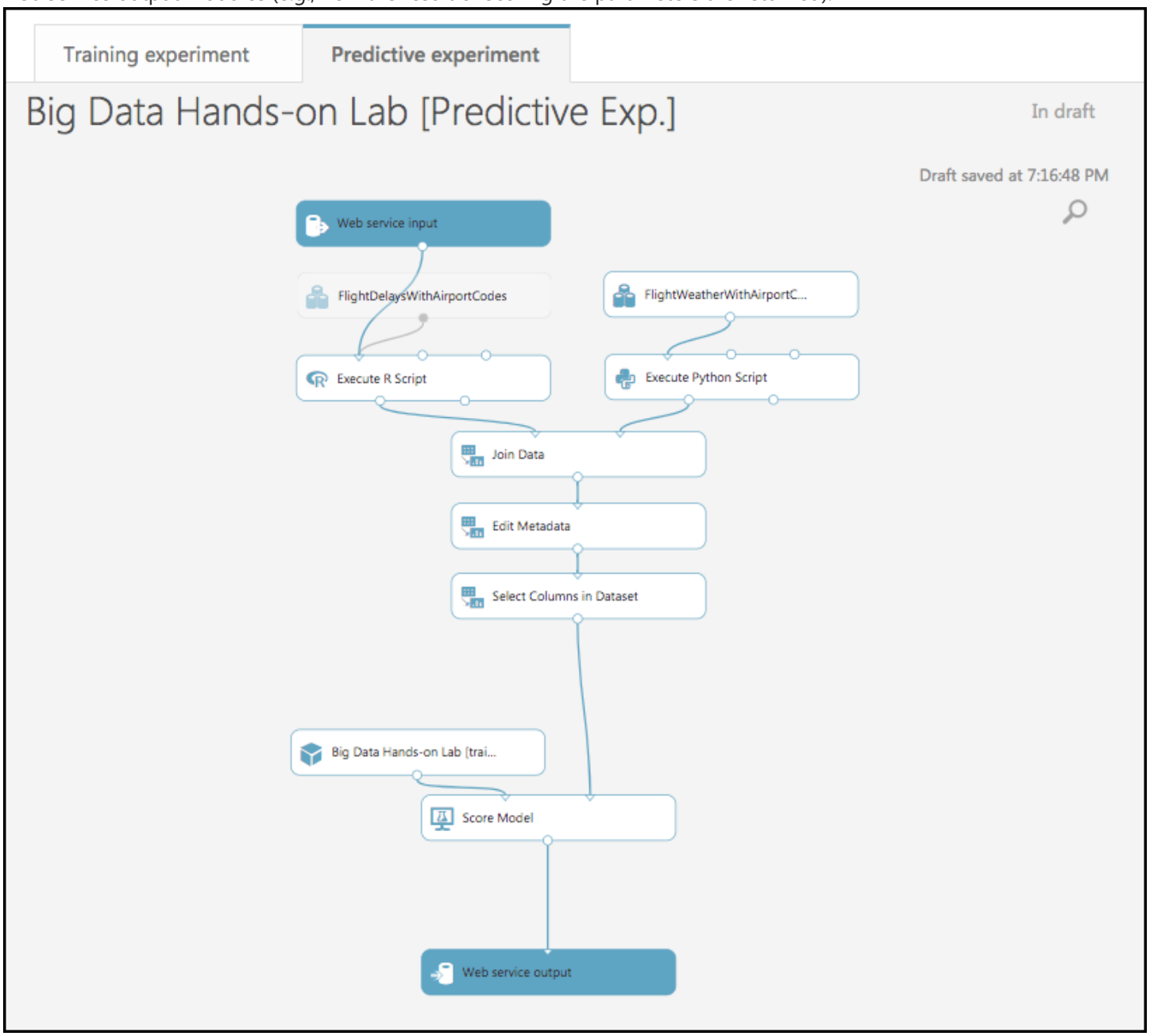
Task 8: Operationalize the experiment

1. Now that we have a functioning model, let us package it up into a predictive experiment that can be called as a web service.
2. In the command bar at the bottom, select **Set Up Web Service** and then select **Predictive Web Service [Recommended]**. (If Predictive Web Service is grayed out, run the experiment again)

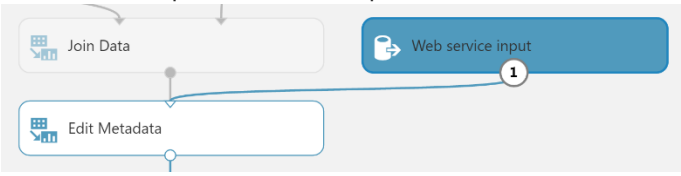


3. A copy of your training experiment is created, and a new tab labeled Predictive Experiment is added, which contains the trained model wrapped between web service input (e.g. the web service action you invoke with parameters) and

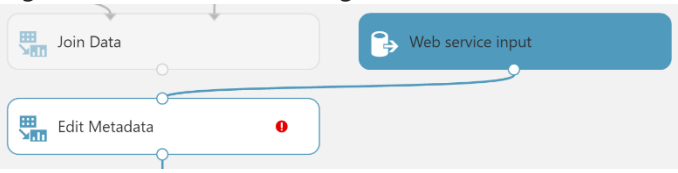
web service output modules (e.g., how the result of scoring the parameters are returned).



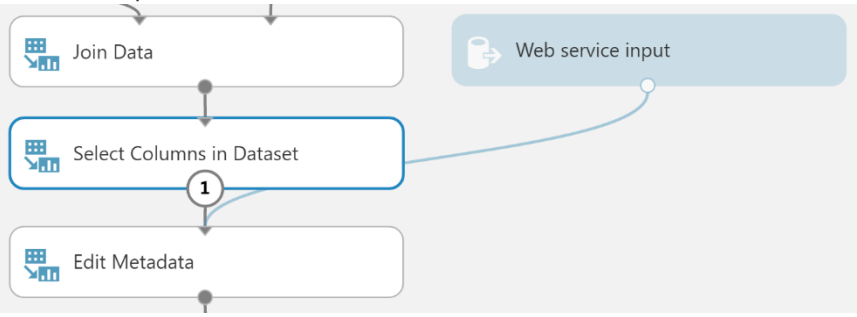
4. We will make some adjustments to the web service input and output modules to control the parameters we require and the results we return.
5. Move the Web Service Input module down, so it is to the right of the Join Data module. Connect the output of the Web service input module to input of the Edit Metadata module.



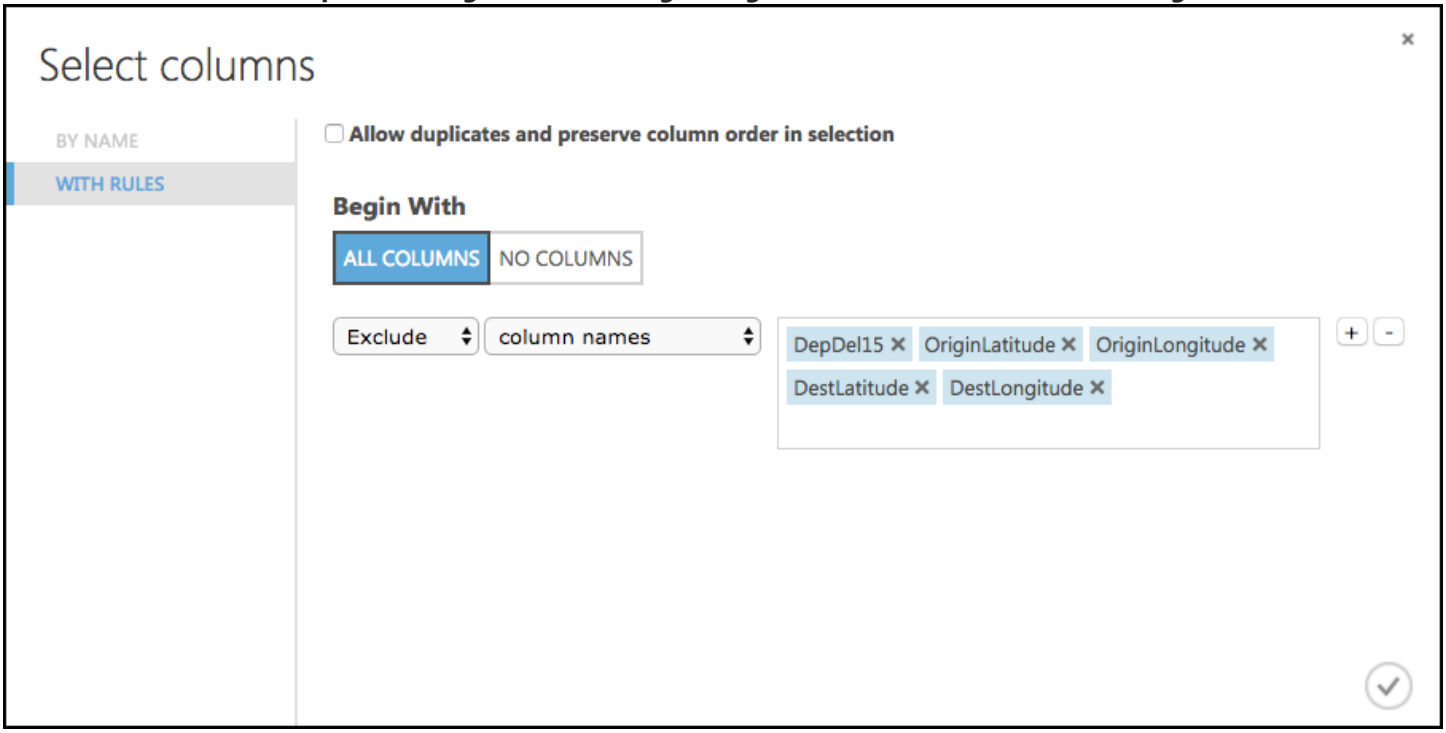
6. Right-click the line connecting the Join Data module and the Edit Metadata module and select **Delete**.



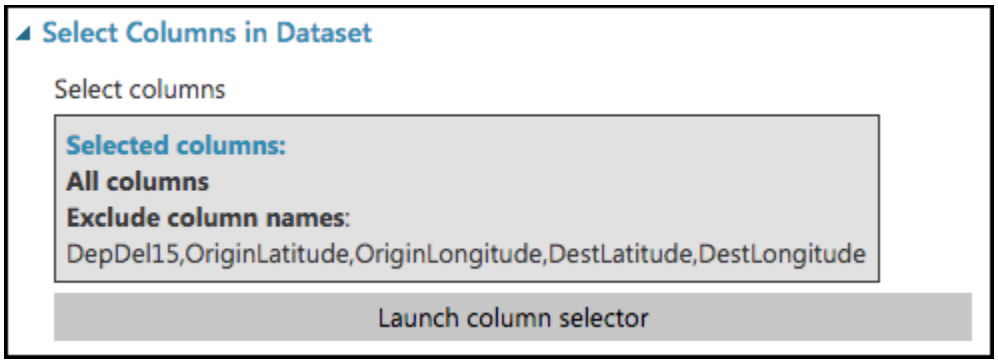
7. In between the Join Data and the Metadata Editor modules, drop a Select Columns in Dataset module. Connect the Join Data module's output to the Select Columns module's input, and the Select Columns output to the Edit Metadata module's input.



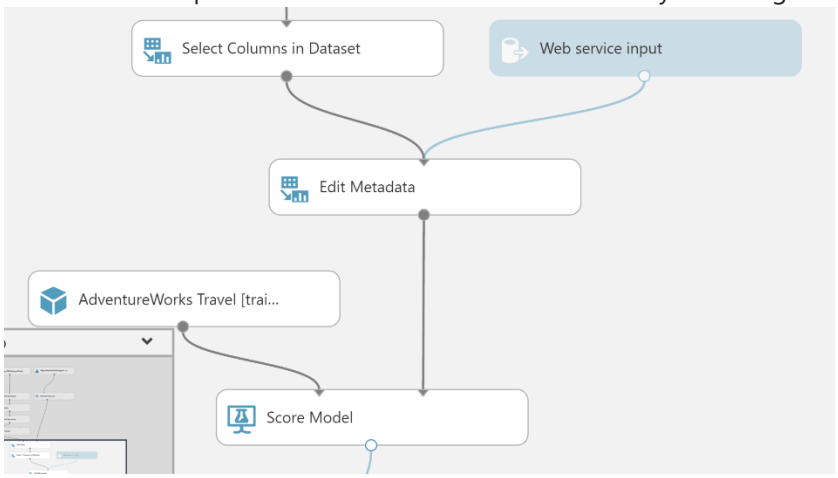
8. In the **Properties** panel for the Select Columns in Dataset module, set the Select columns to **All Columns**, and select **Exclude**. Enter columns **DepDel15, OriginLatitude, OriginLongitude, DestLatitude,** and **DestLongitude**.



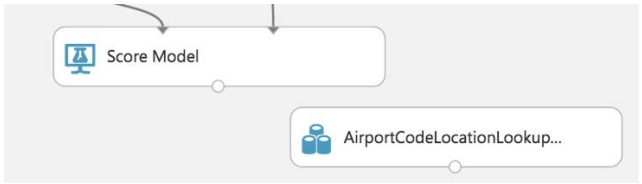
9. This configuration will update the web service metadata so that these columns do not appear as required input parameters for the web service.



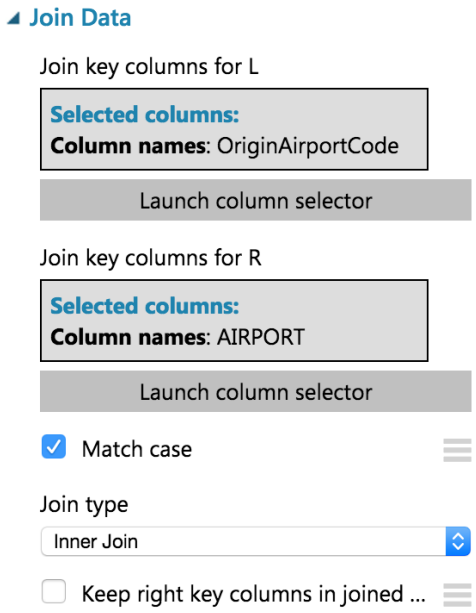
10. Select the Select Columns in Dataset module that comes after the Metadata Editor module, and delete it.
11. Connect the output of the Edit Metadata module directly to the right input of the Score Model module.



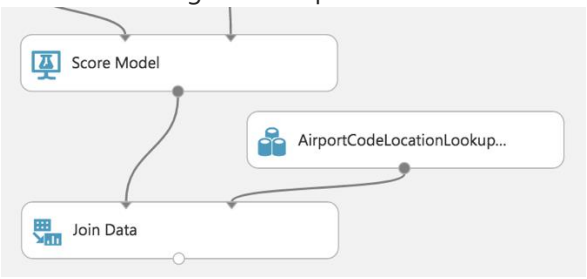
12. As we removed the latitude and longitude columns from the dataset to remove them as input to the web service, we must add them back in before we return the result so that the results can be easily visualized on a map.
13. To add these fields back, begin by deleting the line between the Score Model and Web service output.
14. Drag the AirportCodeLocationLookupClean dataset on to the design surface, positioning it below and to the right of the Score Model module.



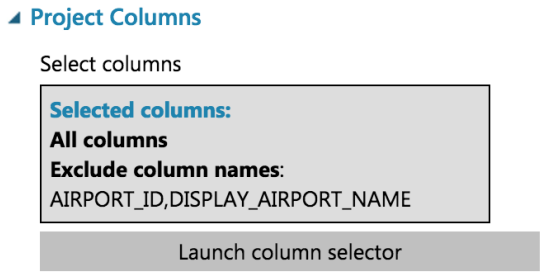
15. Add a Join Data module, and position it below and to the left of the AirportCodeLocationLookupClean module. In the **Properties** panel for the Join Data module, for the Join key columns for L set the selected columns to **OriginAirportCode**. For the Join key columns for R, set the Selected columns to **AIRPORT**. Uncheck Keep right key columns in joined table.



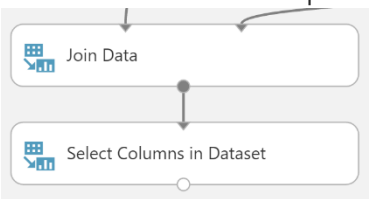
16. Connect the output of the Score Model module to the leftmost input of the Join Data module and the output of the dataset to the rightmost input of the Join Data module.



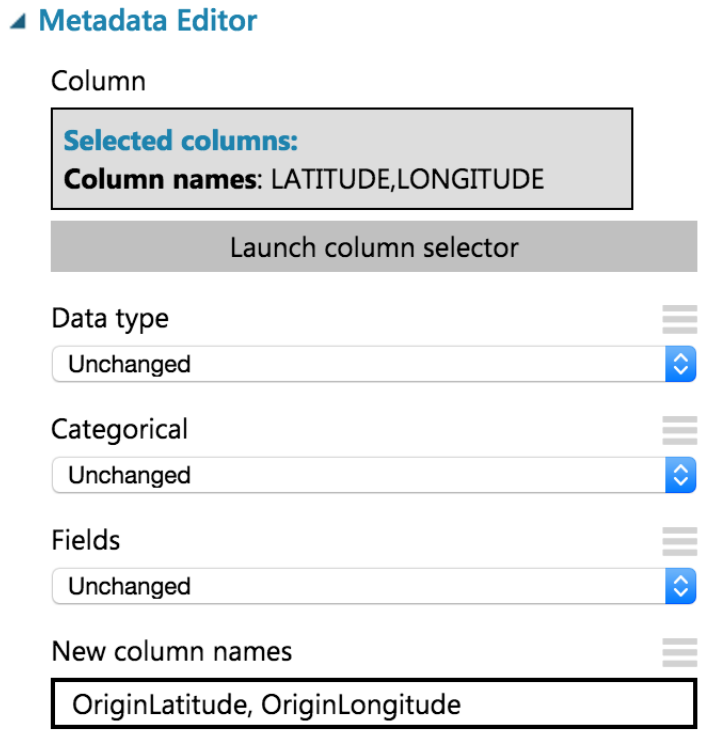
17. Add a Select Columns in Dataset module beneath the Join Data module. In the **Property** panel, begin with **All Columns**, and set the Selected columns to **Exclude** the columns: **AIRPORT_ID** and **DISPLAY_AIRPORT_NAME**.



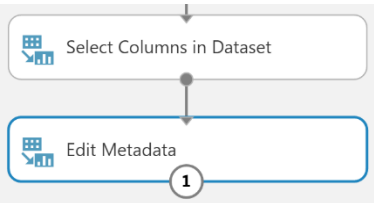
18. Connect the Join Data output to the input of the Select Columns in Dataset module.



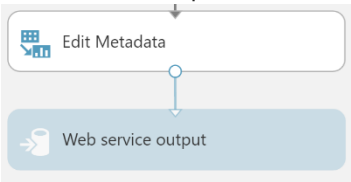
19. Add an Edit Metadata module. In the **Properties** panel for the Metadata Editor, use the column selector to set the Selected columns to LATITUDE and LONGITUDE. In the New column names enter: **OriginLatitude**, **OriginLongitude**.



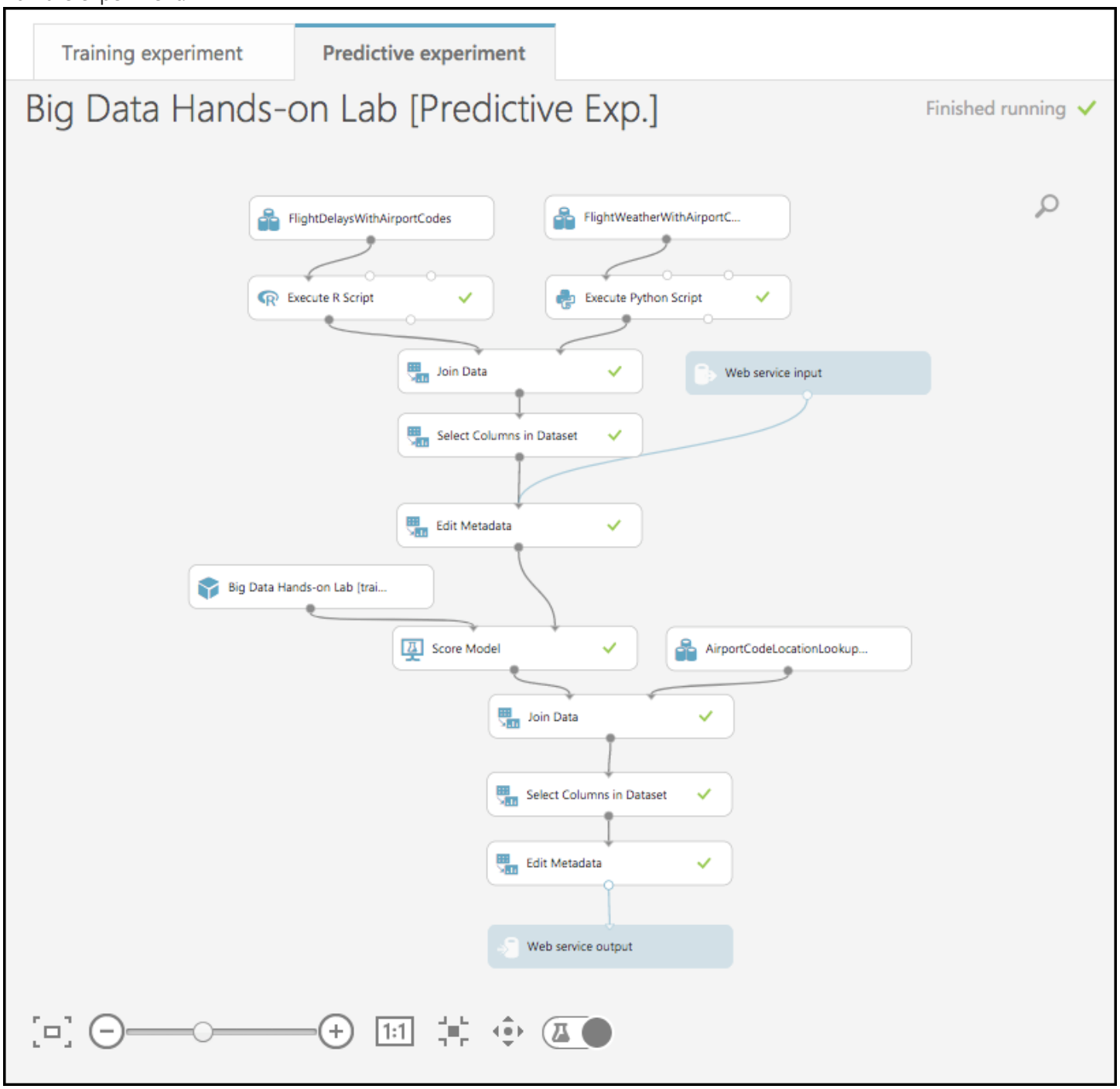
20. Connect the output of the Select Columns in Dataset module to the input of the Edit Metadata module.



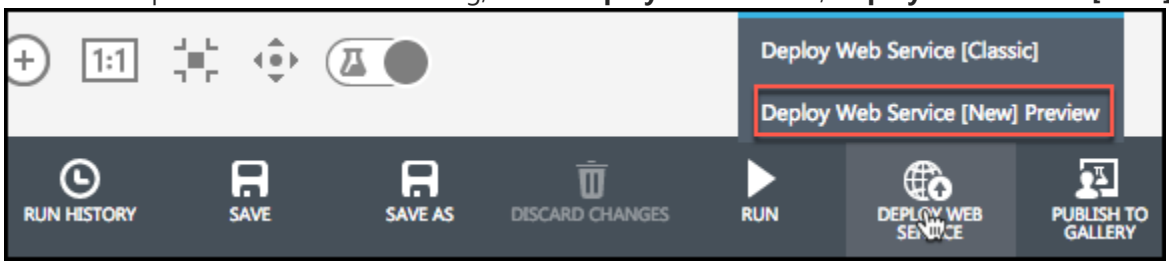
21. Connect the output of the Edit Metadata to the input of the web service output module.



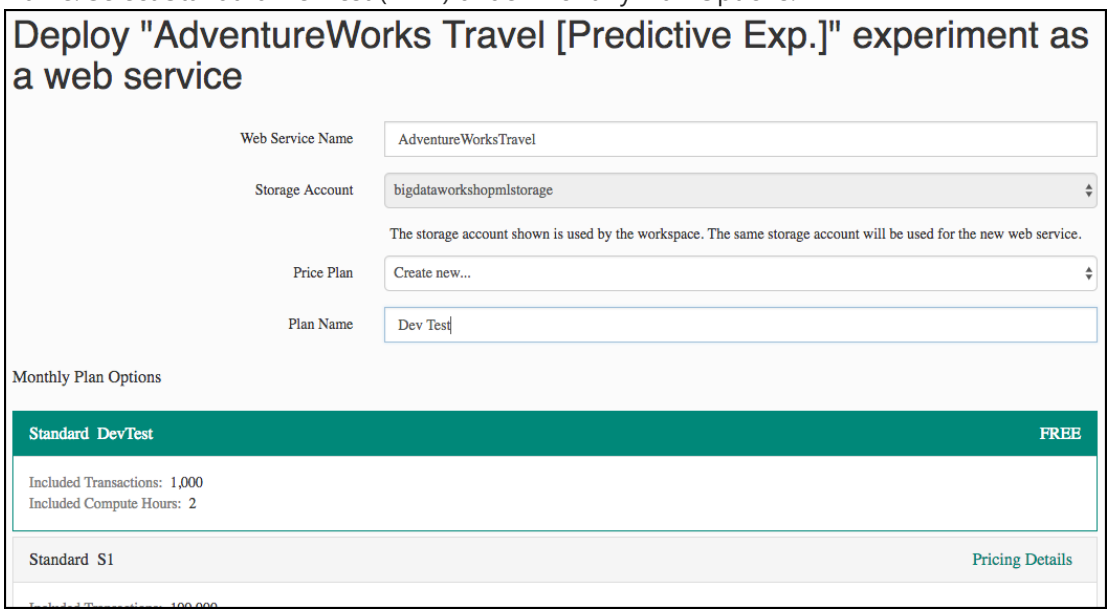
22. Run the experiment.



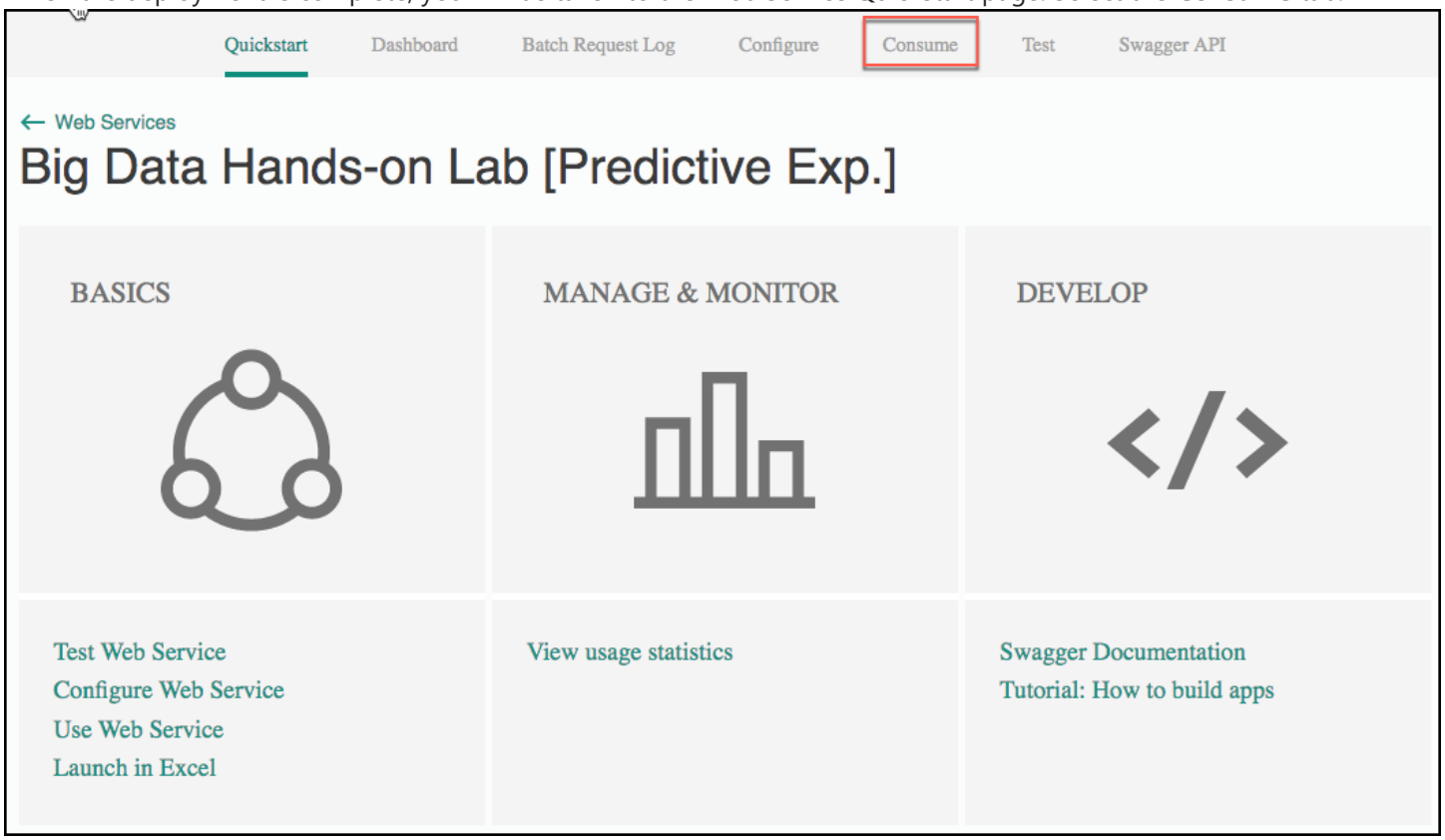
23. When the experiment is finished running, select **Deploy Web Service**, **Deploy Web Service [NEW] Preview**.



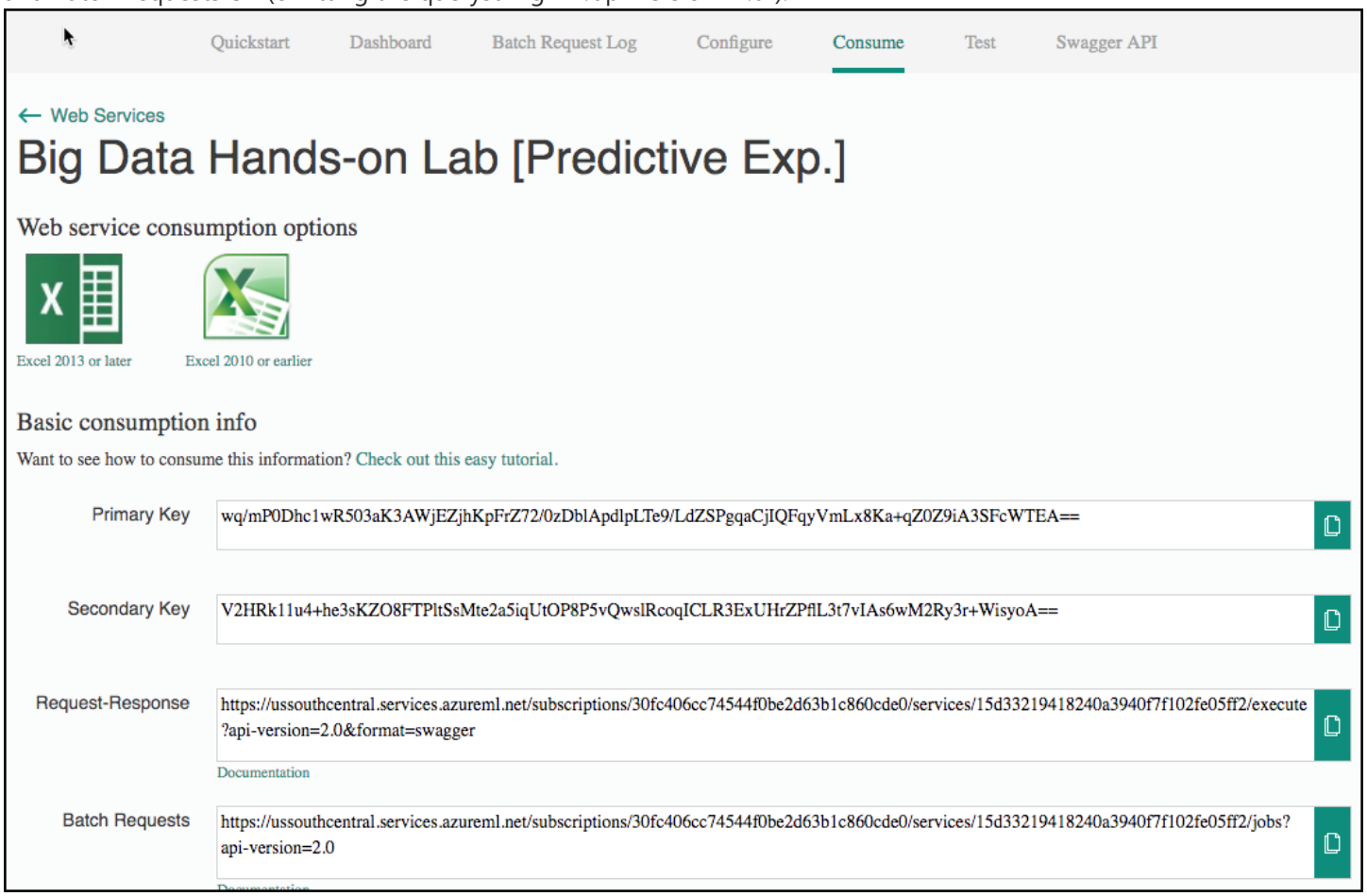
24. On the Deploy experiment page, select **Create New...** in the Price Plan drop down, and enter **Dev Test** as the Plan Name. Select Standard DevTest (FREE) under Monthly Plan Options.



25. Select **Deploy**.
26. When the deployment is complete, you will be taken to the Web Service Quickstart page. Select the **Consume** tab.



27. Leave the Consume page open for reference during Exercise 4, Task 1. At that point, you need to copy the Primary Key and Batch Requests Uri (omitting the querystring – "?api-version=2.0").



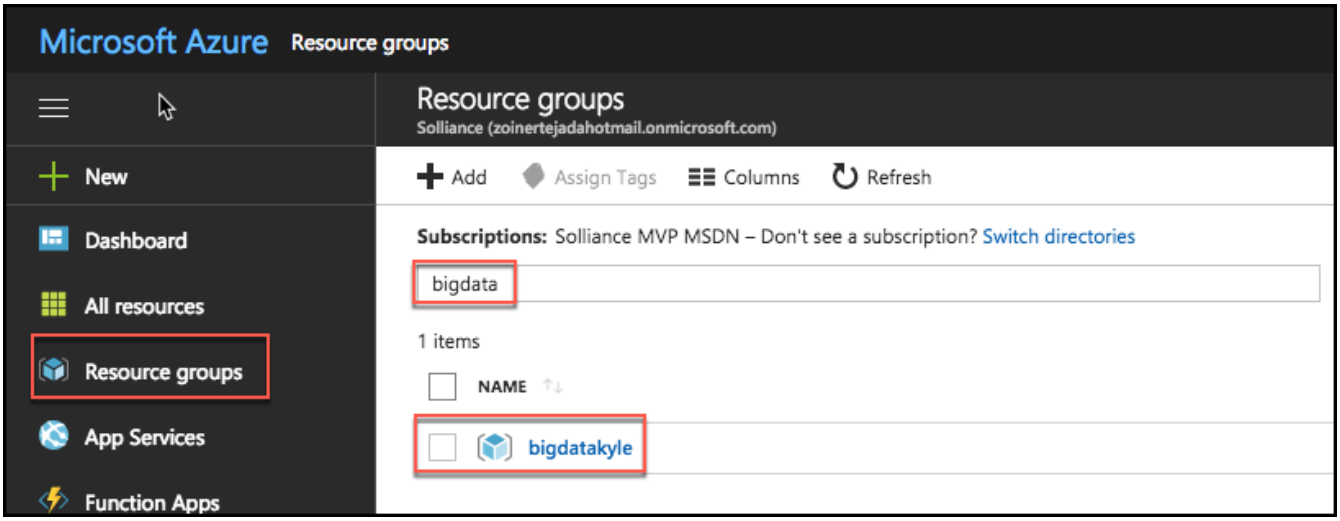
Exercise 2: Setup Azure Data Factory

Duration: 20 minutes

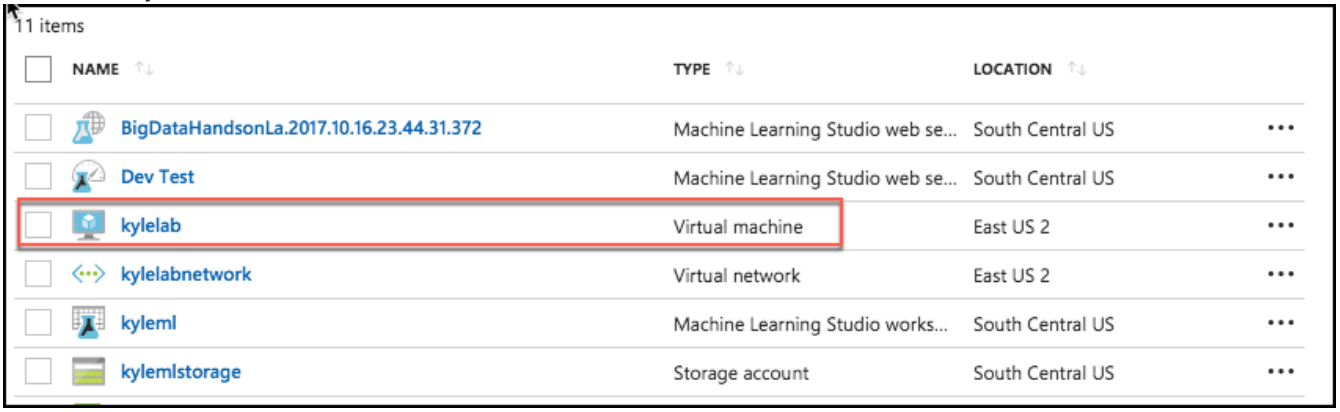
In this exercise, attendees will create a baseline environment for Azure Data Factory development for further operationalization of data movement and processing. You will create a Data Factory service, and then install the Data Management Gateway which is the agent that facilitates data movement from on-premises to Microsoft Azure.

Task 1: Connect to the lab VM

1. NOTE: If you are already, connected to your Lab VM, skip to Task 2.
2. From the left side menu in the Azure portal, click on **Resource groups**, then enter your resource group name into the filter box, and select it from the list.



3. Next, select your lab virtual machine from the list.



4. On your Lab VM blade, select **Connect** from the top menu.

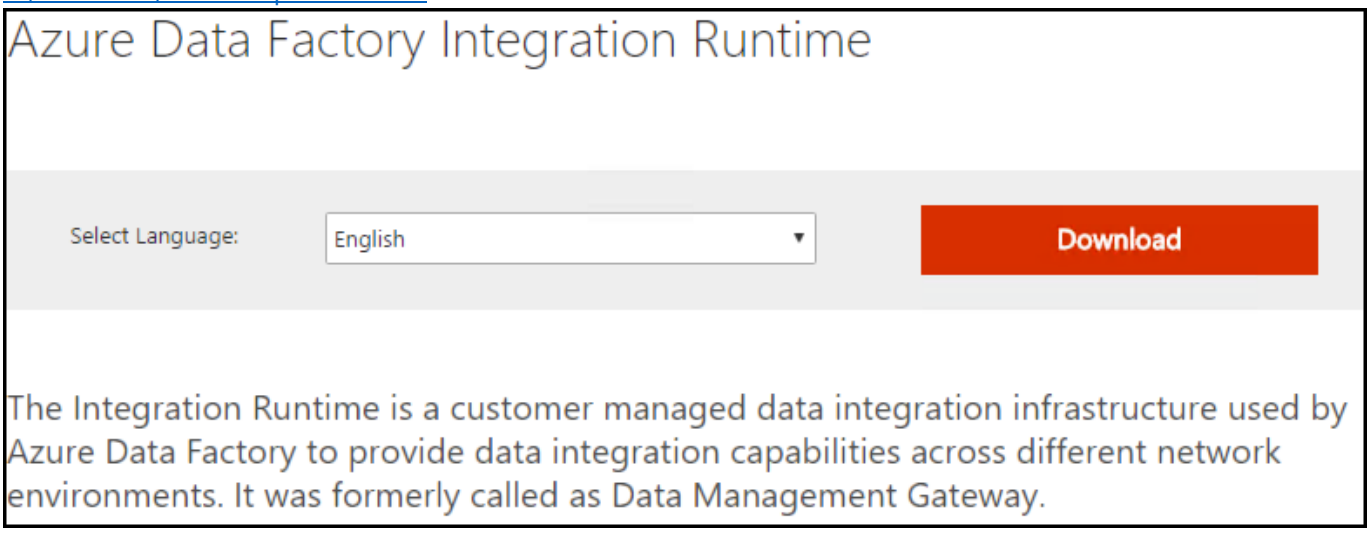
5. Download and open the RDP file.
6. Select Connect, and enter the following credentials:
   - User name: demouser

- Password: Password.1!!

Task 2: Download and stage data to be processed

1. Once you have logged into the Lab VM, open a web browser. A shortcut for Chrome is on the desktop, and Internet Explorer can be accessed from the Start screen.
2. Download the AdventureWorks sample data from http://bit.ly/2zi4Sqa.
3. Extract it to a new folder called **C:\Data**.

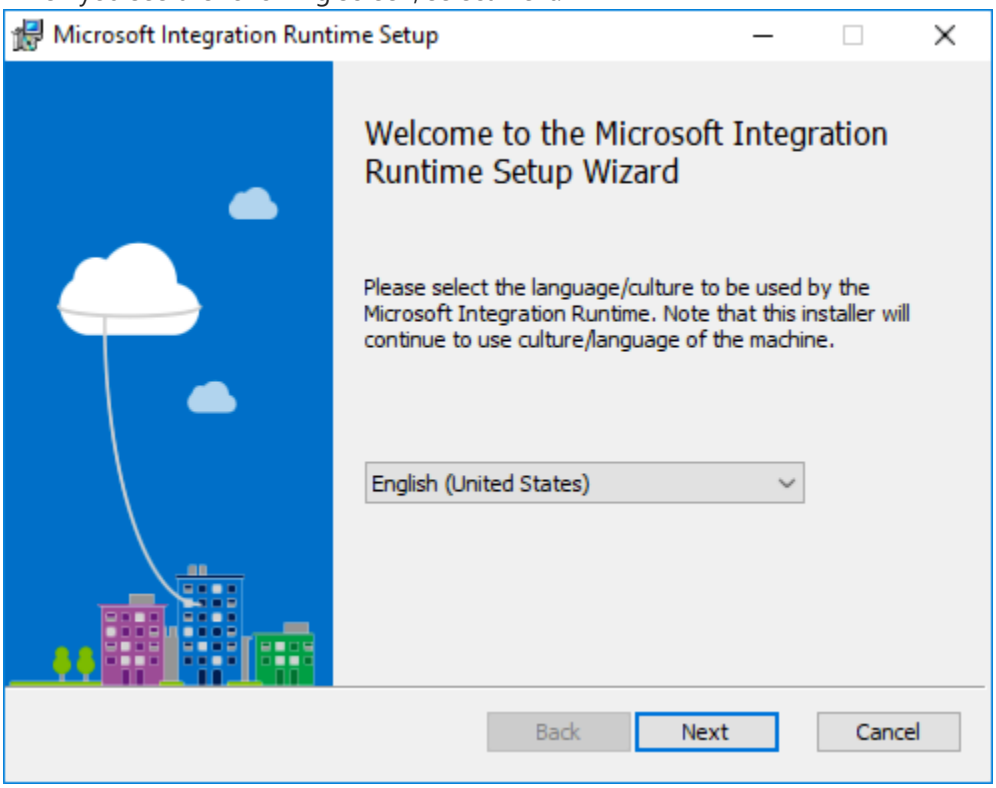
Task 3: Install and configure Azure Data Factory Integration Runtime on the Lab VM

1. To download the latest version of Azure Data Factory Integration Runtime, go to https://www.microsoft.com/en-us/download/details.aspx?id=39717
2. 

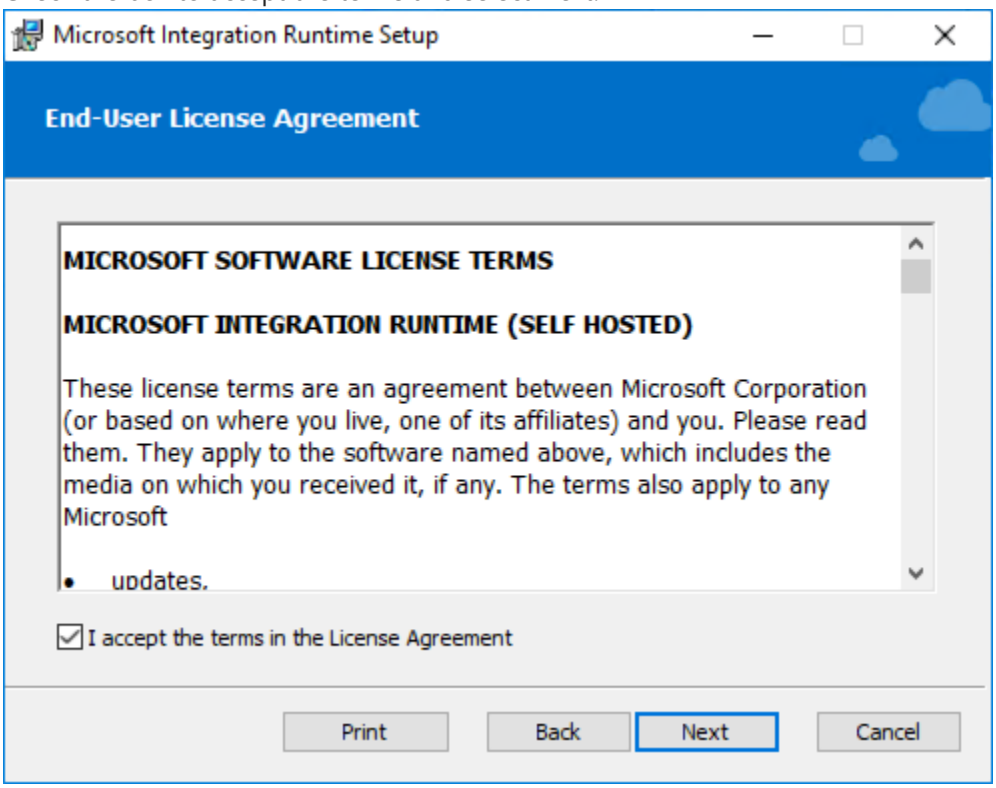
3. Select Download, then choose the download you want from the next screen.

4. Run the installer, once downloaded.

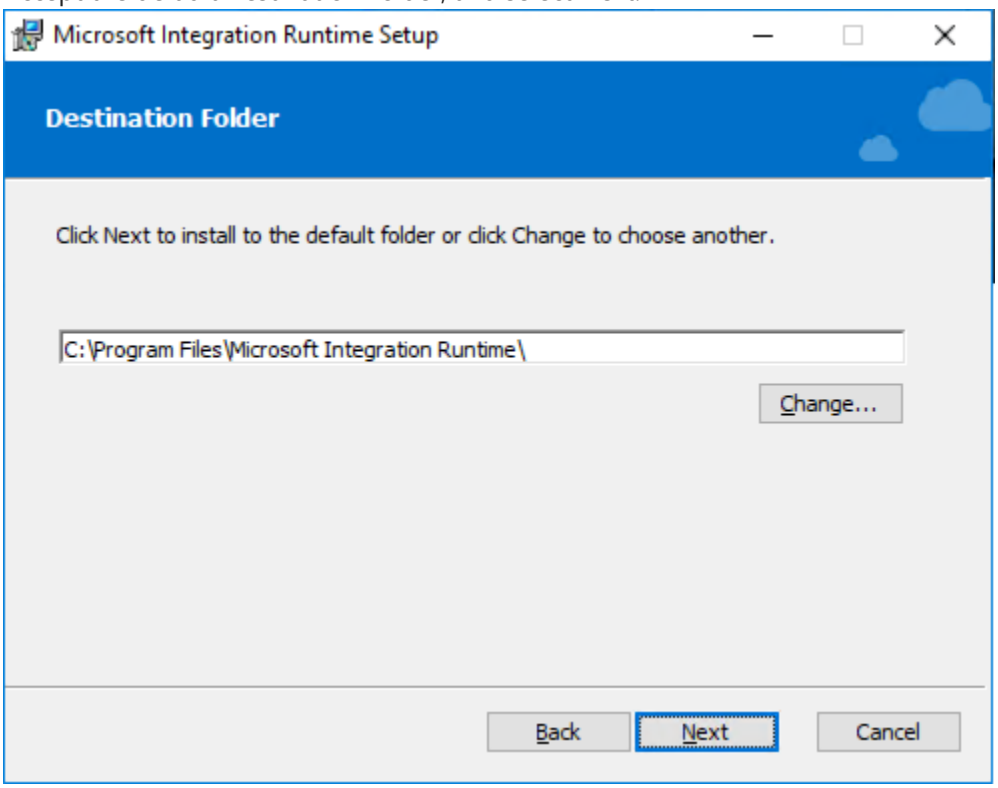
5. When you see the following screen, select Next.



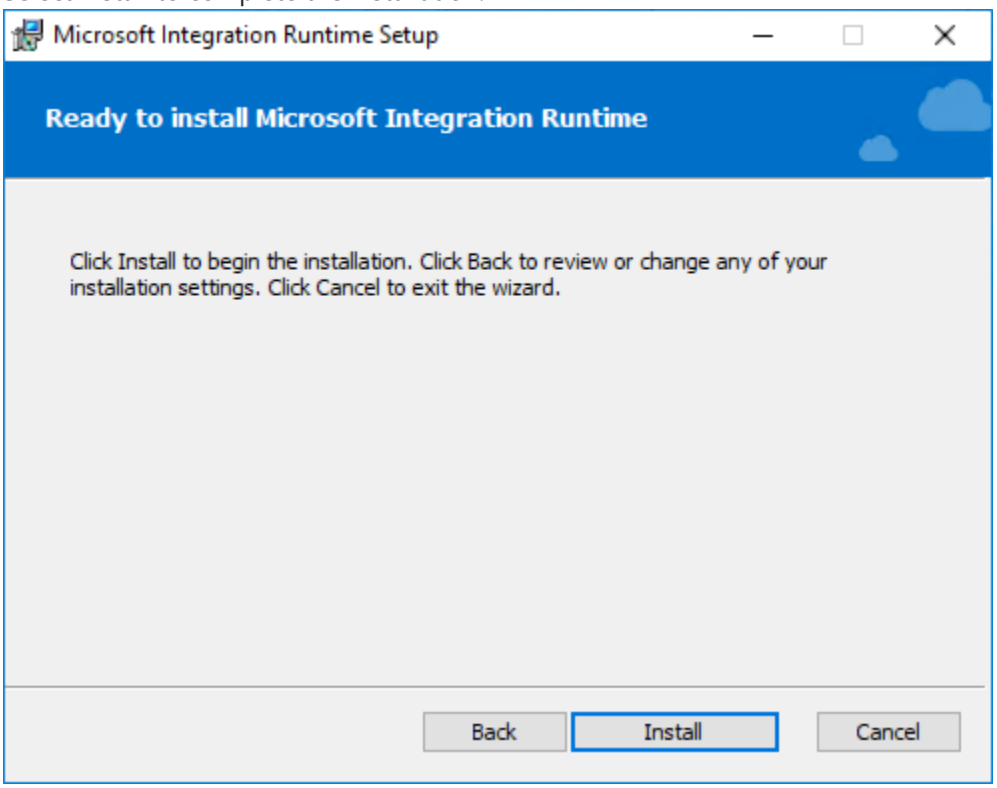
6. Check the box to accept the terms and select Next.



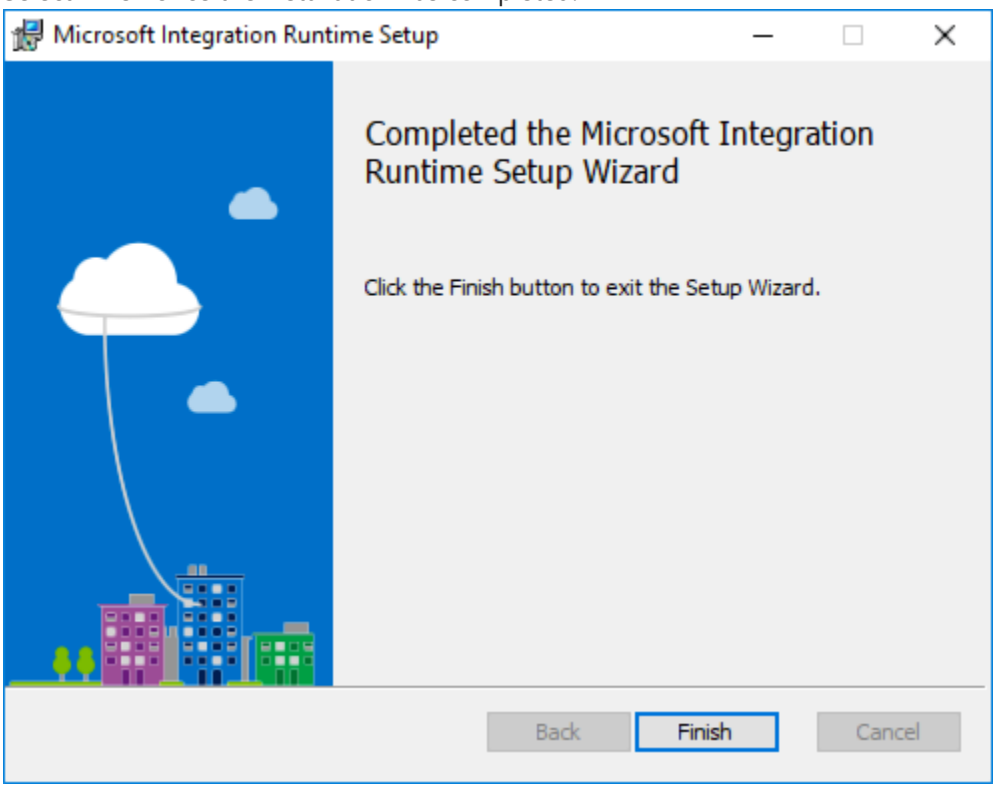
7. Accept the default Destination Folder, and select Next.



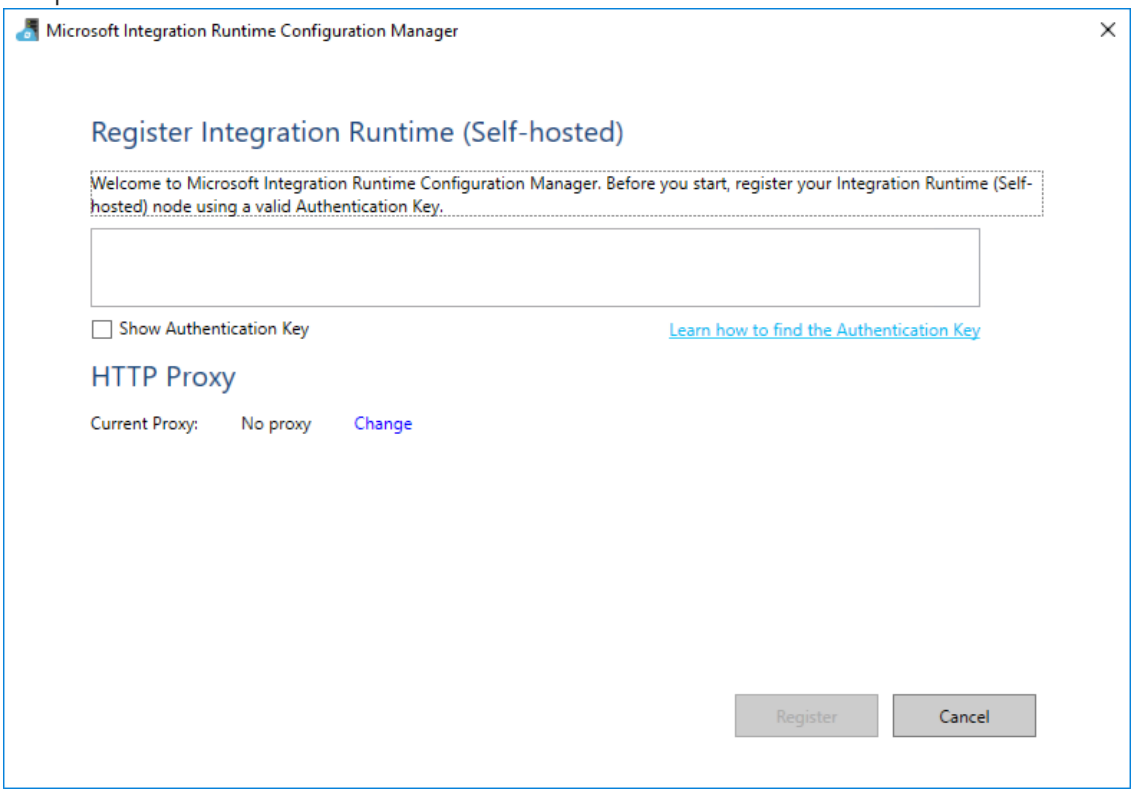
8. Select Install to complete the installation.



9. Select Finish once the installation has completed.

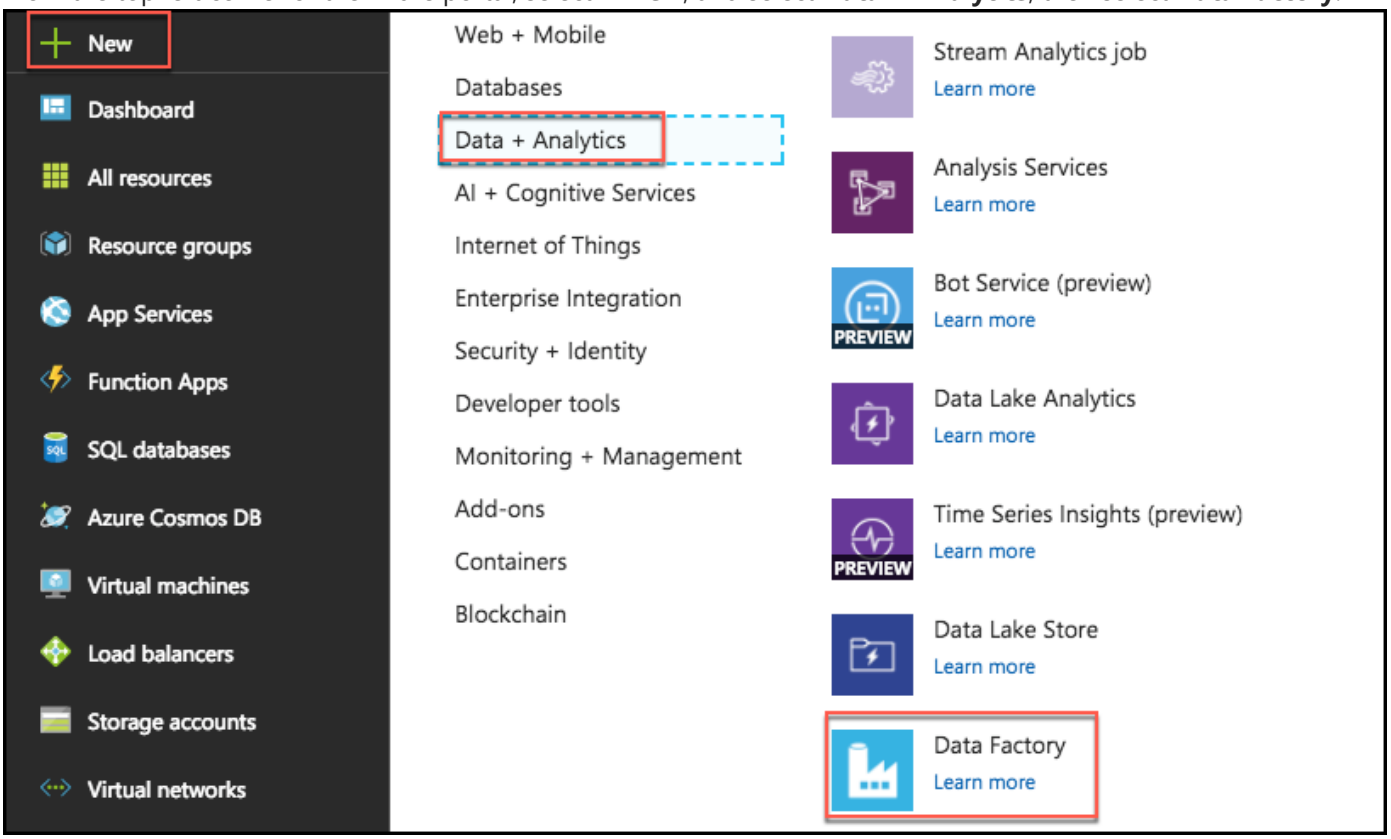


10. After clicking Finish, the following screen will appear. Keep in open for now. We will come back to this screen once we have provisioned the Data Factory in Azure, and obtain the gateway key so we can connect Data Factory to this "on-premises" server.

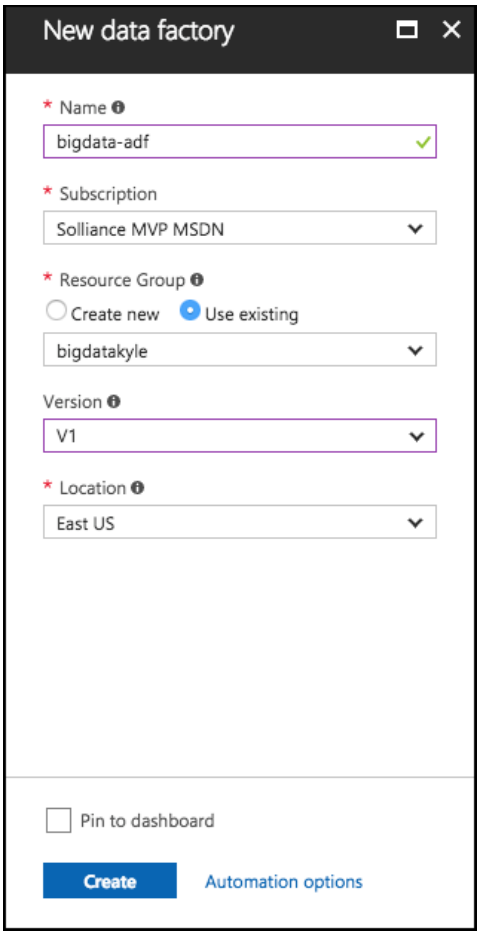


Task 4: Create an Azure data factory

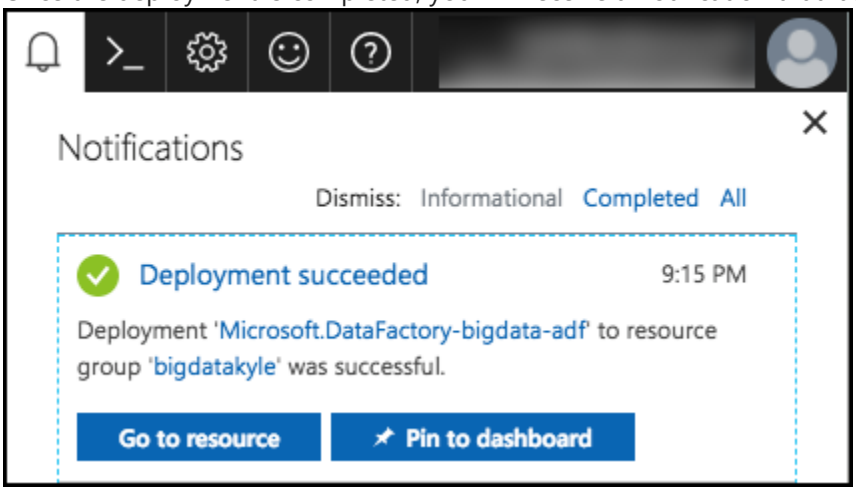
1. Launch a new browser window, and navigate to the Azure portal (https://portal.azure.com). Once prompted, log in with your Microsoft Azure credentials. If prompted, choose whether your account is an organization account or a Microsoft account. This will be based on which account was used to provision your Azure subscription that is being used for this lab.
    - **Note**: You may need to launch an InPrivate/Incognito session in your browser if you have multiple Microsoft accounts.
2. From the top left corner of the Azure portal, select **+ New**, and select **Data + Analytics**, then select **Data Factory**.



3. On the New data factory blade, enter the following:
    - Name: Provide a name, such as bigdata-adf
    - Subscription: Select your subscription
    - Resource Group: Choose Use existing, and select the Resource Group you created when deploying the lab prerequisites
    - Version: Select V1
    - Location: Select one of the available locations from the list nearest the one used by your Resource Group
    - Select **Create**

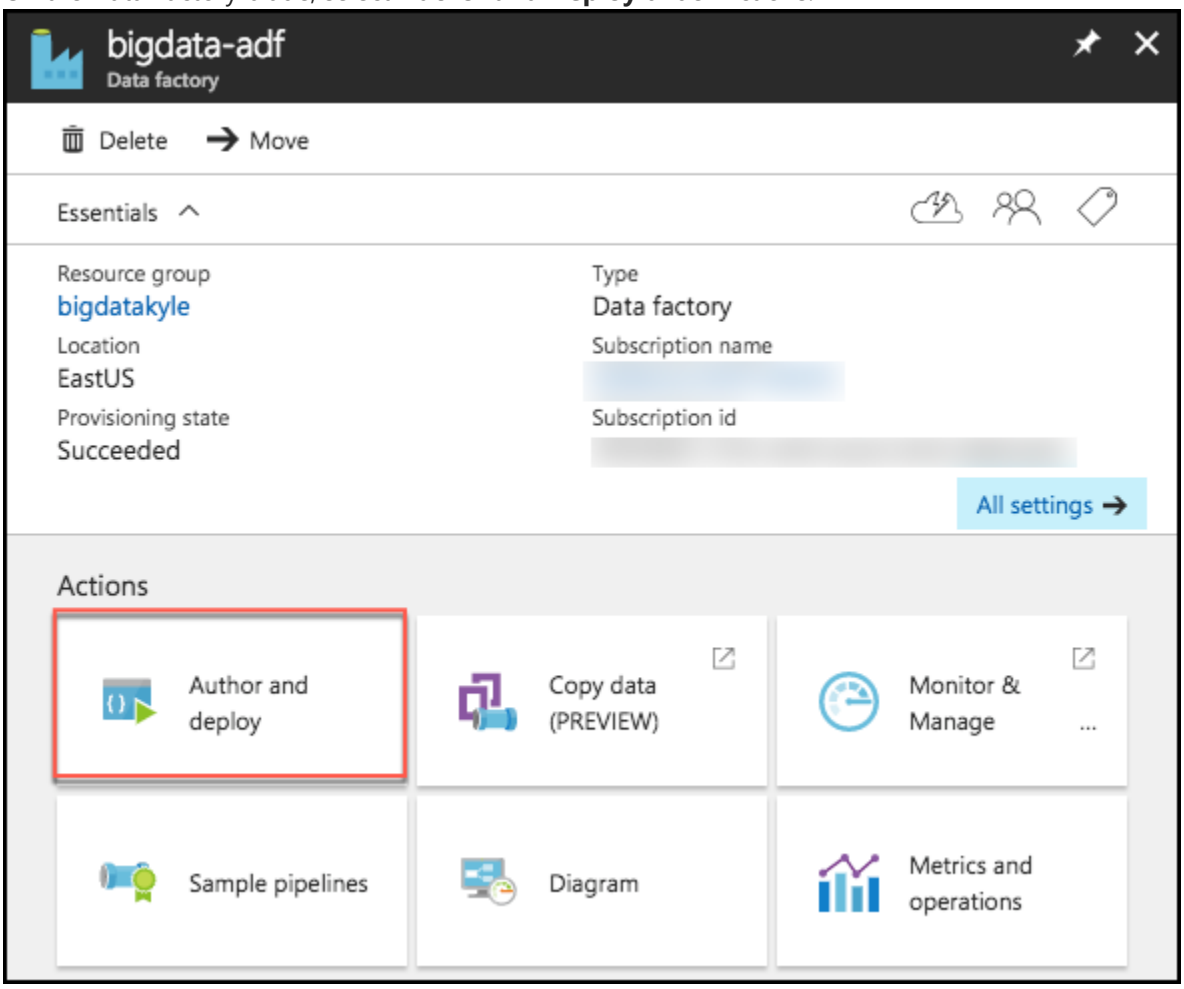


4. The ADF deployment will take several minutes.
5. Once the deployment is completed, you will receive a notification that it succeeded.

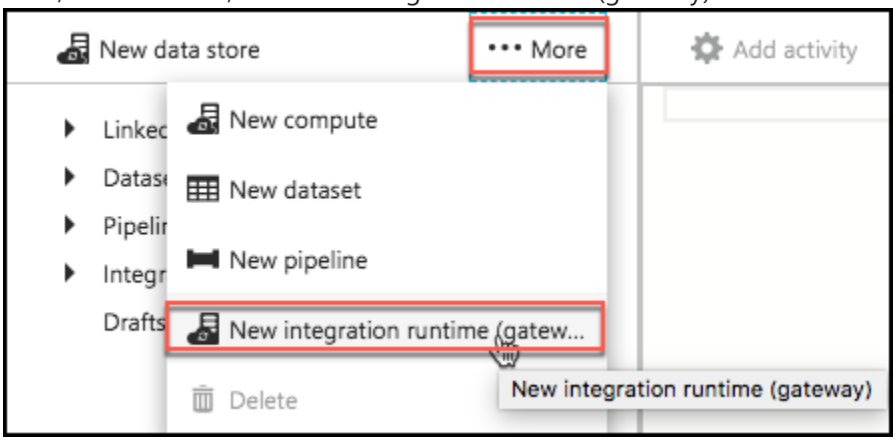


6. Select the Go to resource button, to navigate to the newly created Data Factory.

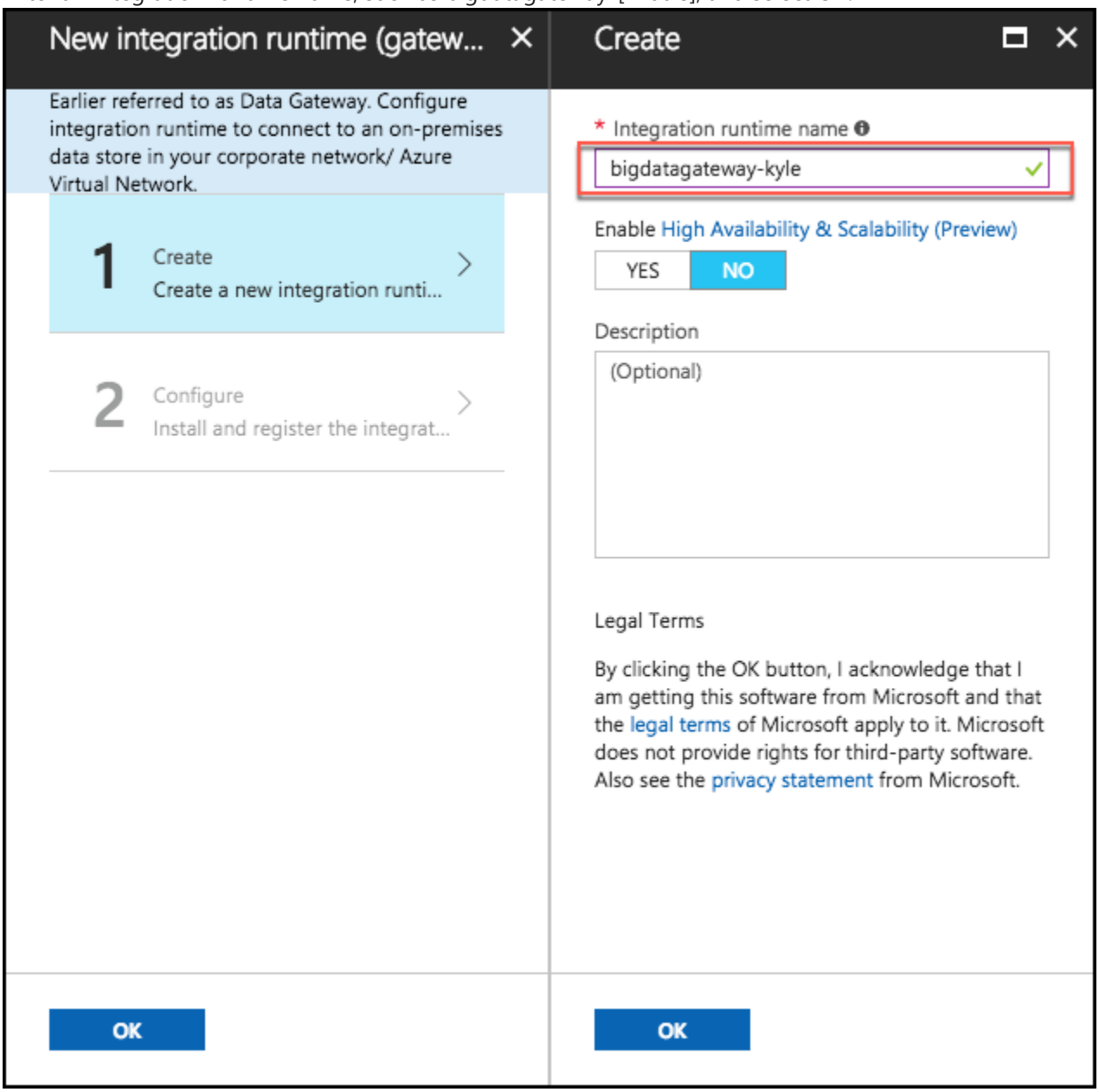
7. On the Data Factory blade, select **Author and Deploy** under Actions.



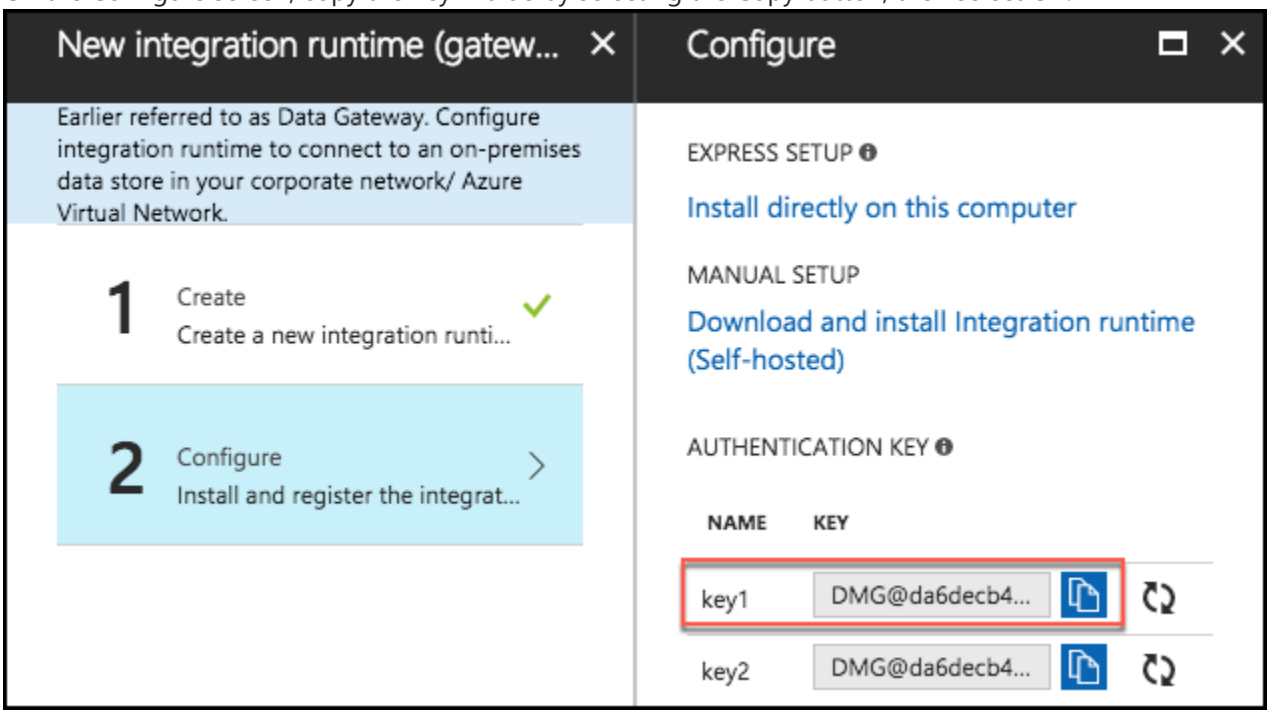
8. Next, select …More, then New integration runtime (gateway).



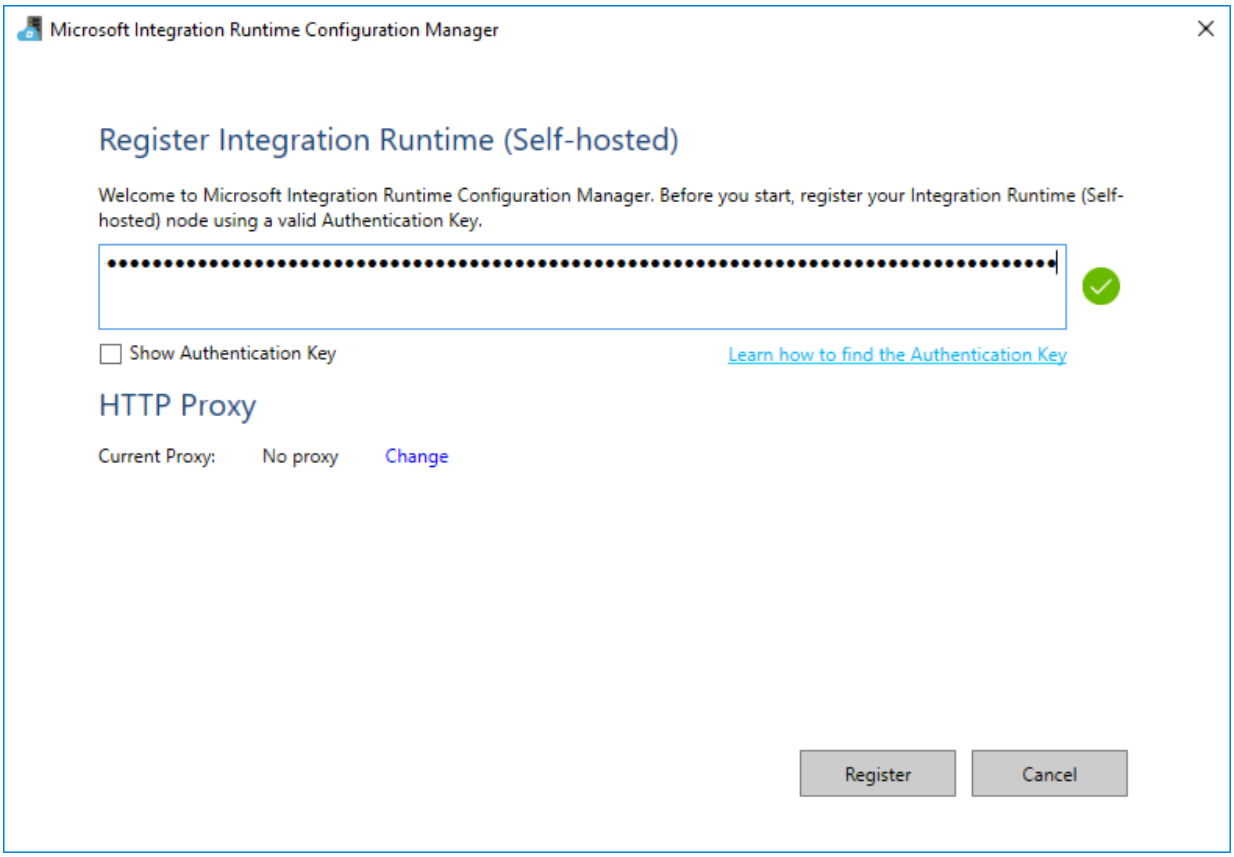
9. Enter an Integration runtime name, such as bigdatagateway-[initials], and select OK.



10. On the Configure screen, copy the key1 value by selecting the Copy button, then select OK.

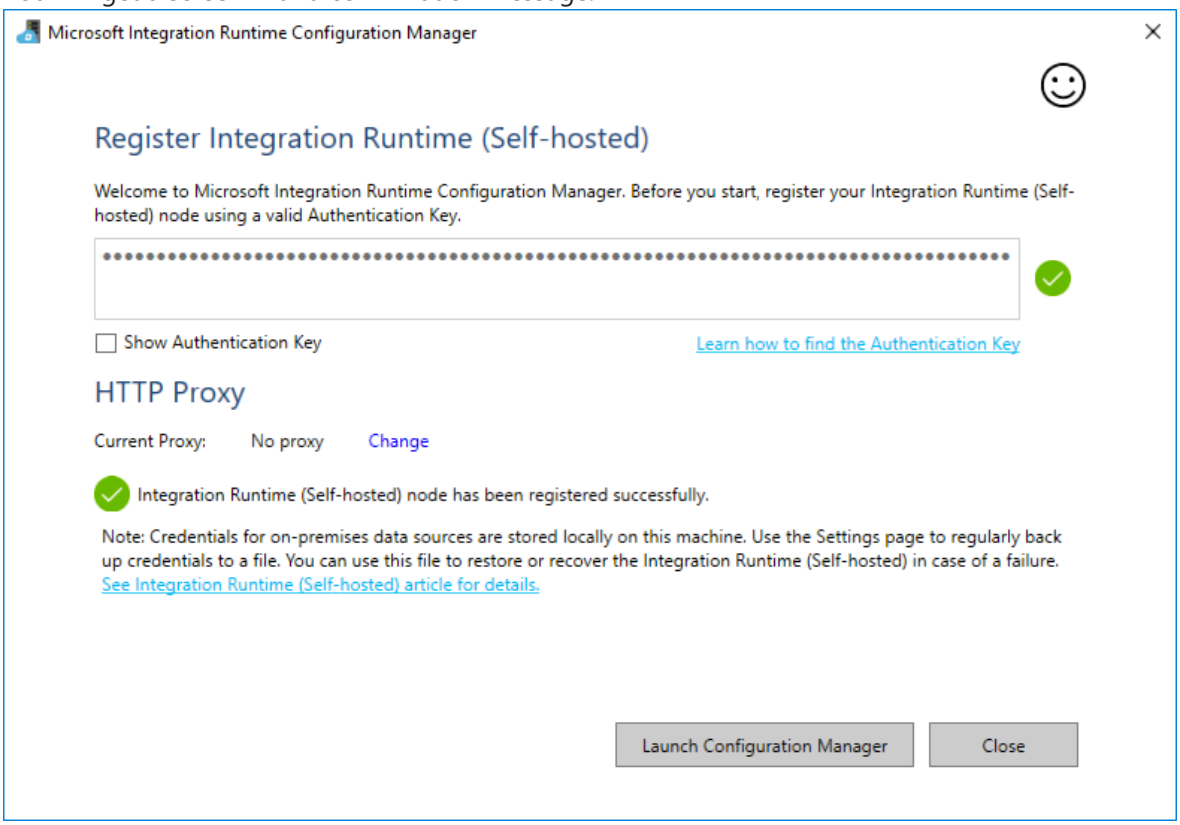


11. *Don't close the current screen or browser session.*
12. Go back to the Remote Deskop session of the Lab VM.
13. Paste the **key1** value into the box in the middle of the Microsoft Integration Runtime Configuration Manager screen.

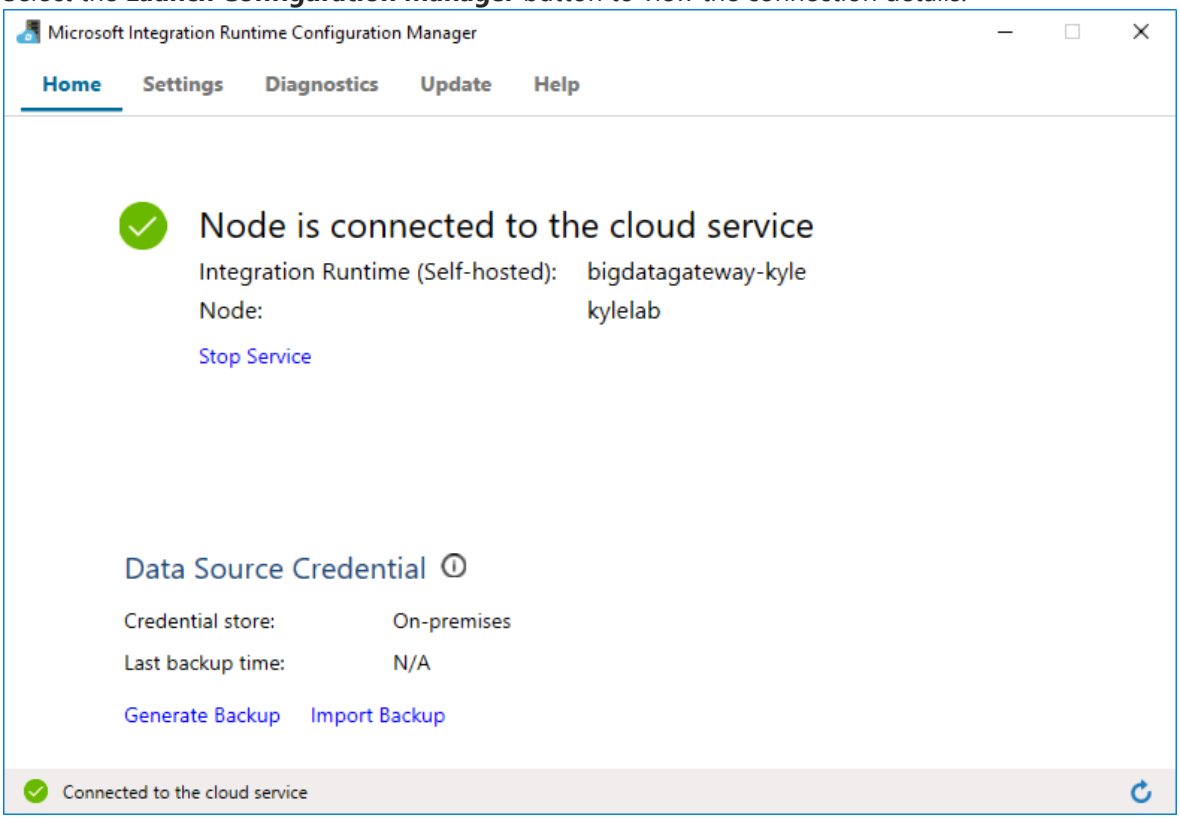


14. Select **Register**.

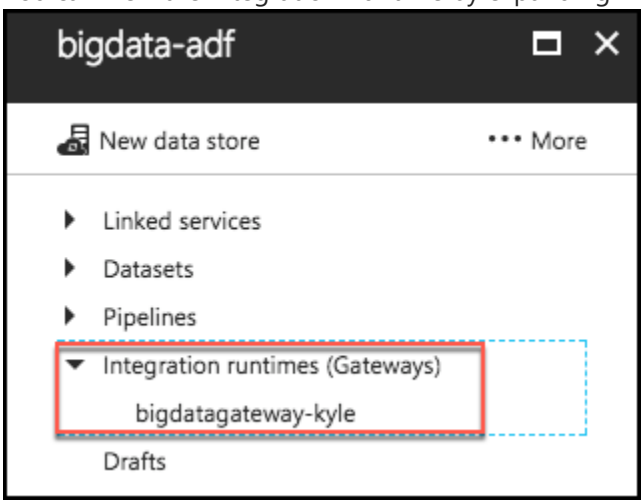
15. It will take a minute or two to register. If it takes more than a couple of minutes, and the screen does not respond or returns an error message, close the screen by clicking the **Cancel** button.
16. You will get a screen with a confirmation message.



17. Select the **Launch Configuration Manager** button to view the connection details.



18. You can now return to the Azure portal, and click **OK** twice to complete the Integration Runtime setup.
19. You can view the Integration Runtime by expanding Integration runtimes on the Author and Deploy blade.



20. Close the Author and Deploy blade, to return to the the Azure Data Factory blade. Leave this open for the next exercise.

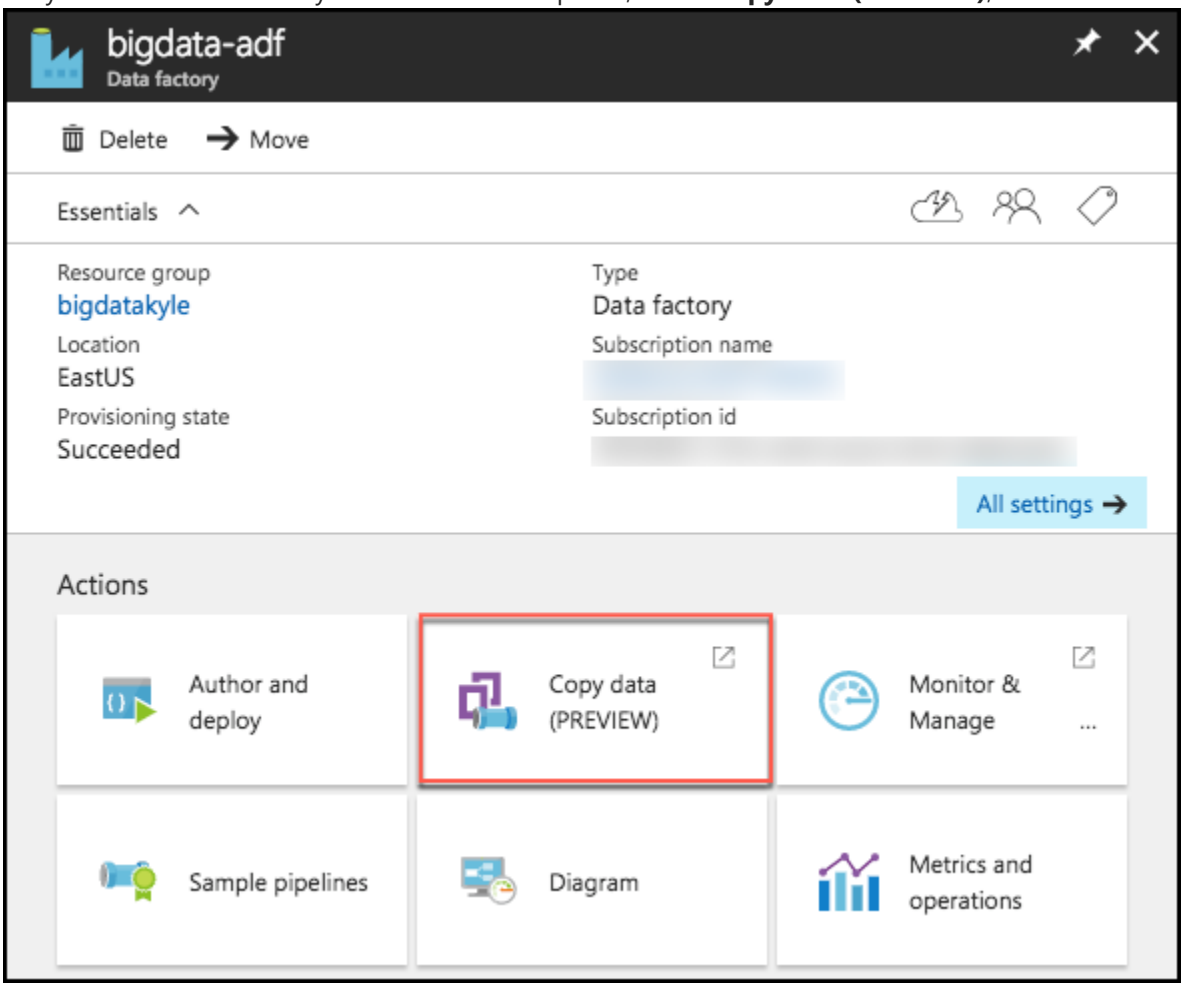
Exercise 3: Develop a data factory pipeline for data movement

Duration: 20 minutes

In this exercise, you will create an Azure Data Factory pipeline to copy data (.CSV file) from an on-premises server (Lab VM) to Azure Blob Storage. The goal of the exercise is to demonstrate data movement from an on-premises location to Azure Storage (via the Integration Runtime). You will see how assets are created, deployed, executed, and monitored.

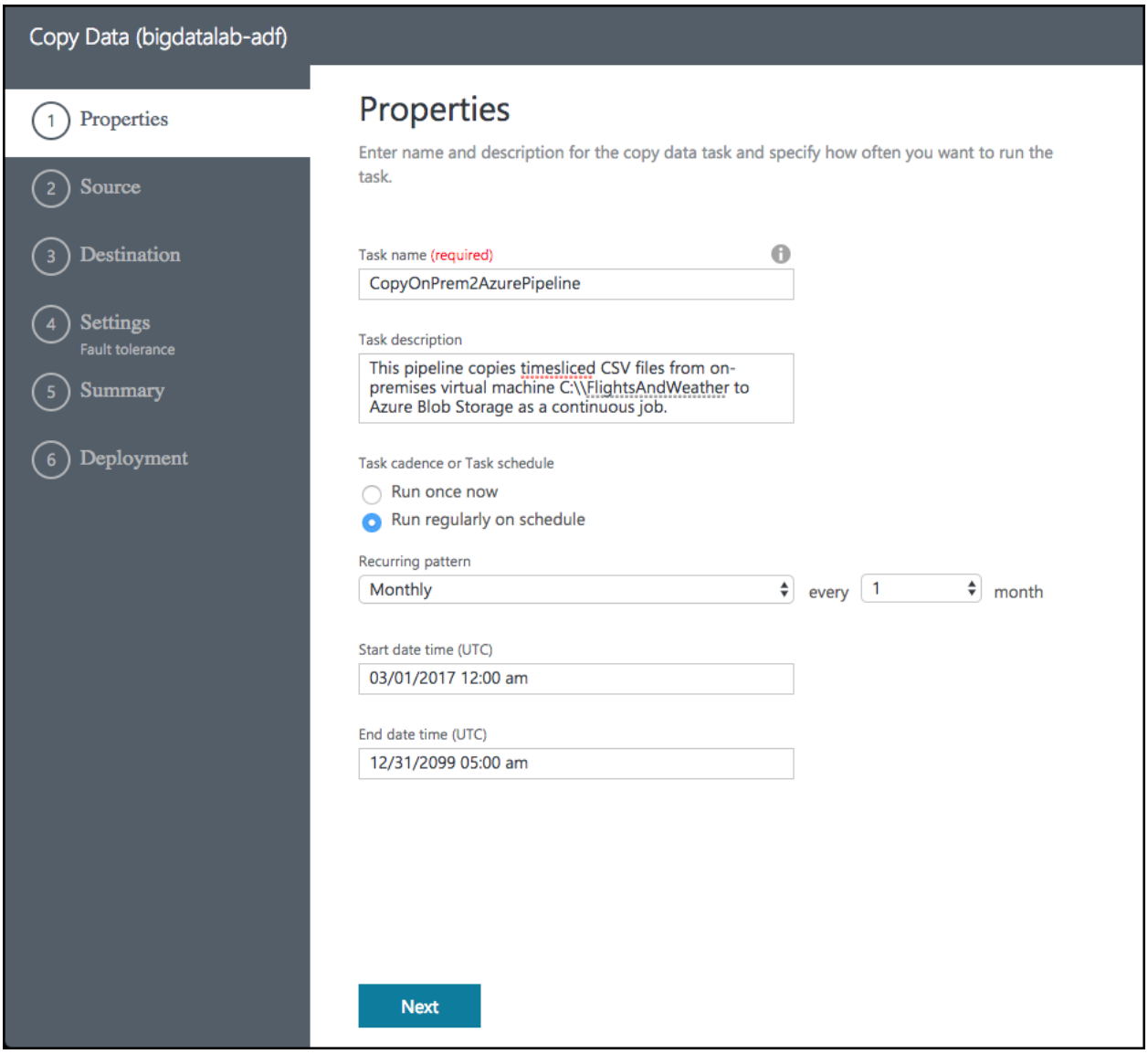
Task 1: Create copy pipeline using the Copy Data Wizard

1. On your Azure Data Factory blade in the Azure portal, select **Copy Data (PREVIEW)**, under Actions.

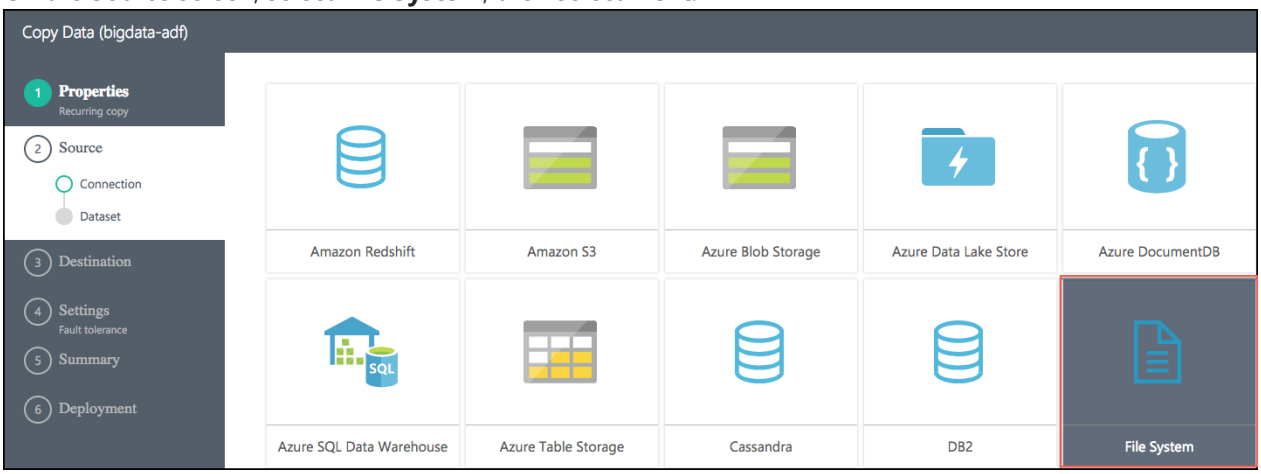


2. This will launch a new browser window. Log in with the same credentials you used to create the Data Factory.
3. In the new browser window, enter the following:
    a. Task name: Enter "CopyOnPrem2AzurePipeline"
    b. Task description: (Optional) Enter a description, such as "This pipeline copies timesliced CSV files from on-premises virtual machine C:\\FlightsAndWeather to Azure Blob Storage as a continuous job."
    c. Task cadence (or) Task schedule: Select Run regularly on schedule.
    d. Recurring pattern: Select Monthly, and every 1 month.
    e. Start date time (UTC): Set to 03/01/2017 12:00 am
    f. End date time (UTC): Set to 12/31/2099 11:59 pm

g. Select **Next**



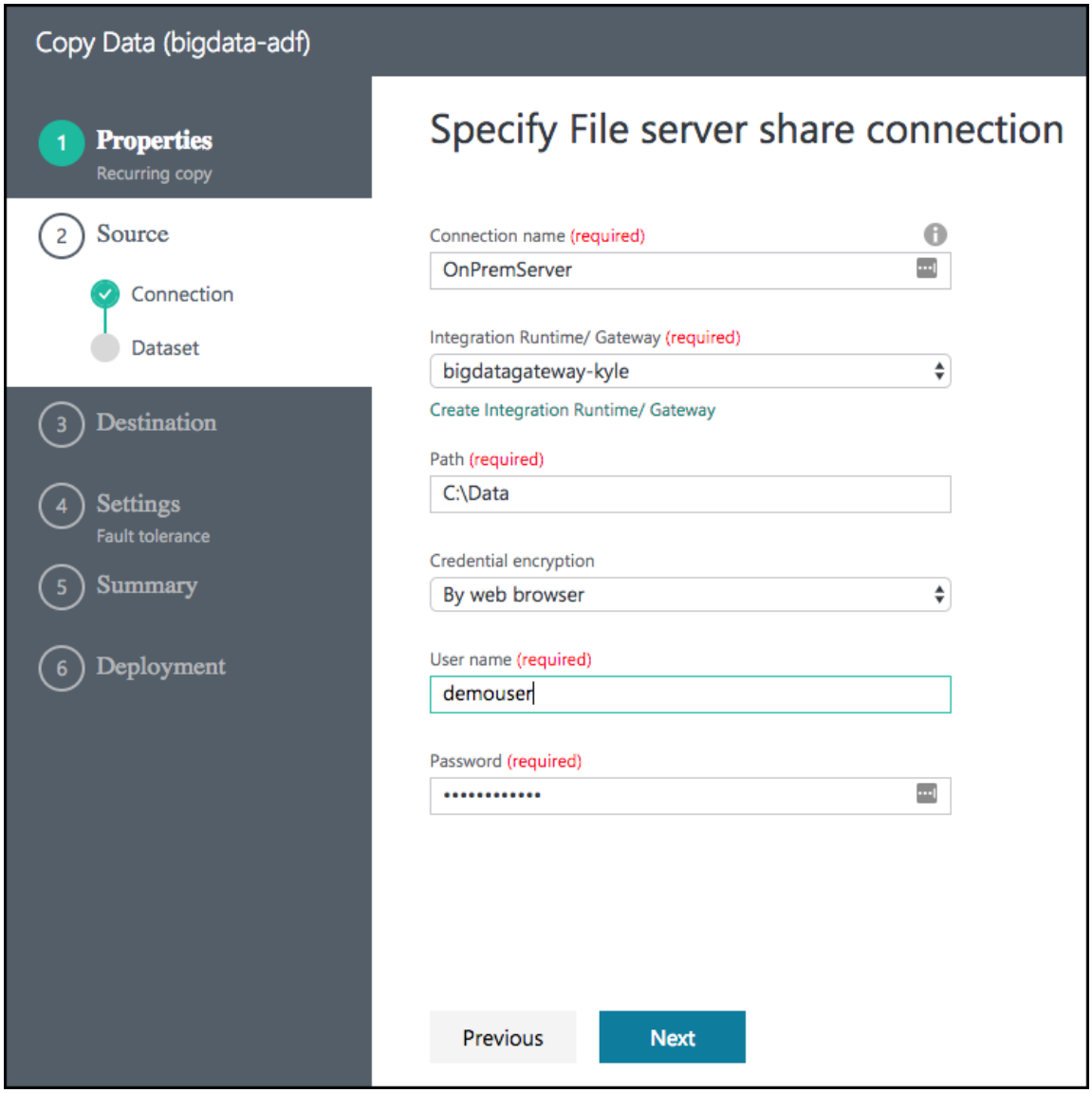
h. On the Source screen, select **File System**, then select **Next**.



i. From the Specify File server share connection screen, enter the following:

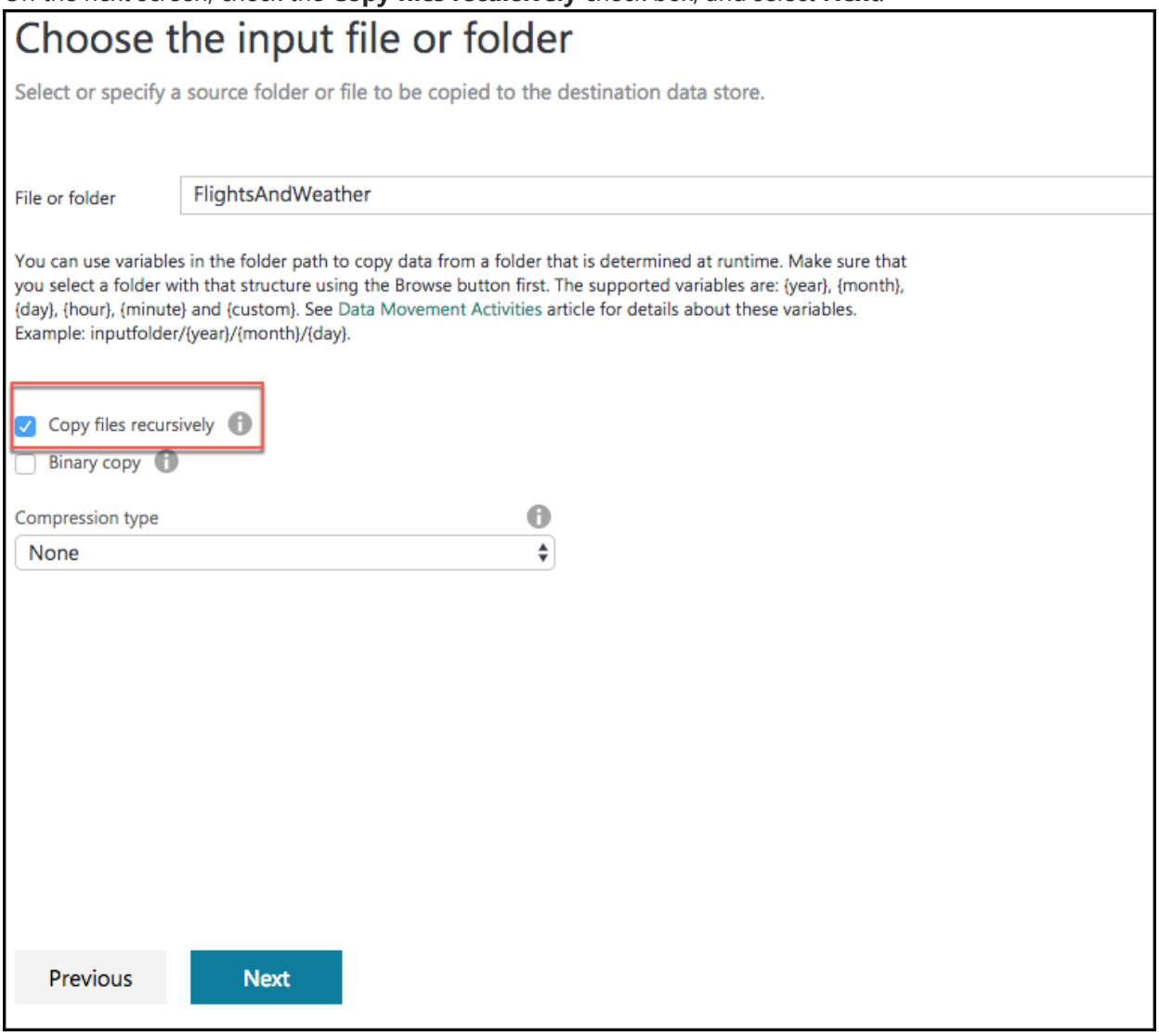
   i. Connection name: OnPremServer

ii. Integration Runtime/Gateway: Select the Integration runtime created previously in this exercise (this value should already be populated)
iii. Path: Enter C:\Data
iv. Credential encryption: Select By web browser
v. User name: Enter **demouser**
vi. Password: Enter **Password.1!!**
vii. Select **Next**

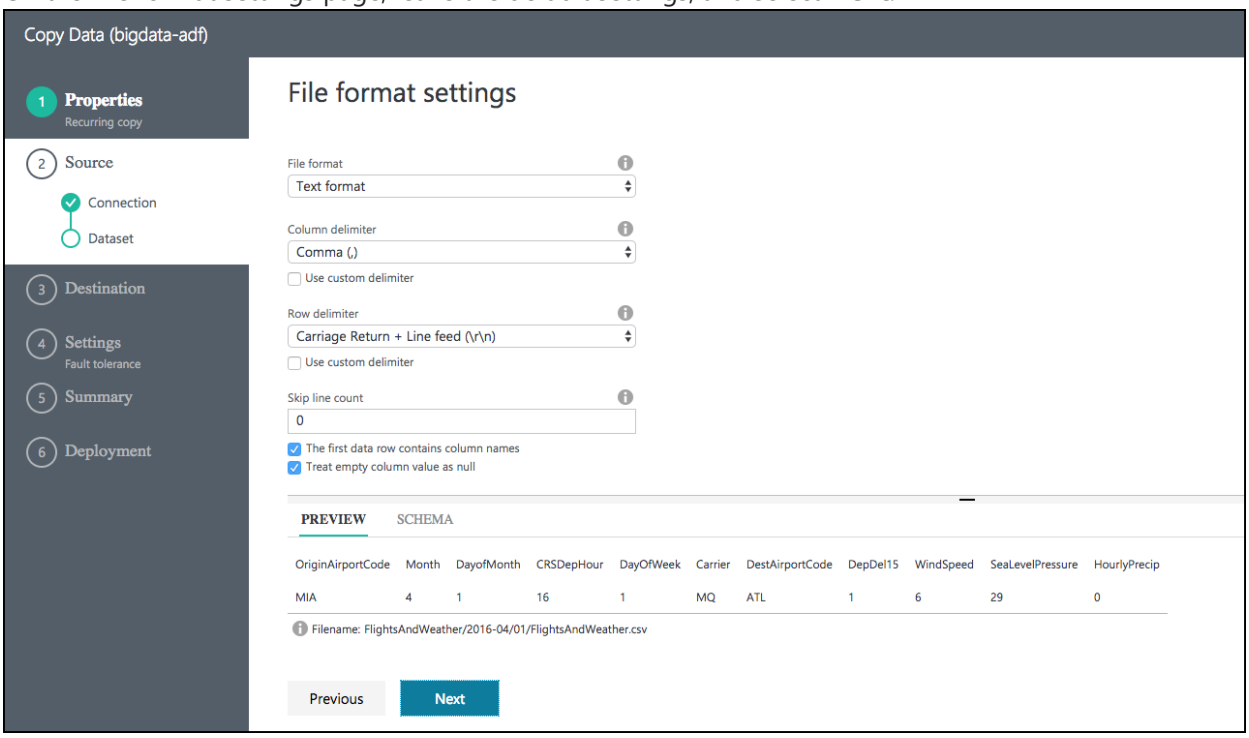


j. On the **Choose the input file or folder** screen, select the folder **FlightsAndWeather**, and select **Choose**.

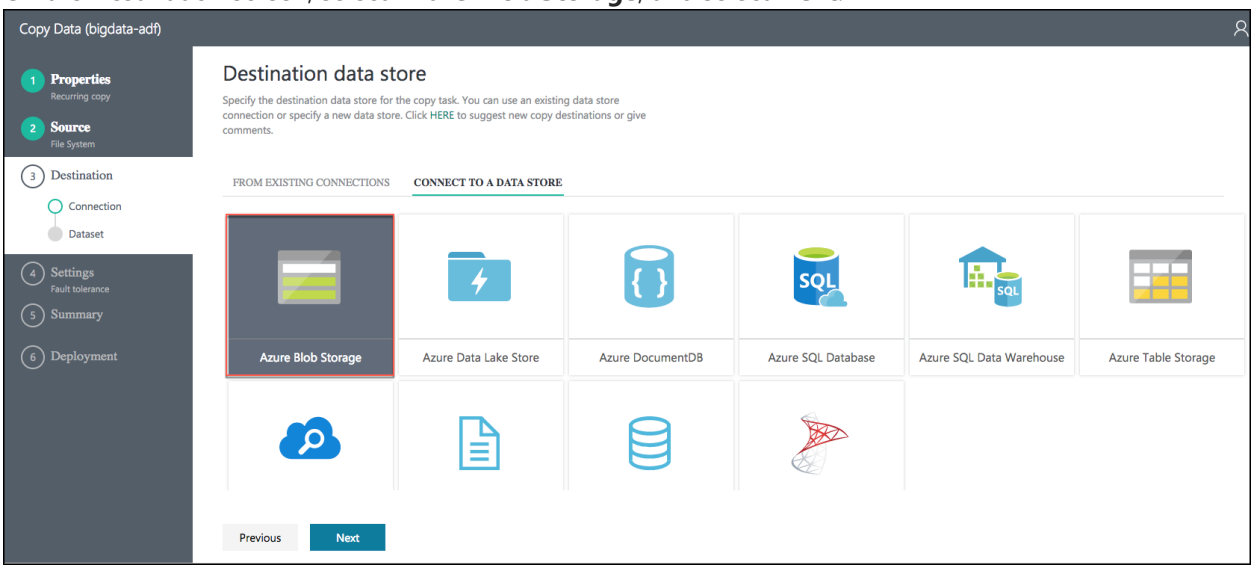
k. On the next screen, check the **Copy files recursively** check box, and select **Next**.



l. On the File format settings page, leave the default settings, and select **Next**.



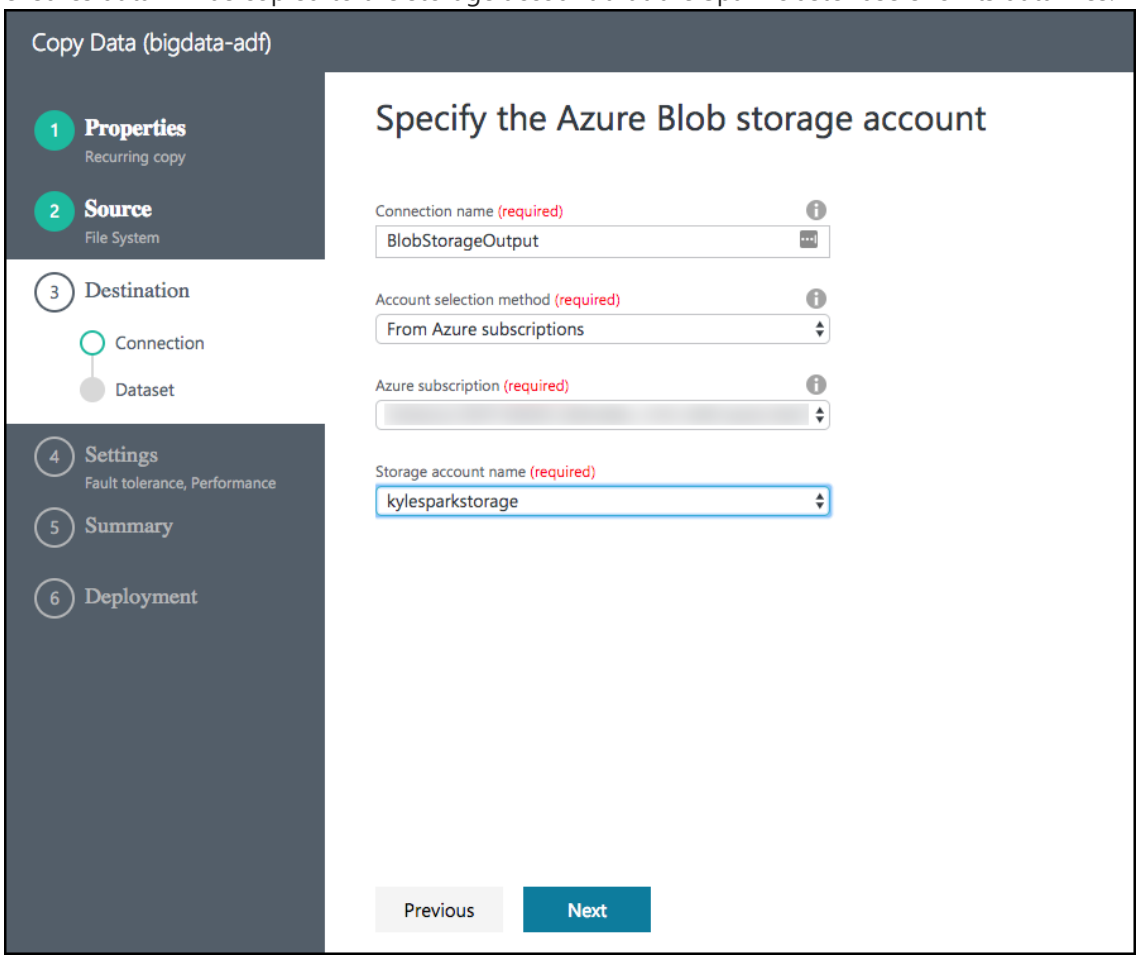
m. On the Destination screen, select **Azure Blob Storage**, and select **Next**.



n. On the Specify the Azure Blob storage account screen, enter the following:

   i. Connection name: BlobStorageOutput
   ii. Account selection method: Leave as From Azure subscriptions
   iii. Azure Subscription: Select your subscription
   iv. Storage account name: Select <YOUR_APP_NAME>sparkstorage. *Make sure you select the storage account with the **sparkstorage** suffix, or you will have issues with subsequent exercises.* This

ensures data will be copied to the storage account that the Spark cluster users for its data files.



o. Before selecting Next, please ensure you have selected the proper ***sparkstorage*** account. Finally, select **Next**.

p. From the Choose the output file or folder tab, enter the following:

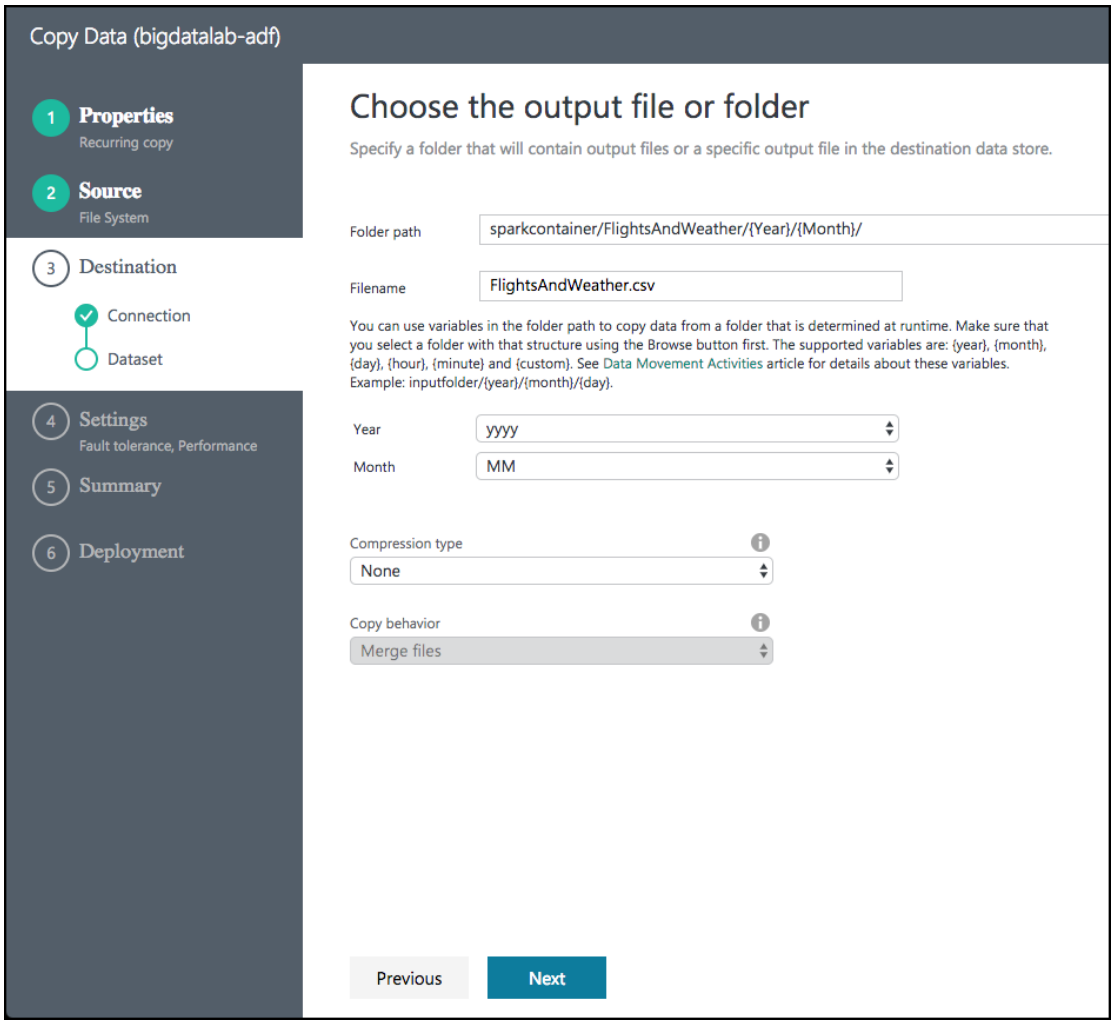
   i. Folder path: Enter sparkcontainer/FlightsAndWeather/{Year}/{Month}/

   ii. Filename: Enter FlightsAndWeather.csv

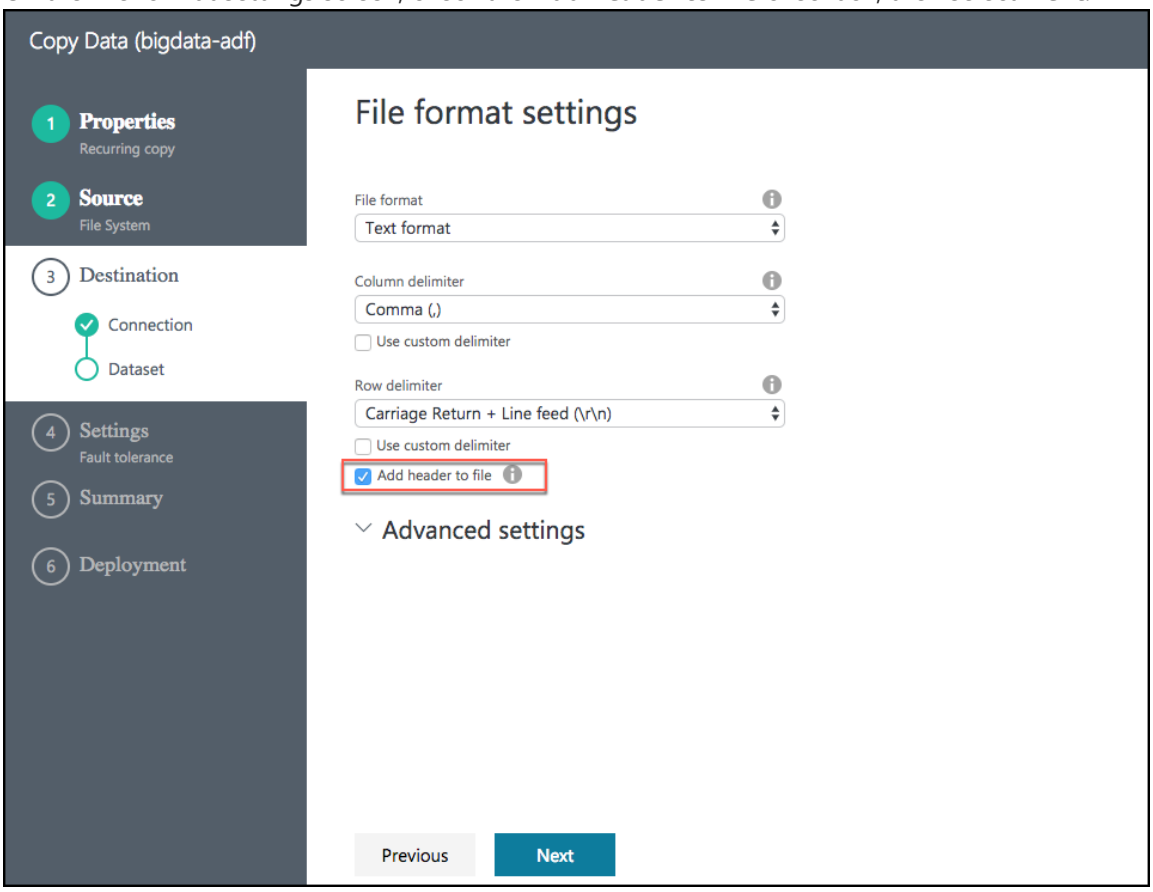
   iii. Year: Select yyyy from the drop down

   iv. Month: Leave as MM

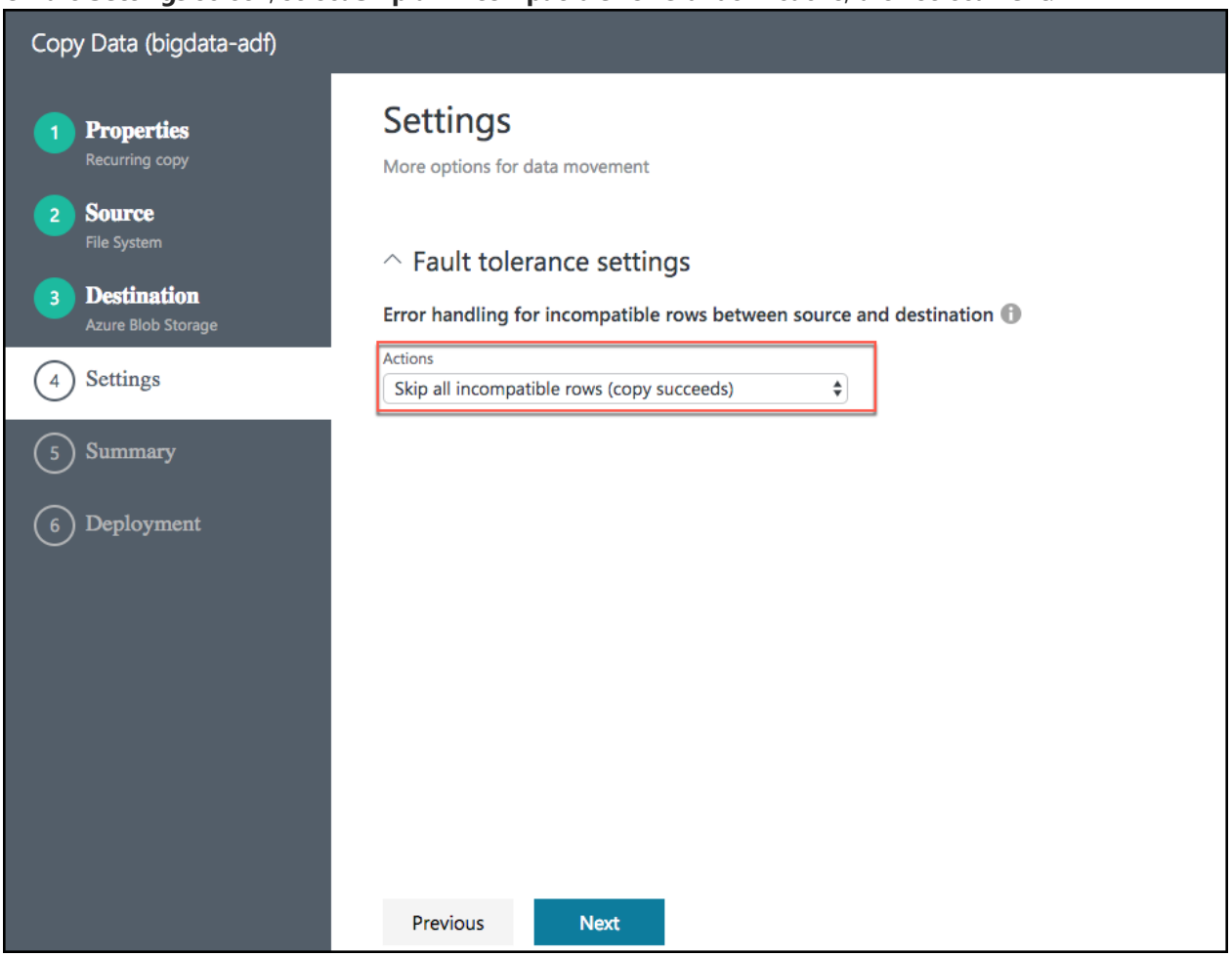
v. Select **Next**.



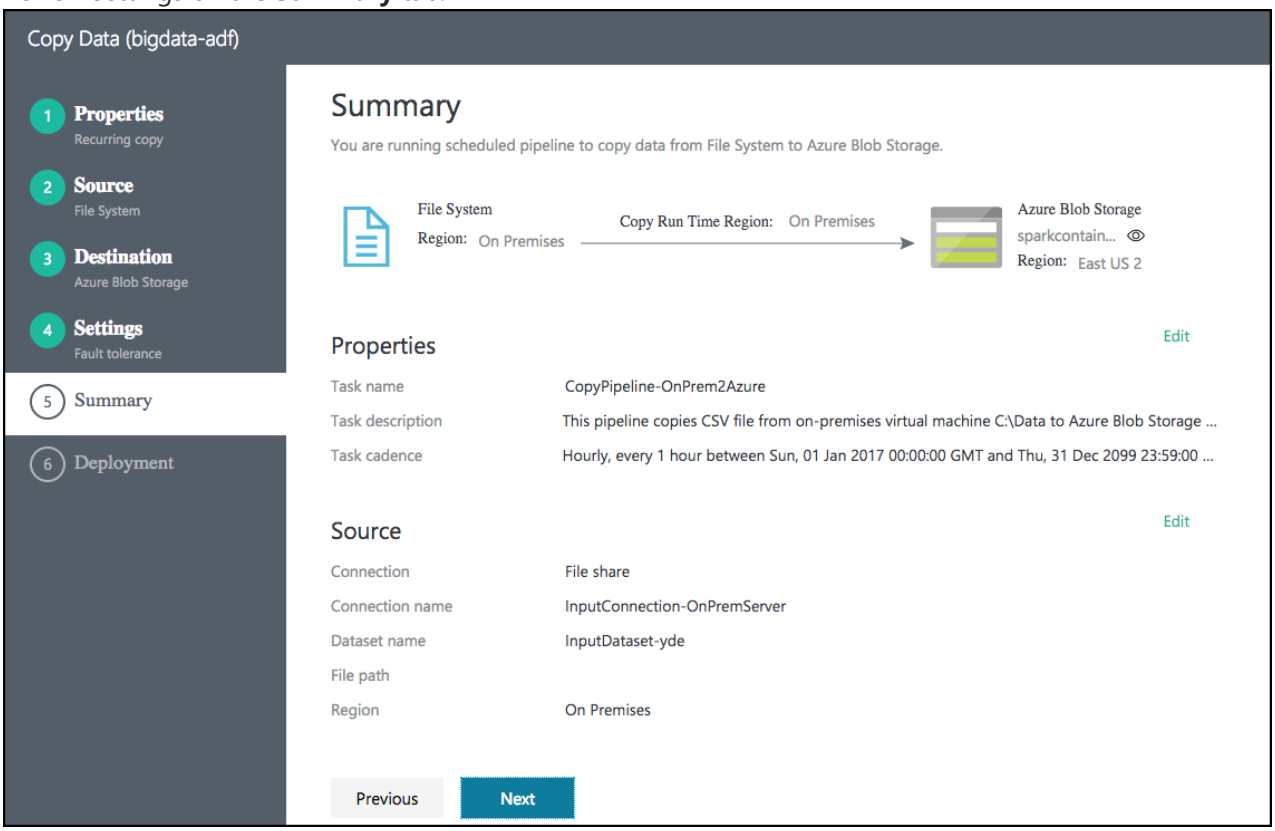
q. On the File format settings screen, check the **Add header to file** checkbox, then select **Next**.



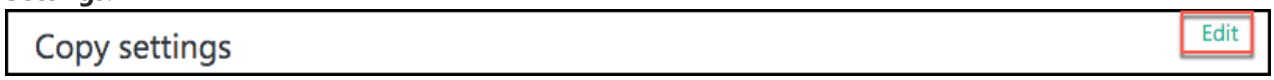
r. On the **Settings** screen, select **Skip all incompatible rows** under Actions, then select **Next**.



s. Review settings on the **Summary** tab.



t. Scroll down on the summary page until you see the **Copy Settings** section. Select **Edit** next to **Copy Settings**.

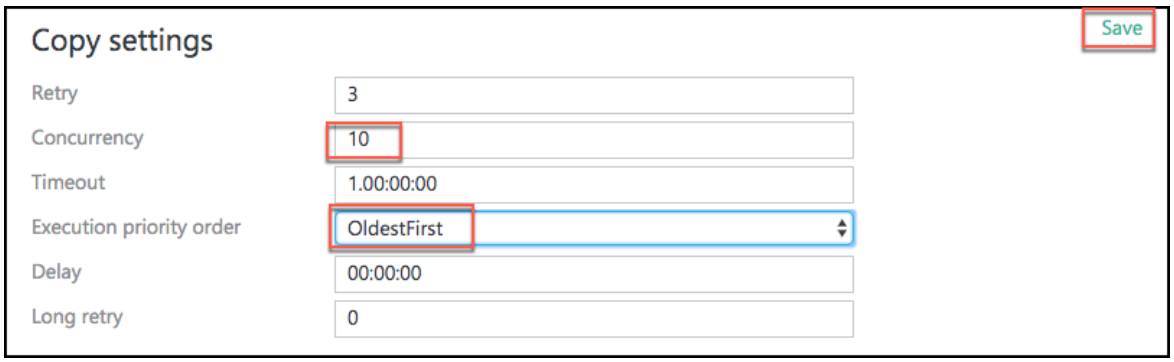


u. Change the following Copy settings

i. Concurrency: Set to 10

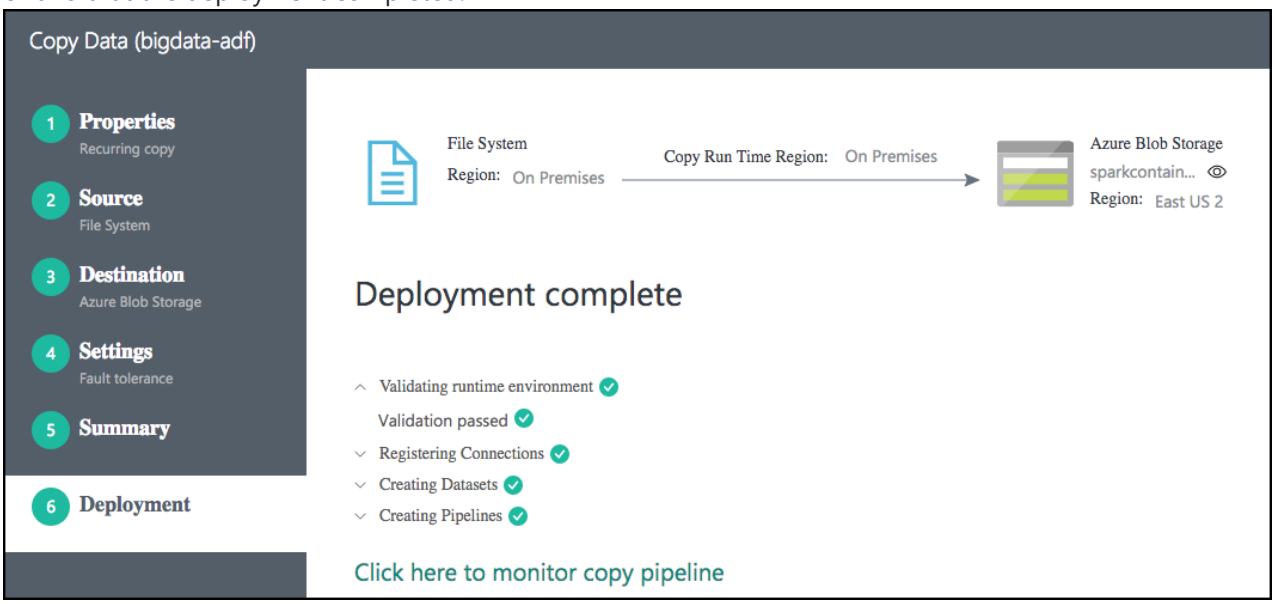
ii. Execution priority order: Change to OldestFirst

iii. Select **Save**



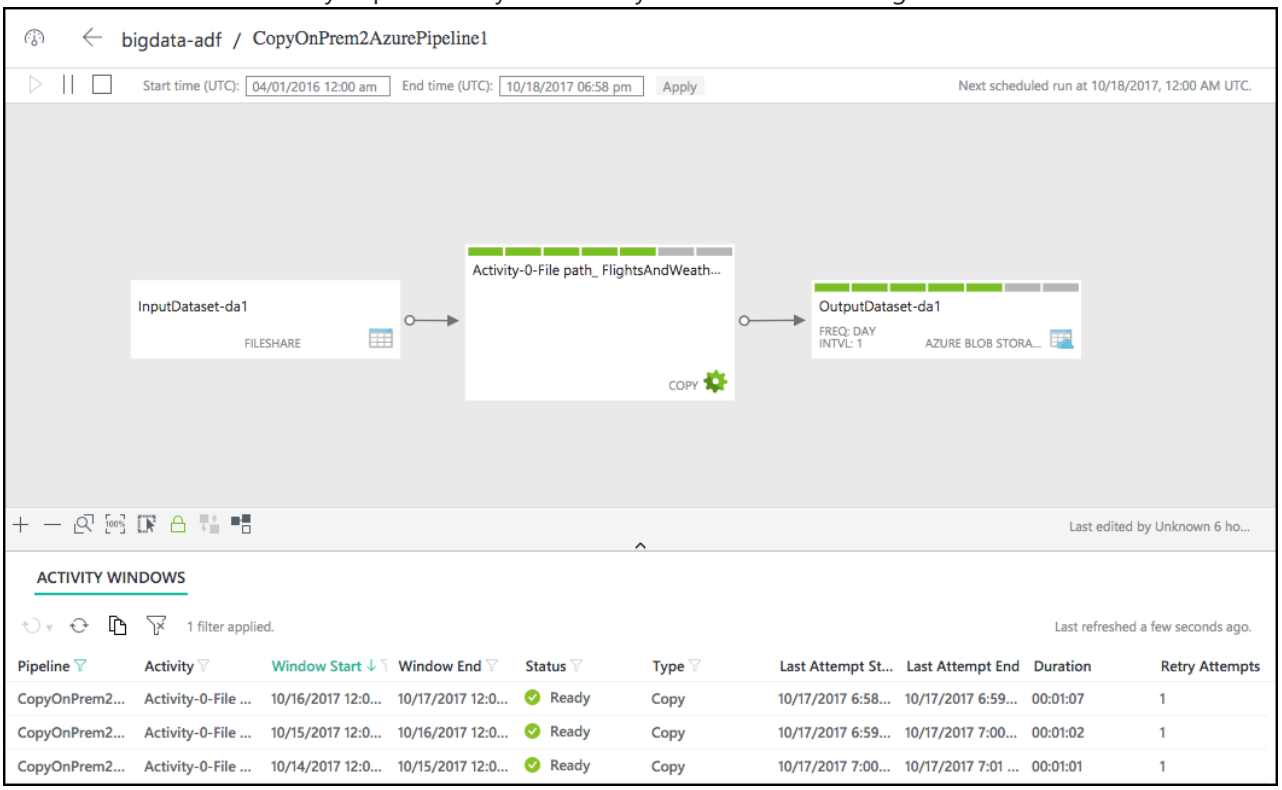
v. After saving the Copy settings, select **Next** on the Summary tab.

w. On the **Deployment** screen you will see a message that the deployment in is progress, and after a minute or two that the deployment completed.

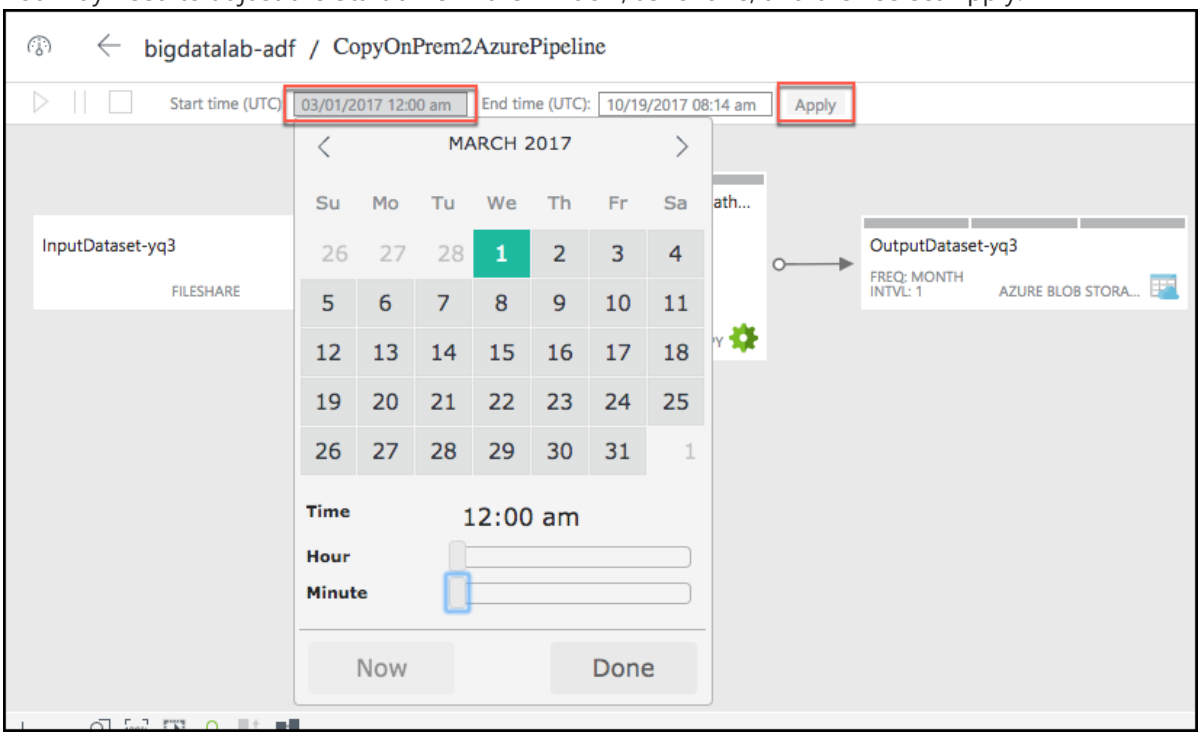


x. Select the **Click here to monitor copy pipeline** link at the bottom of the **Deployment** screen.

y. From the Data Factory Resource Explorer, you should see the pipeline activity status **Ready**. This indicates the CSV files are successfully copied from your VM to your Azure Blob Storage location.



z. You may need to adjust the Start time in the window, as follows, and then select Apply.



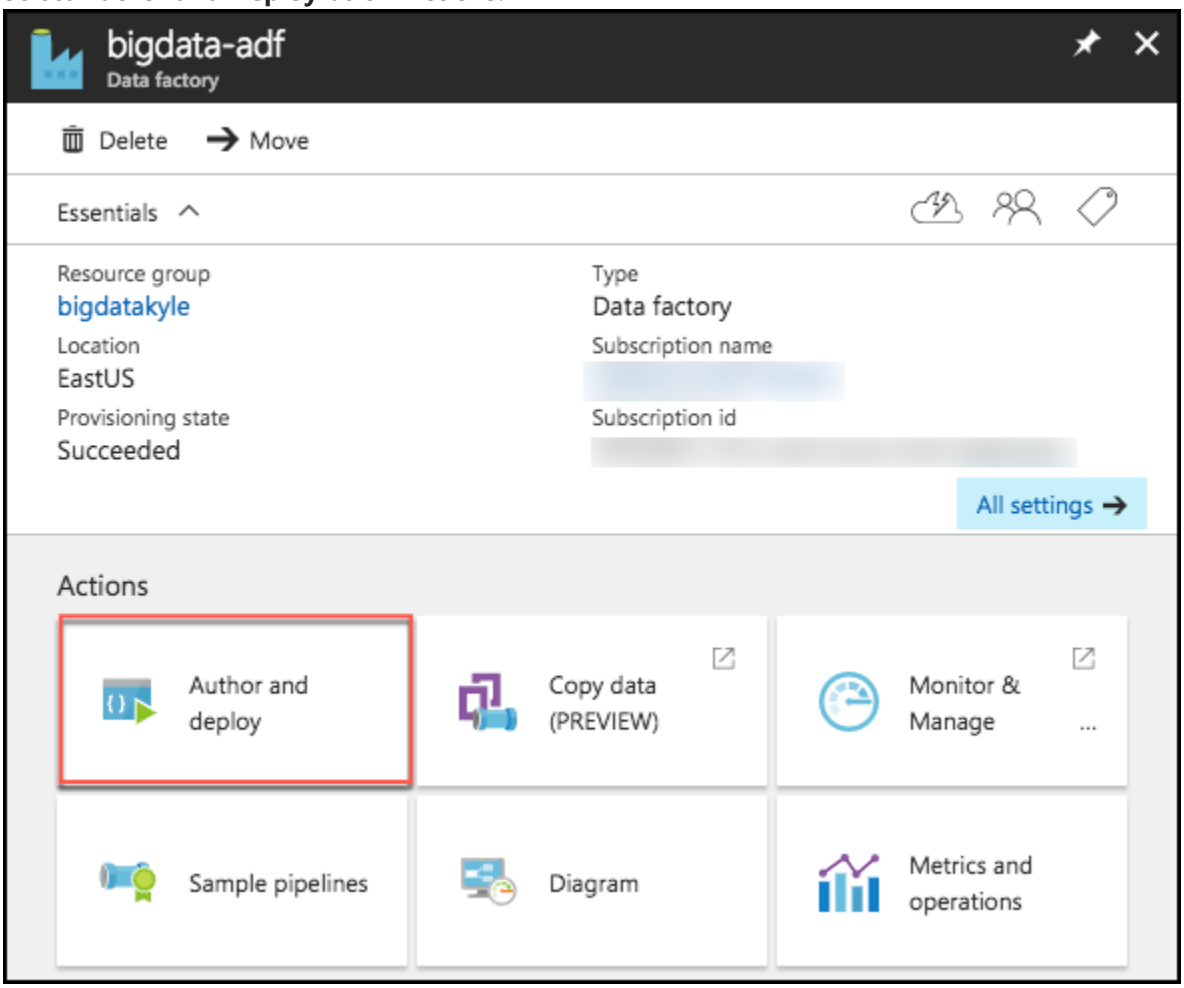
Exercise 4: Operationalize ML scoring with Azure ML and Data Factory

Duration: 20 minutes

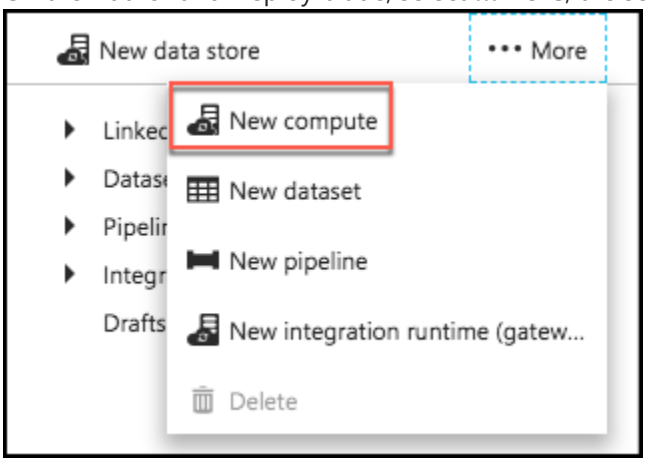
In this exercise, you will extend the Data Factory to operationalize the scoring of data using the previously created Azure Machine Learning (ML) model.

Task 1: Create Azure ML Linked Service

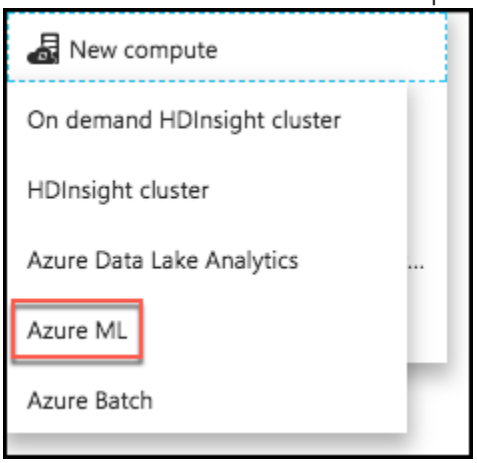
1. Return to the Azure Data Factory blade in the Azure portal
2. Select **Author and Deploy** below **Actions**.



3. On the Author and Deploy blade, select **...More**, the select **New Compute**.



4. Select **Azure ML** from the New Compute list.



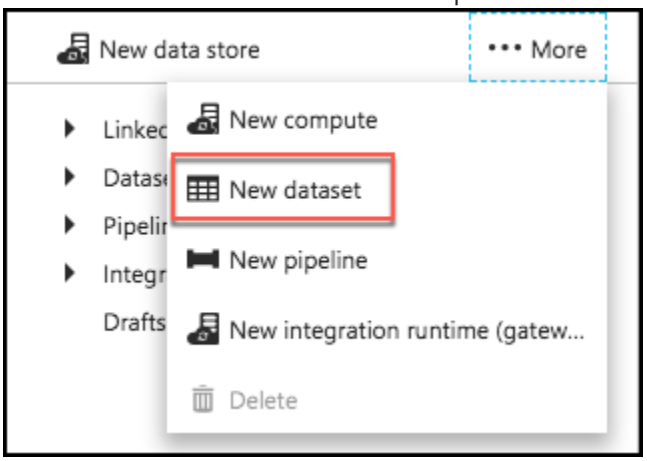
5. In the new window, replace the contents of the file with the following JSON.
   - Back in Exercise 1, Task 9, you left your ML Web Service's Consume page open. Return to that page, and copy and paste the following values into the JSON below.
   - The value of **mlEndpoint** below is your web service's **Batch Request URL,** remember to ***remove*** the query string (e.g., "?api_version=2.0").
   - **apiKey** is the **Primary Key** of your web service.
   - Your tenant string should be populated automatically.
   - Delete the other optional settings (updateResourceEndpoint, servicePrincipalId, servicePrincipalKey).

```
{
    "name": "AzureMLLinkedService",
    "properties": {
        "type": "AzureML",
        "description": "",
        "typeProperties": {
            "mlEndpoint": "<Specify the batch scoring URL>",
            "apiKey": "<Specify the published workspace model's API key>",
            "tenant": "<Specify your tenant string>"
        }
    }
}
```

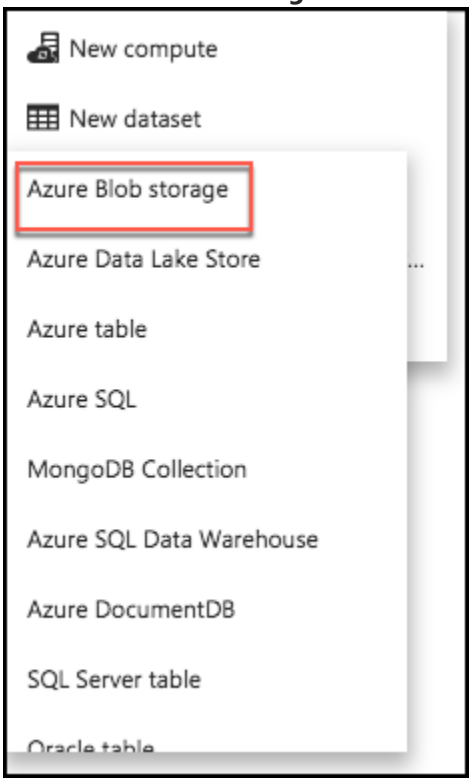
6. Select **Deploy**.

Task 2: Create Azure ML input dataset

1. Still on the Author and Deploy blade, select **...More** again.
2. To create a new dataset that will be copied into Azure Blob storage, select **New dataset** from the top.



3. Select **Azure Blob storage** from the list of available datasets.



4. Replace the JSON text in the draft window with following JSON.

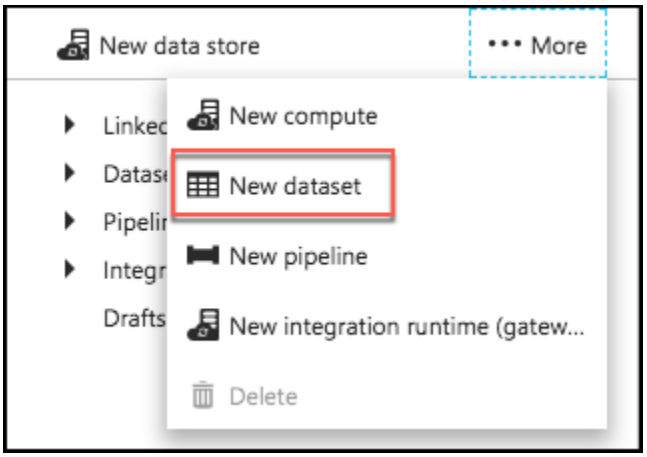
```
{
    "name": "PartitionedBlobInput",
    "properties": {
        "published": false,
        "type": "AzureBlob",
        "linkedServiceName": "BlobStorageOutput",
        "typeProperties": {
```

```
            "fileName": "FlightsAndWeather.csv",
            "folderPath": "sparkcontainer/FlightsAndWeather/{Year}/{Month}/",
            "format": {
                "type": "TextFormat"
            },
            "partitionedBy": [
                {
                    "name": "Year",
                    "value": {
                        "type": "DateTime",
                        "date": "SliceStart",
                        "format": "yyyy"
                    }
                },
                {
                    "name": "Month",
                    "value": {
                        "type": "DateTime",
                        "date": "SliceStart",
                        "format": "MM"
                    }
                }
            ]
        },
        "availability": {
            "frequency": "Month",
            "interval": 1
        },
        "external": true,
        "policy": {}
    }
}
```

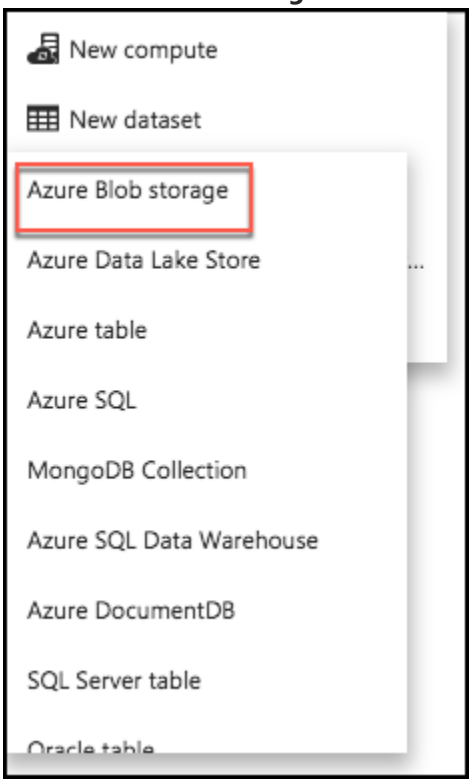
5. Select **Deploy**.

Task 3: Create Azure ML scored dataset

1. Select **...More** again, and select **New dataset**.



2. Select **Azure Blob storage** from the list of available datasets.



3. Replace the JSON text in the draft window with following JSON.

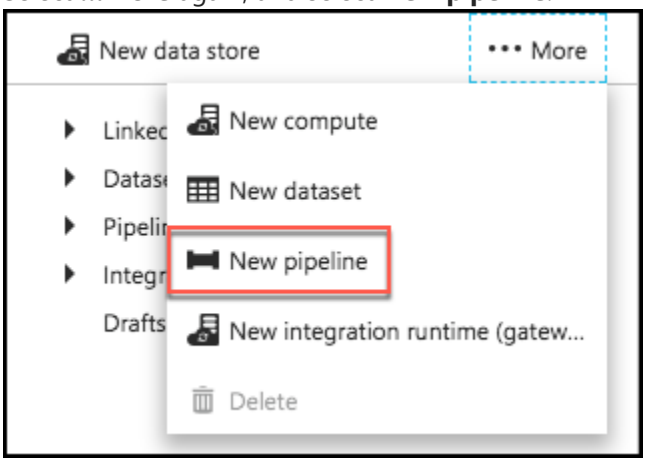
```
{
    "name": "ScoredBlobOutput",
    "properties": {
        "published": false,
        "type": "AzureBlob",
        "linkedServiceName": "BlobStorageOutput",
        "typeProperties": {
            "fileName": "Scored_FlightsAndWeather{Year}{Month}.csv",
            "folderPath": "sparkcontainer/ScoredFlightsAndWeather",
            "format": {
                "type": "TextFormat"
            },
            "partitionedBy": [
```

```
                {
                    "name": "Year",
                    "value": {
                        "type": "DateTime",
                        "date": "SliceStart",
                        "format": "yyyy"
                    }
                },
                {
                    "name": "Month",
                    "value": {
                        "type": "DateTime",
                        "date": "SliceStart",
                        "format": "MM"
                    }
                }
            ]
        },
        "availability": {
            "frequency": "Month",
            "interval": 1
        }
    }
}
```

4. Select **Deploy**.

Task 4: Create Azure ML predictive pipeline

1. Select **...More** again, and select **New pipeline**.



2. Replace the JSON text in the draft window with following JSON.

```
{
    "name": "MLPredictivePipeline",
    "properties": {
        "description": "Use AzureML model",
        "activities": [
            {
```



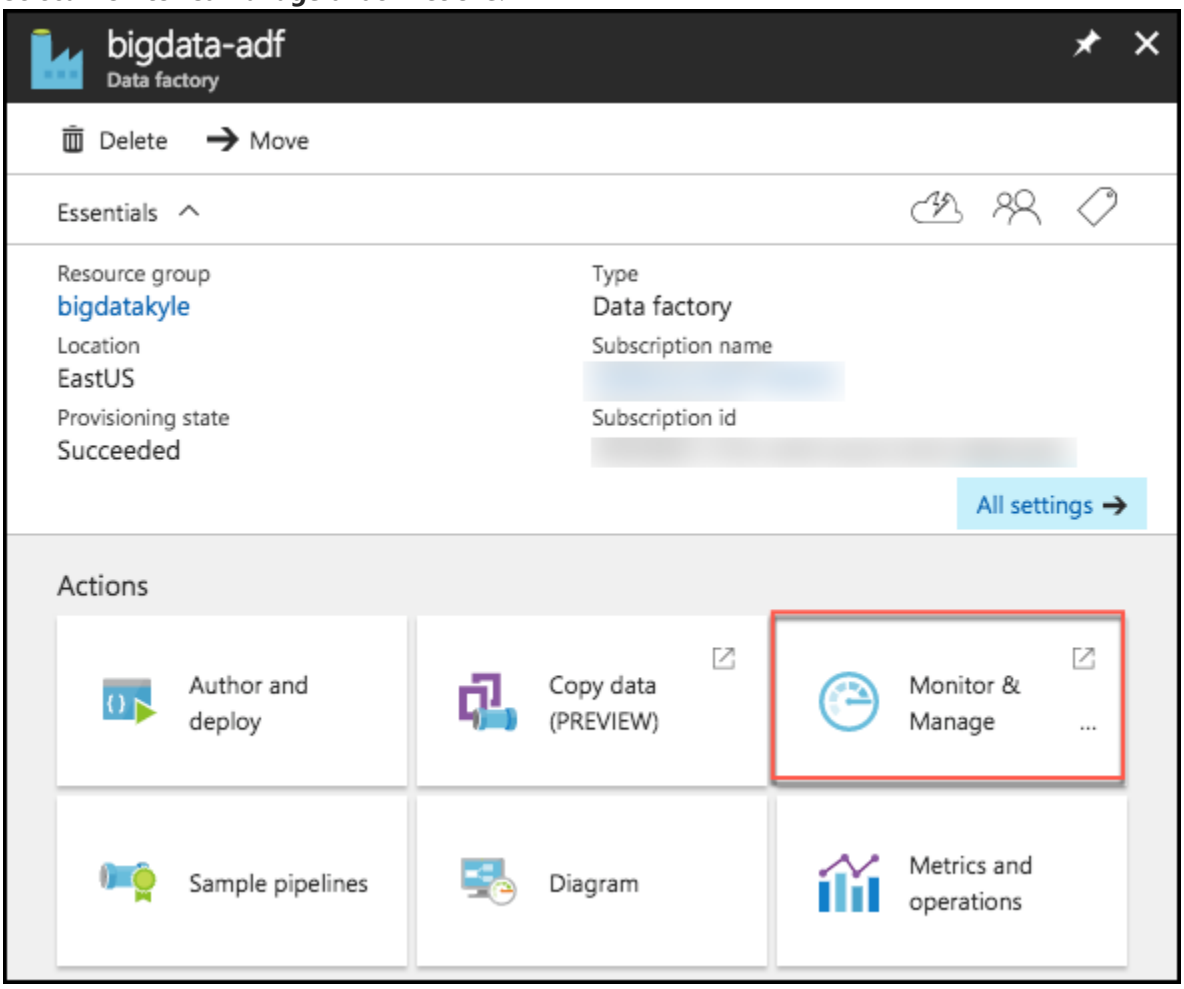
```
                "type": "AzureMLBatchExecution",
                "typeProperties": {
                    "webServiceInput": "PartitionedBlobInput",
                    "webServiceOutputs": {
                        "output1": "ScoredBlobOutput"
                    },
                    "webServiceInputs": {},
                    "globalParameters": {}
                },
                "inputs": [
                    {
                        "name": "PartitionedBlobInput"
                    }
                ],
                "outputs": [
                    {
                        "name": "ScoredBlobOutPut"
                    }
                ],
                "policy": {
                    "timeout": "02:00:00",
                    "concurrency": 10,
                    "retry": 1
                },
                "scheduler": {
                    "frequency": "Month",
                    "interval": 1
                },
                "name": "MLActivity",
                "description": "prediction analysis on batch input",
                "linkedServiceName": "AzureMLLinkedService"
            }
        ],
        "start": "2017-03-01T00:00:00Z",
        "end": "2099-12-31T11:59:59Z",
        "isPaused": false,
        "pipelineMode": "Scheduled"
    }
}
```

3. Select **Deploy**.

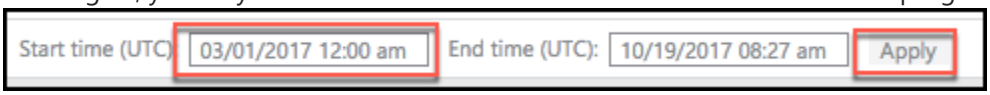
Task 5: Monitor pipeline activities

1. Close the Author and Deploy blade, and return to the Data Factory overview.

2. Select **Monitor & Manage** under **Actions**.



3. Once again, you may need to shift the start time in order to see the items in progress and ready states.



4. Close the **Monitor & Manage** browser tab.

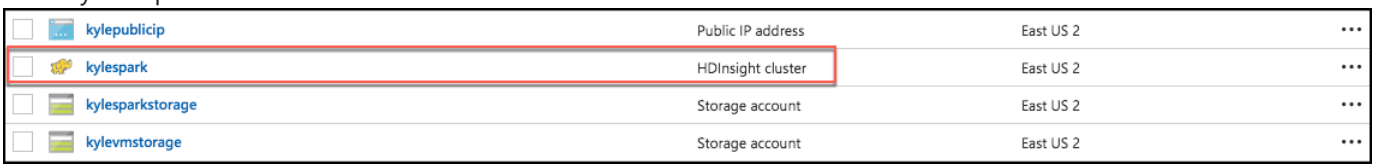
Exercise 5: Summarize data using HDInsight Spark

Duration: 20 minutes

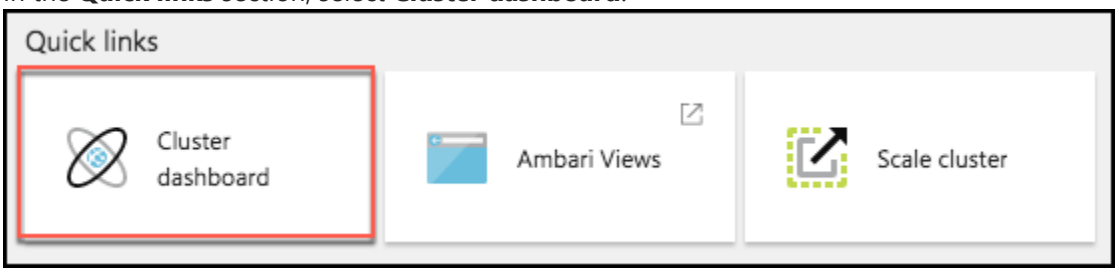
In this exercise, you will prepare a summary of flight delay data in HDFS using Spark SQL.

Task 1: Summarize delays by airport

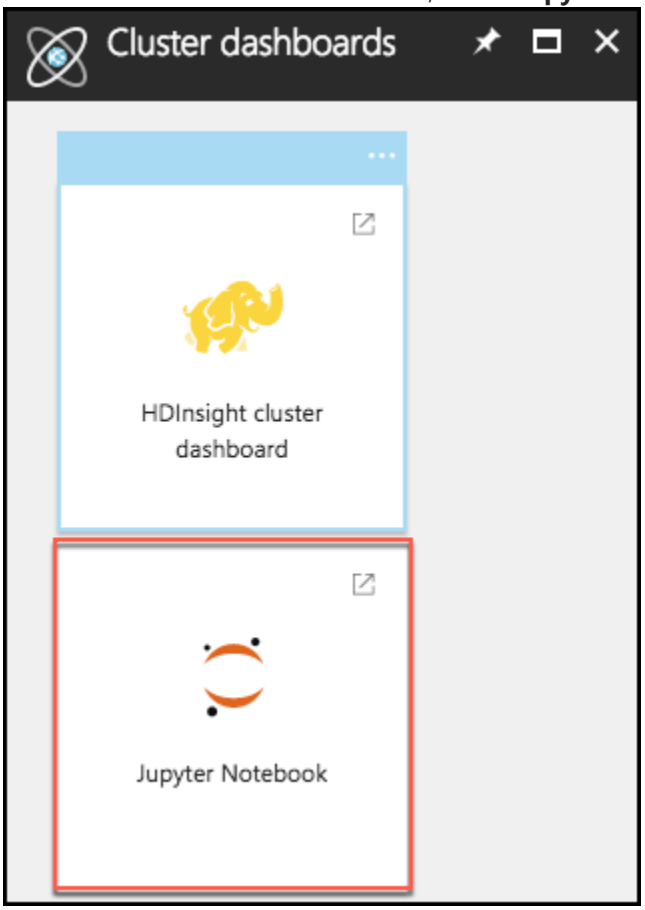
1. In the Azure portal (https://portal.azure.com), navigate to the blade for your Spark cluster. Do this by going to the resource group you created during the lab setup, using the Resource Group link in the left-hand menu. Once you select your resource group, you will see a list of the resources within that group, including your Spark cluster. Select your Spark cluster.



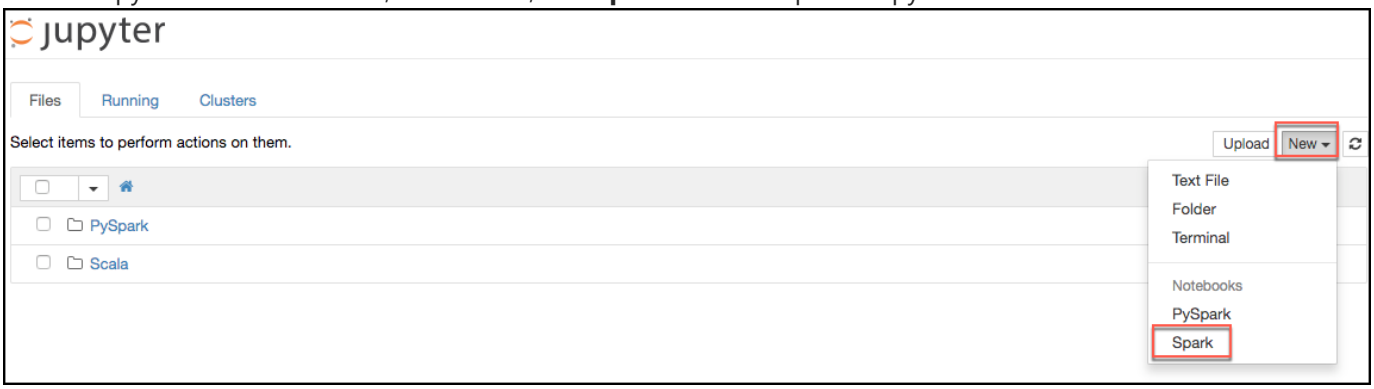
2. In the **Quick links** section, select **Cluster dashboard**.



3. From the **Cluster dashboards** blade, select **Jupyter Notebook**.



4. Juptyer Notebook will open in a new browser window. Log in with the following credentials:
    - User name: demouser
    - Password: Password.1!!
    - Note: If you get a 403 – Forbidden: Access is denied error, try to open the jupyter URL in a private or incognito browser window. You can also clear the browser cache.
5. On the Jupyter Notebook screen, select **New**, and **Spark**. This will open a Jupyter notebook in a new browser tab.



6. Copy the text below, and paste it into the first cell in the Jupyter notebook. This will read the data from our Scored_FlightsAndWeather.csv file, and output it into a Hive table named "FlightDelays."

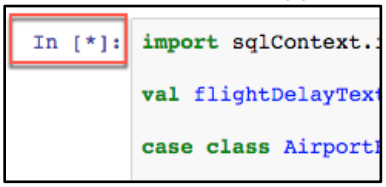
```
import spark.sqlContext.implicits._

val flightDelayTextLines = sc.textFile("/ScoredFlightsAndWeather/*.csv")

case class
AirportFlightDelays(OriginAirportCode:String,OriginLatLong:String,Month:Integer,Day:Integer
,Hour:Integer,Carrier:String,DelayPredicted:Integer,DelayProbability:Double)

val flightDelayRowsWithoutHeader = flightDelayTextLines.map(s => s.split(",")).filter(line
=> line(0) != "OriginAirportCode")

val resultDataFrame = flightDelayRowsWithoutHeader.map(
    s => AirportFlightDelays(
        s(0), //Airport code
        s(13) + "," + s(14), //Lat,Long
        s(1).toInt, //Month
        s(2).toInt, //Day
        s(3).toInt, //Hour
        s(5), //Carrier
        s(11).toInt, //DelayPredicted
        s(12).toDouble //DelayProbability
        )
).toDF()

resultDataFrame.write.mode("overwrite").saveAsTable("FlightDelays")
```


7. The notebook should now look like the image below.

```
In [28]: import spark.sqlContext.implicits._

val flightDelayTextLines = sc.textFile("/ScoredFlightsAndWeather/*.csv")

case class AirportFlightDelays(OriginAirportCode:String,OriginLatLong:String,Month:Integer,Day:Integer,Hour:Integer,Car

val flightDelayRowsWithoutHeader = flightDelayTextLines.map(s => s.split(",")).filter(line => line(0) != "OriginAirport(

val resultDataFrame = flightDelayRowsWithoutHeader.map(
    s => AirportFlightDelays(
        s(0), //Airport code
        s(13) + "," + s(14), //Lat,Long
        s(1).toInt, //Month
        s(2).toInt, //Day
        s(3).toInt, //Hour
        s(5), //Carrier
        s(11).toInt, //DelayPredicted
        s(12).toDouble //DelayProbability
        )
).toDF()

resultDataFrame.write.mode("overwrite").saveAsTable("FlightDelays")
```

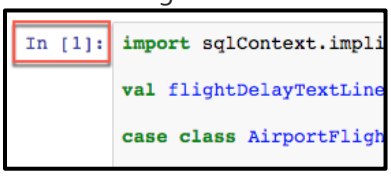
8. Select the **Run cell** button on the toolbar.

9. You will see in asterisk appear between the brackets in front of the cell.



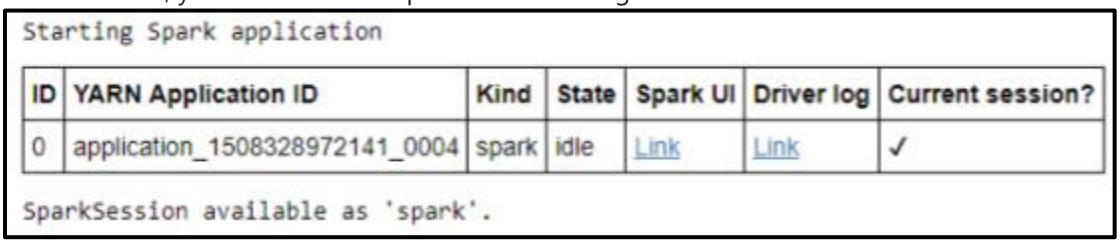
```
In [*]: import sqlContext.i

val flightDelayTex

case class AirportI
```

10. This will change to a number once the command is complete.



```
In [1]: import sqlContext.impli

val flightDelayTextLine

case class AirportFligh
```

11. Below the cell, you will see the output from executing the command.



Starting Spark application

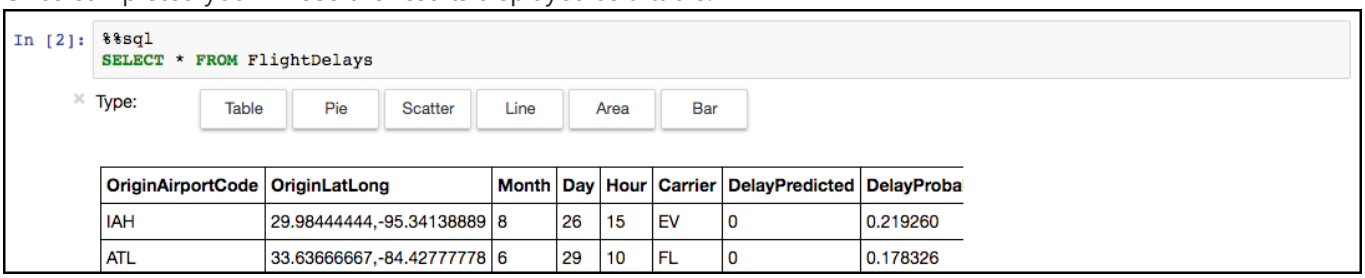
ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1508328972141_0004	spark	idle	Link	Link	✓

SparkSession available as 'spark'.

12. Now, we can query the hive table which was created by the previous command. Paste the text below into the empty cell at the bottom on the notebook, and select the Run cell button for that cell.

```
%%sql
SELECT * FROM FlightDelays
```

13. Once completed you will see the results displayed as a table.



```
In [2]: %%sql
SELECT * FROM FlightDelays
```

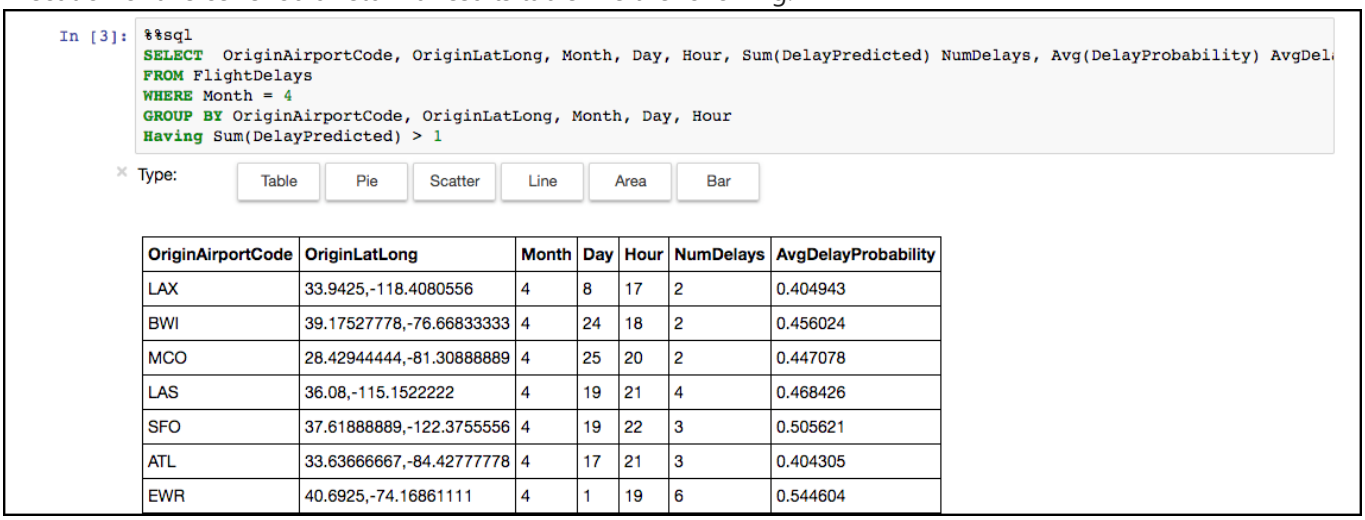
Type: Table | Pie | Scatter | Line | Area | Bar

OriginAirportCode	OriginLatLong	Month	Day	Hour	Carrier	DelayPredicted	DelayProba
IAH	29.98444444,-95.34138889	8	26	15	EV	0	0.219260
ATL	33.63666667,-84.42777778	6	29	10	FL	0	0.178326

14. Next, you will create a table that summarizes the flight delays data. Instead of containing one row per flight, this new summary table will contain one row per origin airport at a given hour, along with a count of the quantity of anticipated delays. In a new cell below the results of our previous cell, paste the following text, and select the Run cell button from the toolbar.

```
%%sql
SELECT  OriginAirportCode, OriginLatLong, Month, Day, Hour, Sum(DelayPredicted) NumDelays,
Avg(DelayProbability) AvgDelayProbability
FROM FlightDelays
WHERE Month = 4
GROUP BY OriginAirportCode, OriginLatLong, Month, Day, Hour
Having Sum(DelayPredicted) > 1
```

15. Execution of this cell should return a results table like the following.



```
In [3]: %%sql
SELECT  OriginAirportCode, OriginLatLong, Month, Day, Hour, Sum(DelayPredicted) NumDelays, Avg(DelayProbability) AvgDel
FROM FlightDelays
WHERE Month = 4
GROUP BY OriginAirportCode, OriginLatLong, Month, Day, Hour
Having Sum(DelayPredicted) > 1
```

Type: Table | Pie | Scatter | Line | Area | Bar

OriginAirportCode	OriginLatLong	Month	Day	Hour	NumDelays	AvgDelayProbability
LAX	33.9425,-118.4080556	4	8	17	2	0.404943
BWI	39.17527778,-76.66833333	4	24	18	2	0.456024
MCO	28.42944444,-81.30888889	4	25	20	2	0.447078
LAS	36.08,-115.1522222	4	19	21	4	0.468426
SFO	37.61888889,-122.3755556	4	19	22	3	0.505621
ATL	33.63666667,-84.42777778	4	17	21	3	0.404305
EWR	40.6925,-74.16861111	4	1	19	6	0.544604

16. Since the summary data looks good, the final step is to save this summary calculation as a table, which we can later query using Power BI (in the next exercise).
17. To accomplish this, paste the text below into a new cell, and select the Run cell button from the toolbar.

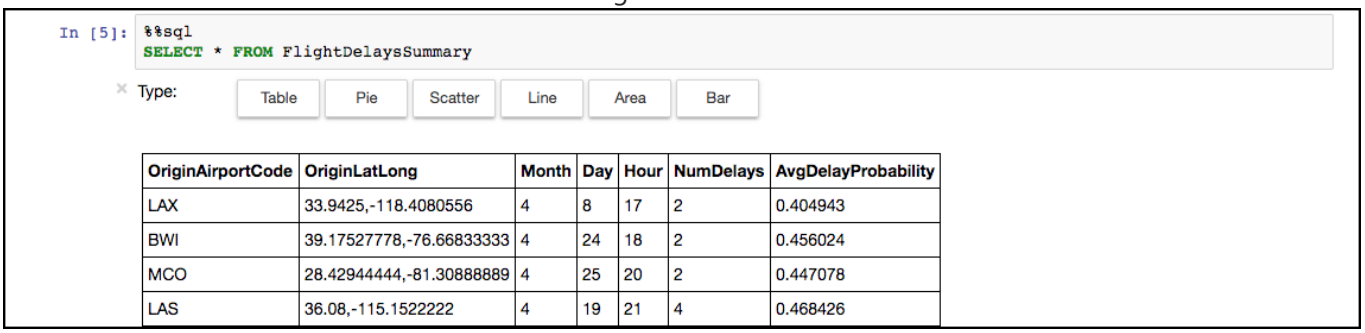
```
val summary = spark.sqlContext.sql("SELECT  OriginAirportCode, OriginLatLong, Month, Day,
Hour, Sum(DelayPredicted) NumDelays, Avg(DelayProbability) AvgDelayProbability FROM
FlightDelays WHERE Month = 4 GROUP BY OriginAirportCode, OriginLatLong, Month, Day, Hour
Having Sum(DelayPredicted) > 1")
summary.write.mode("overwrite").saveAsTable("FlightDelaysSummary")
```

18. To verify the table was successfully created, go to another new cell, and enter the following query.

```
%%sql
SELECT * FROM FlightDelaysSummary
```


19. Select the **Run cell** button on the toolbar.

20. You should see a results table similar to the following.



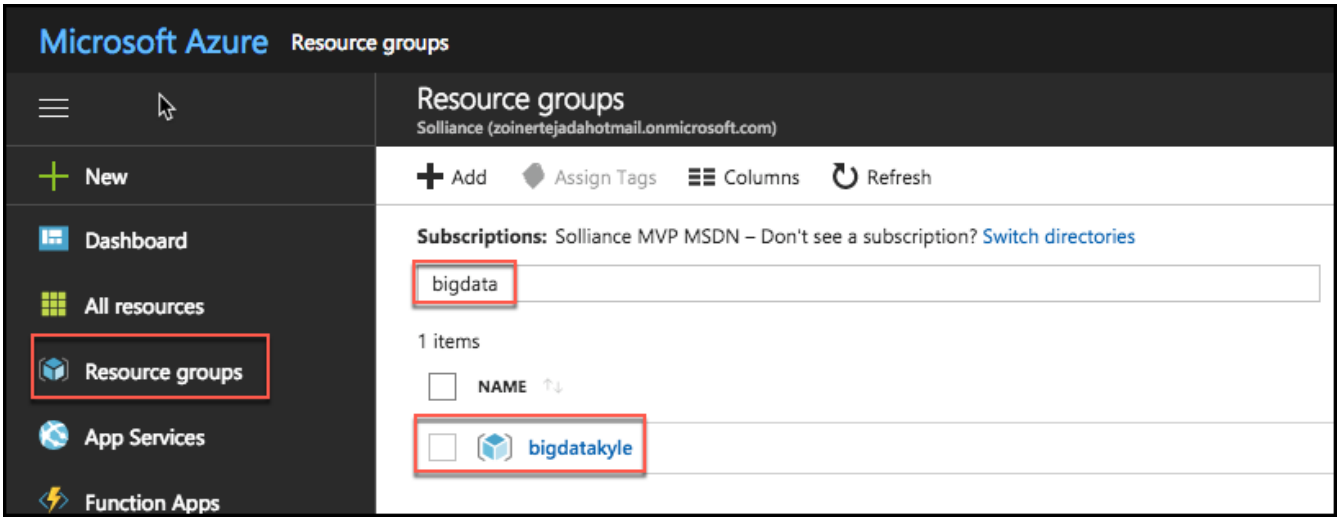
OriginAirportCode	OriginLatLong	Month	Day	Hour	NumDelays	AvgDelayProbability
LAX	33.9425,-118.4080556	4	8	17	2	0.404943
BWI	39.17527778,-76.66833333	4	24	18	2	0.456024
MCO	28.42944444,-81.30888889	4	25	20	2	0.447078
LAS	36.08,-115.1522222	4	19	21	4	0.468426

21. You can also select Pie, Scatter, Line, Area, and Bar chart visualizations of the dataset.

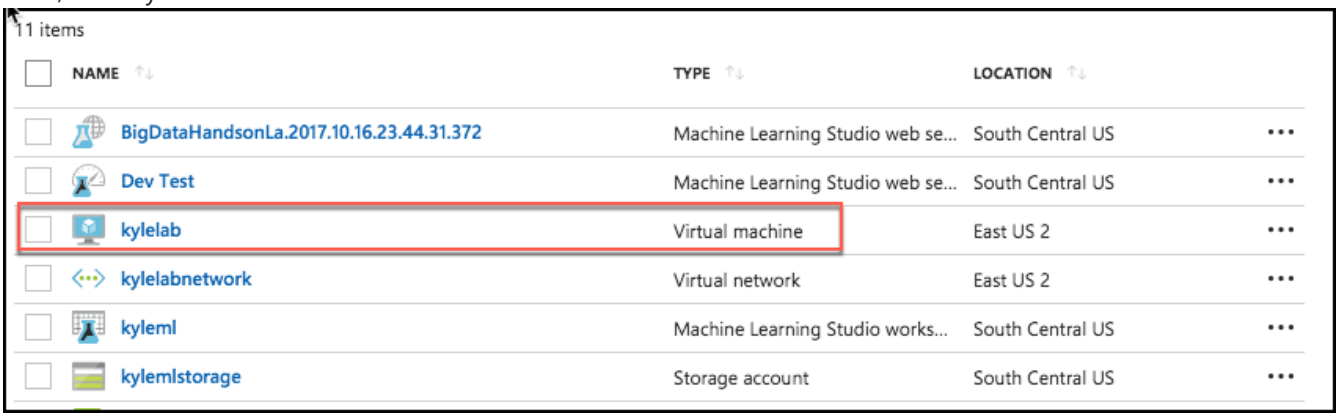
Exercise 6: Visualizing in Power BI Desktop

Task 1: Connect to the Lab VM

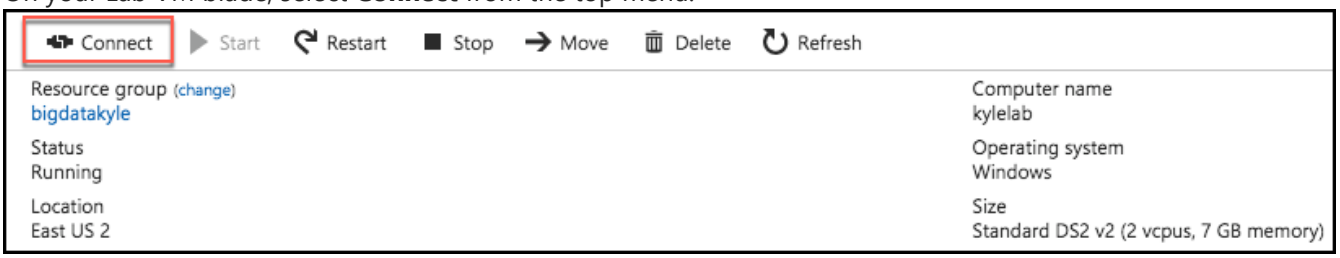
7. NOTE: If you are already, connected to your Lab VM, skip to Task 2.
8. From the left side menu in the Azure portal, click on **Resource groups**, then enter your resource group name into the filter box, and select it from the list.



9. Next, select your lab virtual machine from the list.



10. On your Lab VM blade, select **Connect** from the top menu.

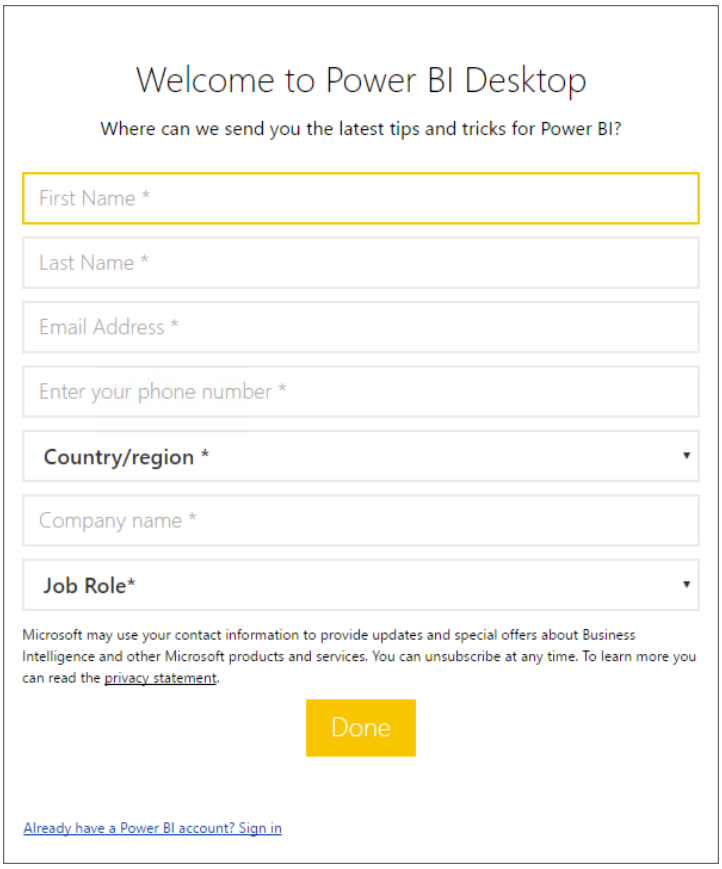


11. Download and open the RDP file.
12. Select Connect, and enter the following credentials:
    - User name: demouser
    - Password: Password.1!!

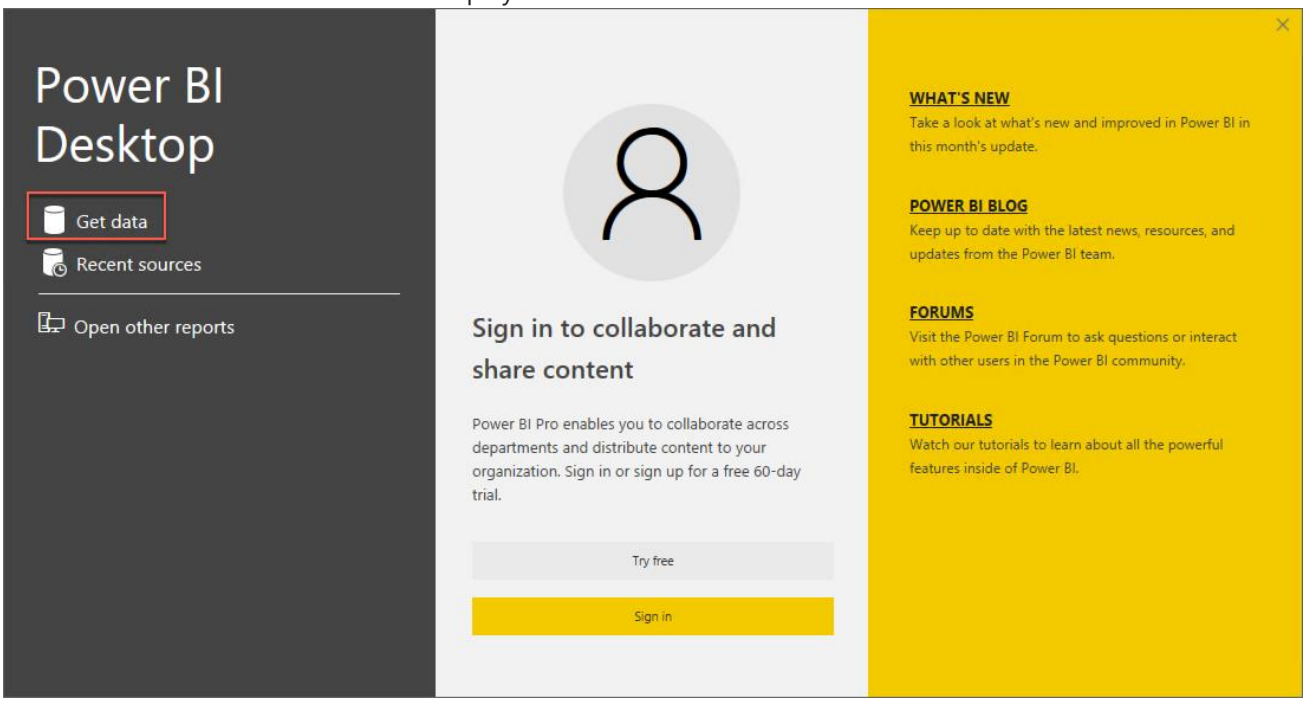
Task 2: Connect to HDInsight Spark using Power BI Desktop

1. On your Lab VM, launch Power BI Desktop by double-clicking on the desktop shortcut you created in the pre-lab setup.

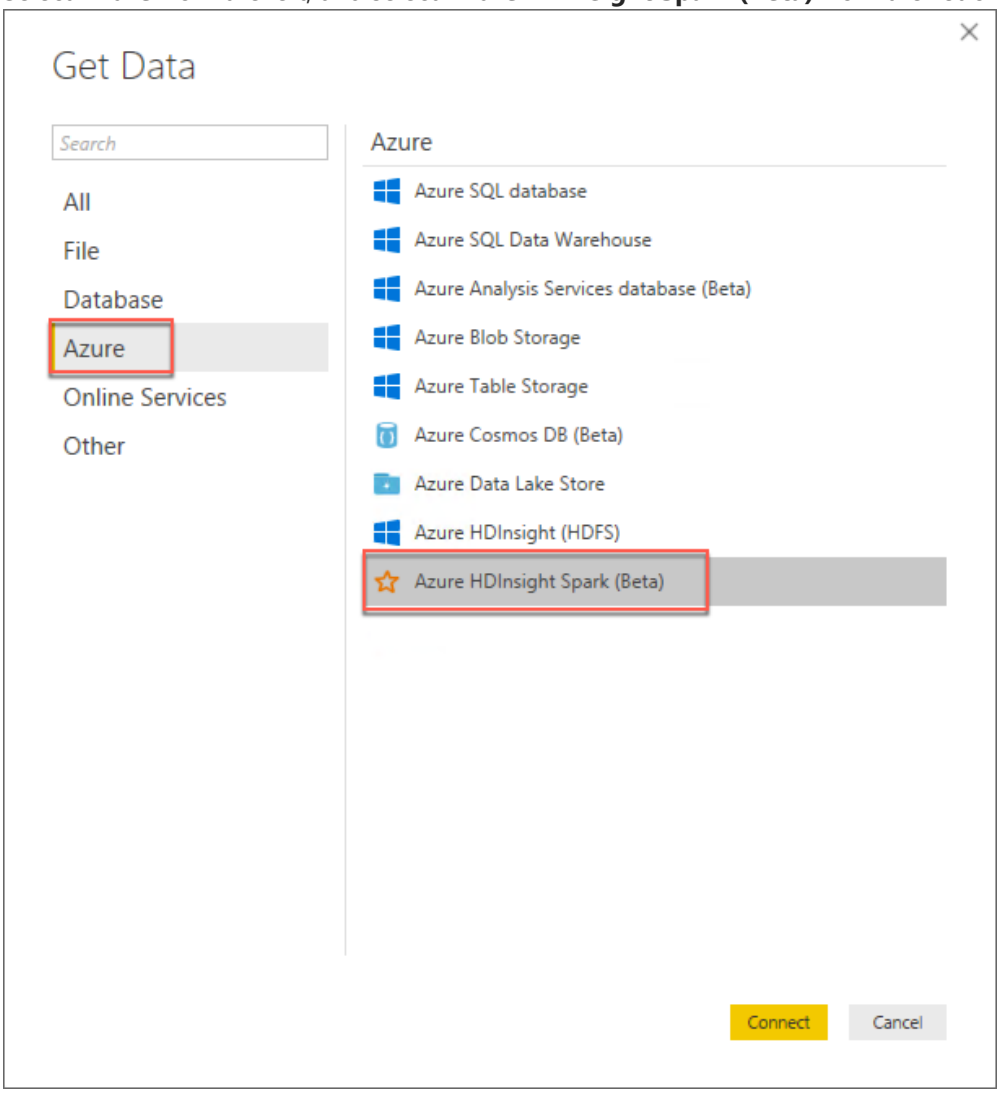
2. When Power BI Desktop opens, you will need to enter your personal information, or Sign in if you already have an account.



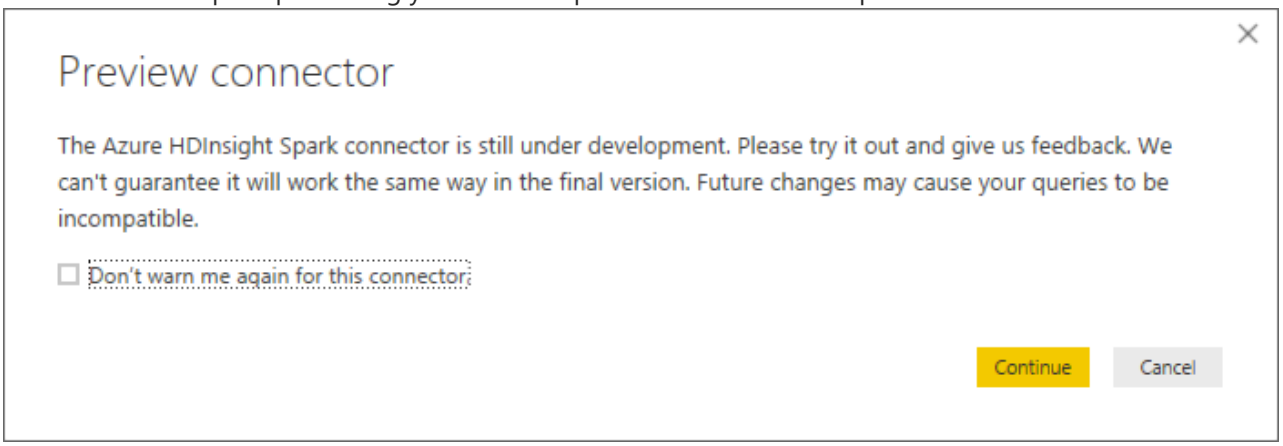
3. Select Get data on the screen that is displayed next.



4. Select **Azure** from the left, and select **Azure HDInsight Spark (Beta)** from the list of available data sources.



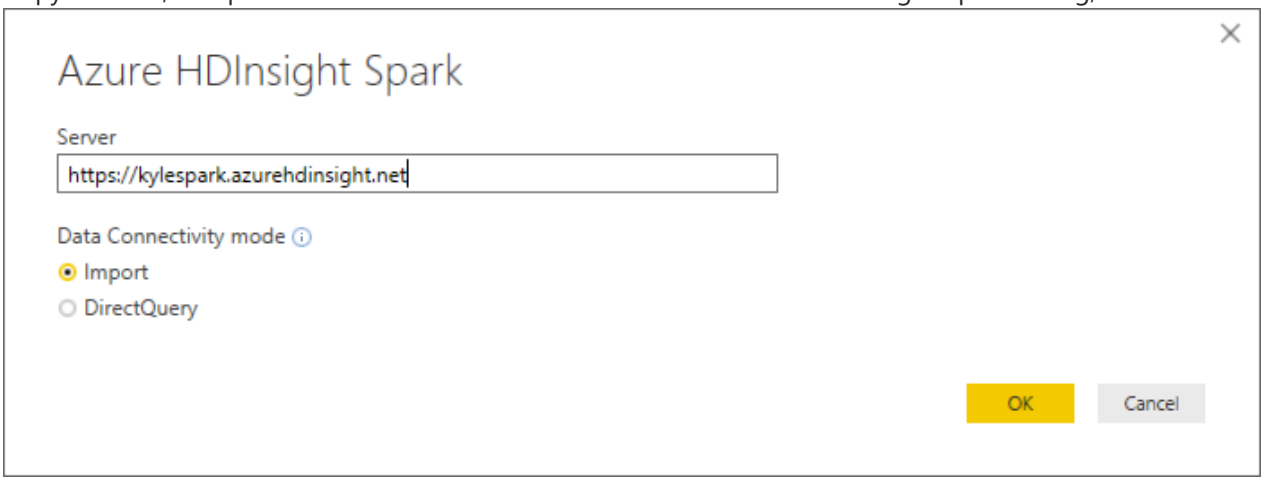
5. Select **Connect**.
6. You will receive a prompt warning you that the Spark connector is still in preview. Select **Continue**.



7. On the next screen, you will be prompted for your HDInsight Spark cluster URL.

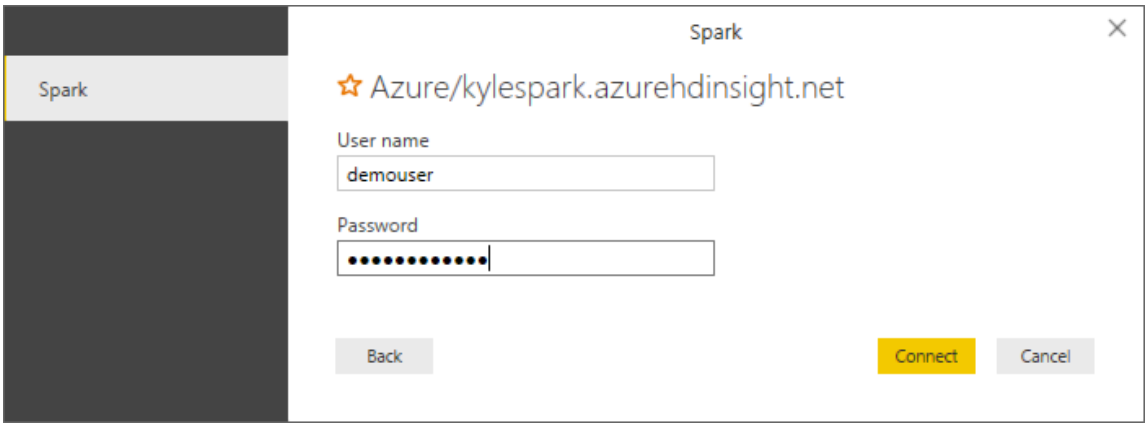
8. To find your Spark cluster URL, go into the Azure portal, and navigate to your Spark cluster, as you did in Exercise 5, Task 1. Once on the cluster blade, look for the URL under the Essentials section.

9. Copy the URL, and paste it into the Server box on the Power BI Azure HDInsight Spark dialog, above.

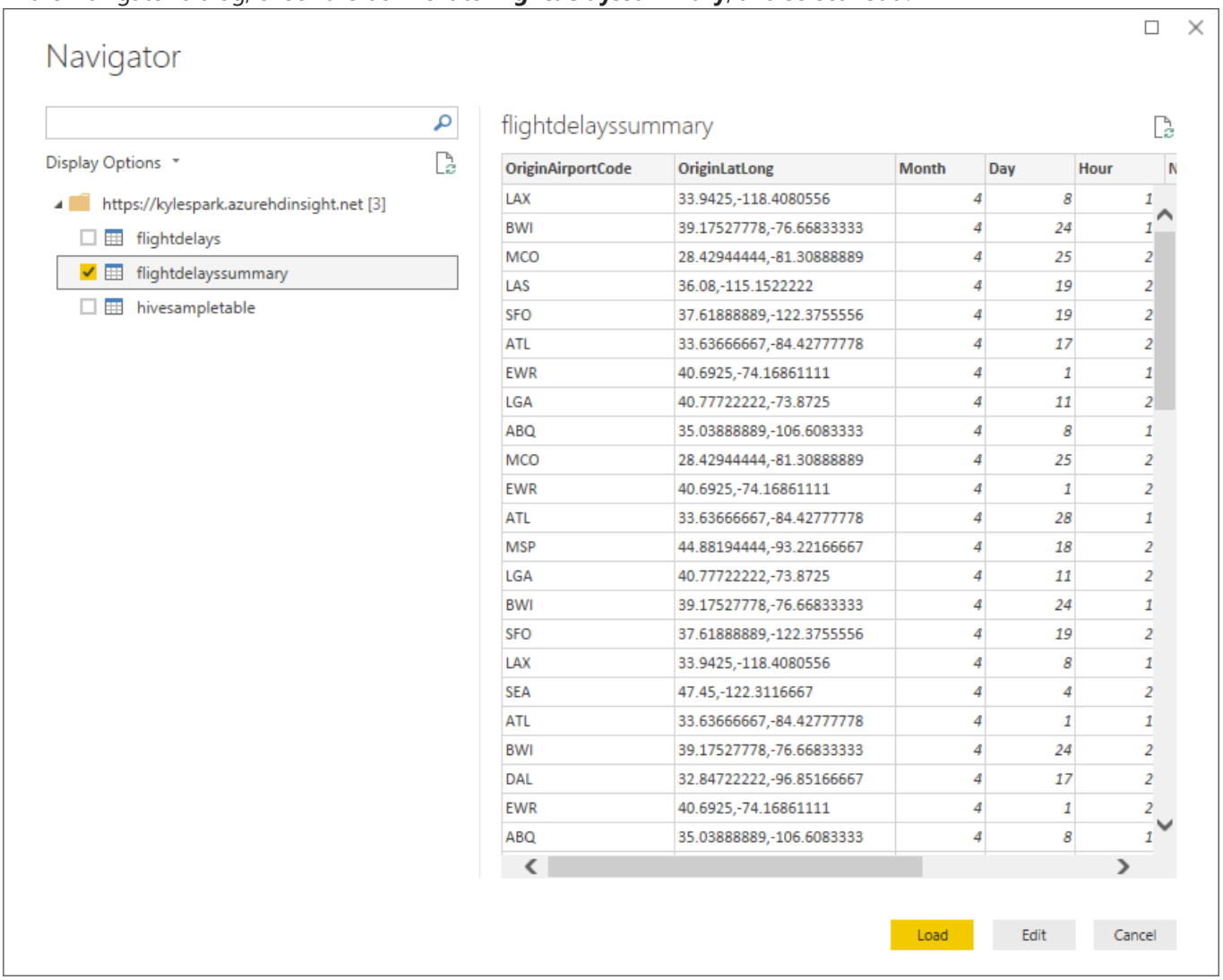


10. Select **OK**.
11. Enter your credentials on the next screen as follows.
    a. User name: demouser

b. Password: Password.1!!



12. Select **Connect**.
13. In the Navigator dialog, check the box next to **flightdelayssummary**, and select **Load**.

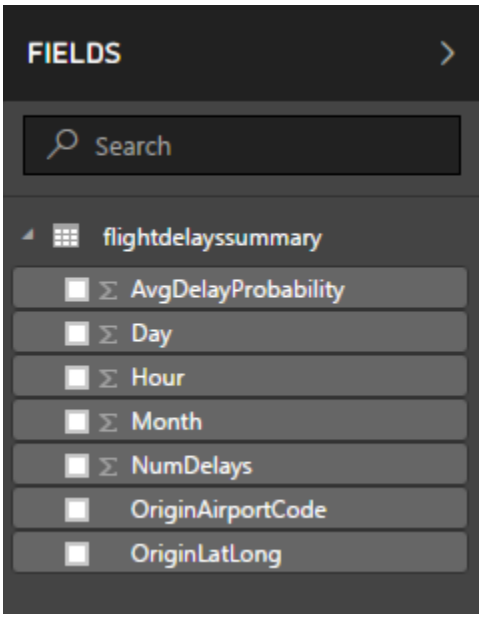


OriginAirportCode	OriginLatLong	Month	Day	Hour
LAX	33.9425,-118.4080556	4	8	1
BWI	39.17527778,-76.66833333	4	24	1
MCO	28.42944444,-81.30888889	4	25	2
LAS	36.08,-115.1522222	4	19	2
SFO	37.61888889,-122.3755556	4	19	2
ATL	33.63666667,-84.42777778	4	17	2
EWR	40.6925,-74.16861111	4	1	1
LGA	40.77722222,-73.8725	4	11	2
ABQ	35.03888889,-106.6083333	4	8	1
MCO	28.42944444,-81.30888889	4	25	2
EWR	40.6925,-74.16861111	4	1	2
ATL	33.63666667,-84.42777778	4	28	1
MSP	44.88194444,-93.22166667	4	18	2
LGA	40.77722222,-73.8725	4	11	2
BWI	39.17527778,-76.66833333	4	24	1
SFO	37.61888889,-122.3755556	4	19	2
LAX	33.9425,-118.4080556	4	8	1
SEA	47.45,-122.3116667	4	4	2
ATL	33.63666667,-84.42777778	4	1	1
BWI	39.17527778,-76.66833333	4	24	2
DAL	32.84722222,-96.85166667	4	17	2
EWR	40.6925,-74.16861111	4	1	2
ABQ	35.03888889,-106.6083333	4	8	1

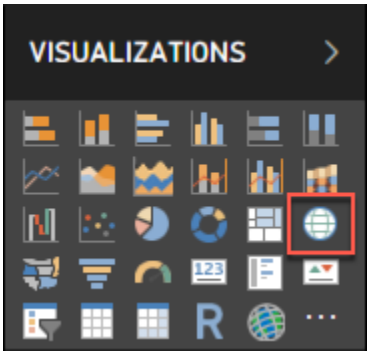
14. It will take several minutes for the data to load into the Power BI Desktop client.

Task 3: Create Power BI report

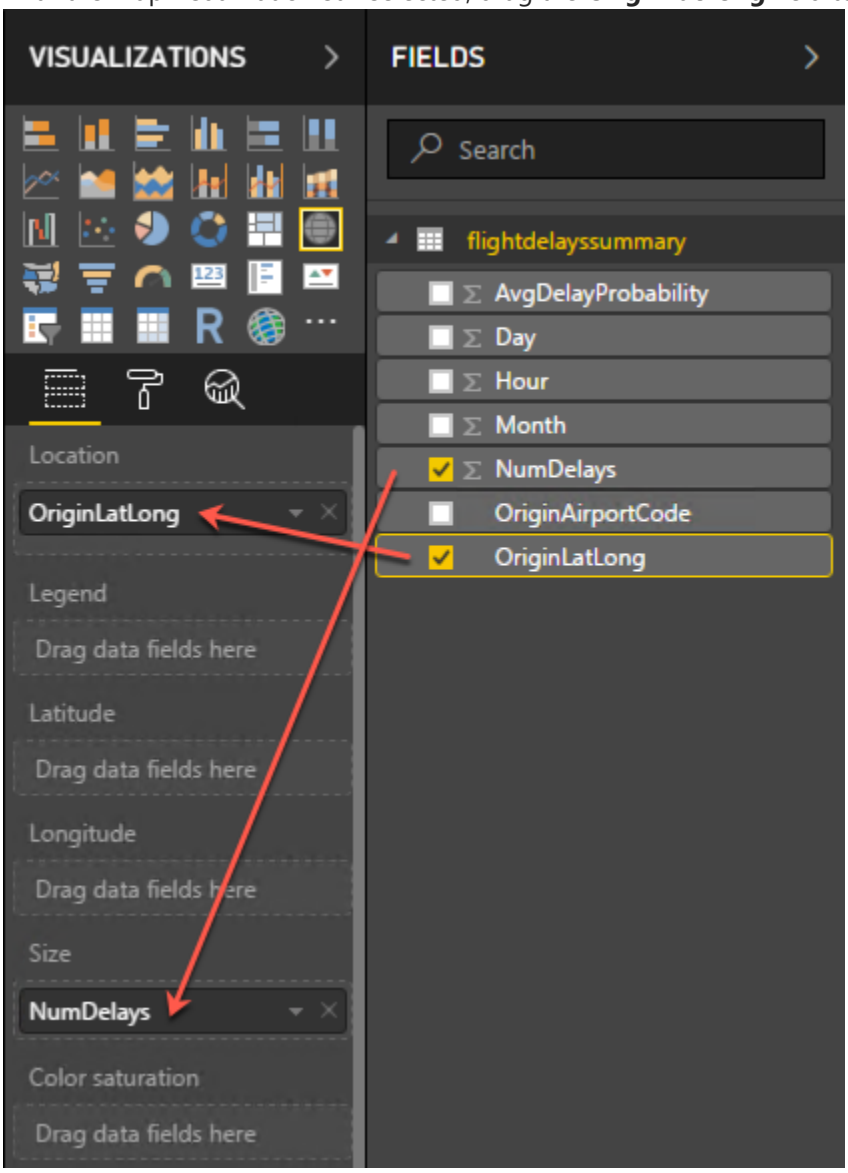
1. Once the data finishes loading, you will see the fields appear on the far right of the Power BI Desktop client window.



2. From the Visualizations area, next to Fields, select the Globe icon to add a Map visualization to the report design surface.

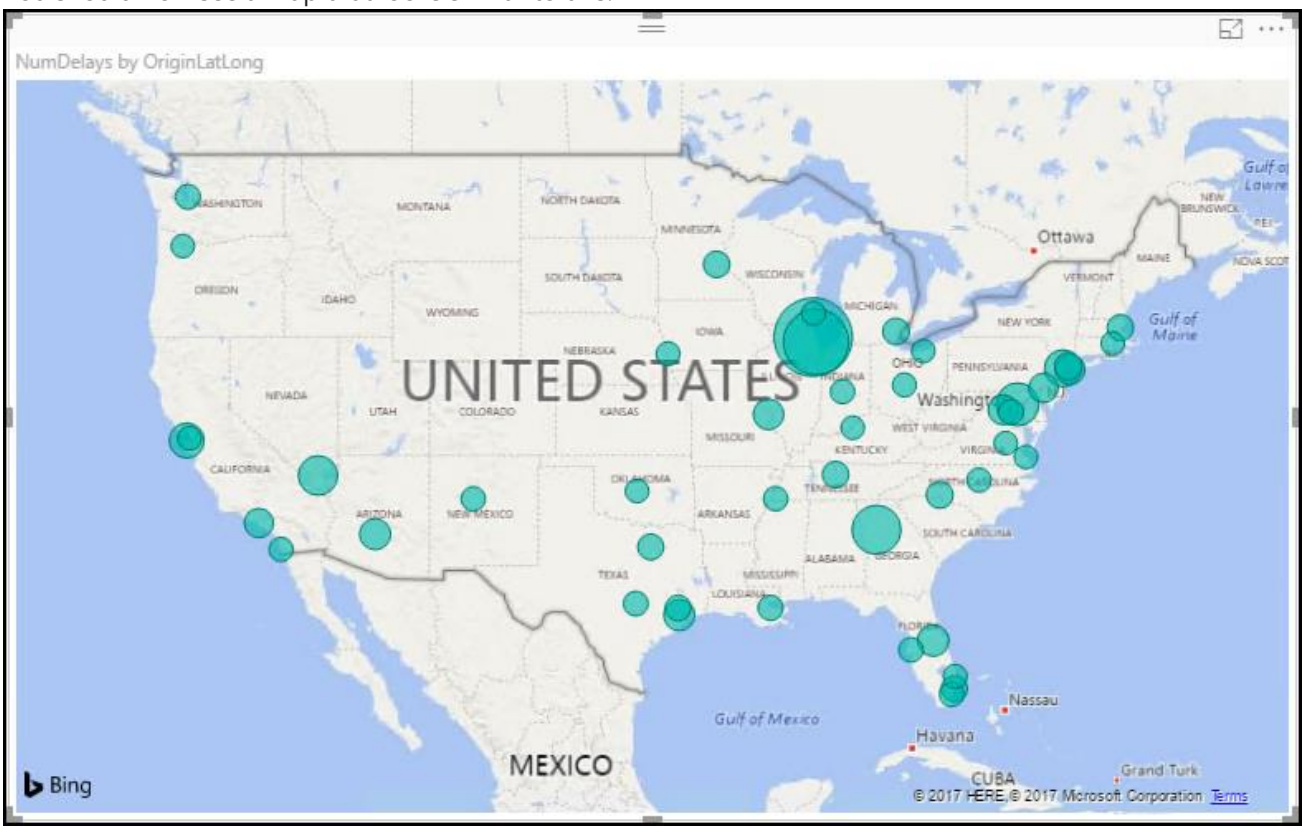


3. With the Map visualization still selected, drag the **OriginLatLong** field to the **Location** field under Visualizations.

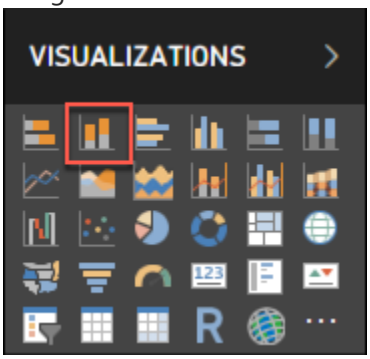


4. Repeat the process, this time dragging the **NumDelays** field to the **Size** field under Visualizations.

5. You should now see a map that looks similar to this:

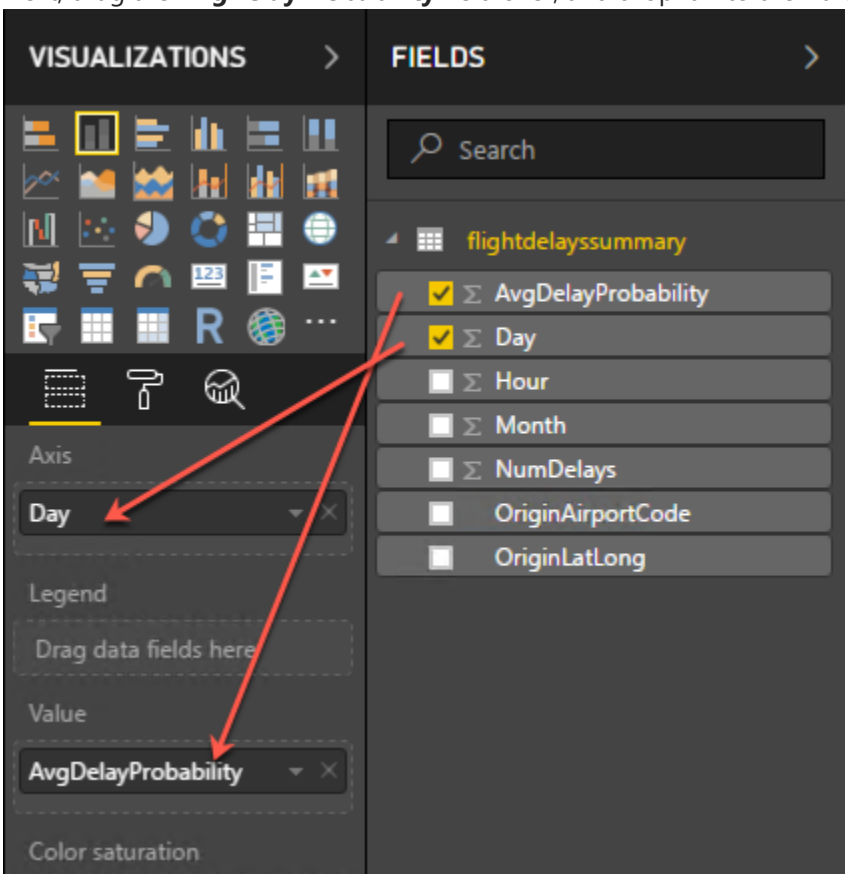


6. Unselect the Map visualization by clicking on the white space next to the map in the report area.
7. From the Visualizations area, select the **Stacked Column Chart** icon to add a bar chart visual to the report's design surface.

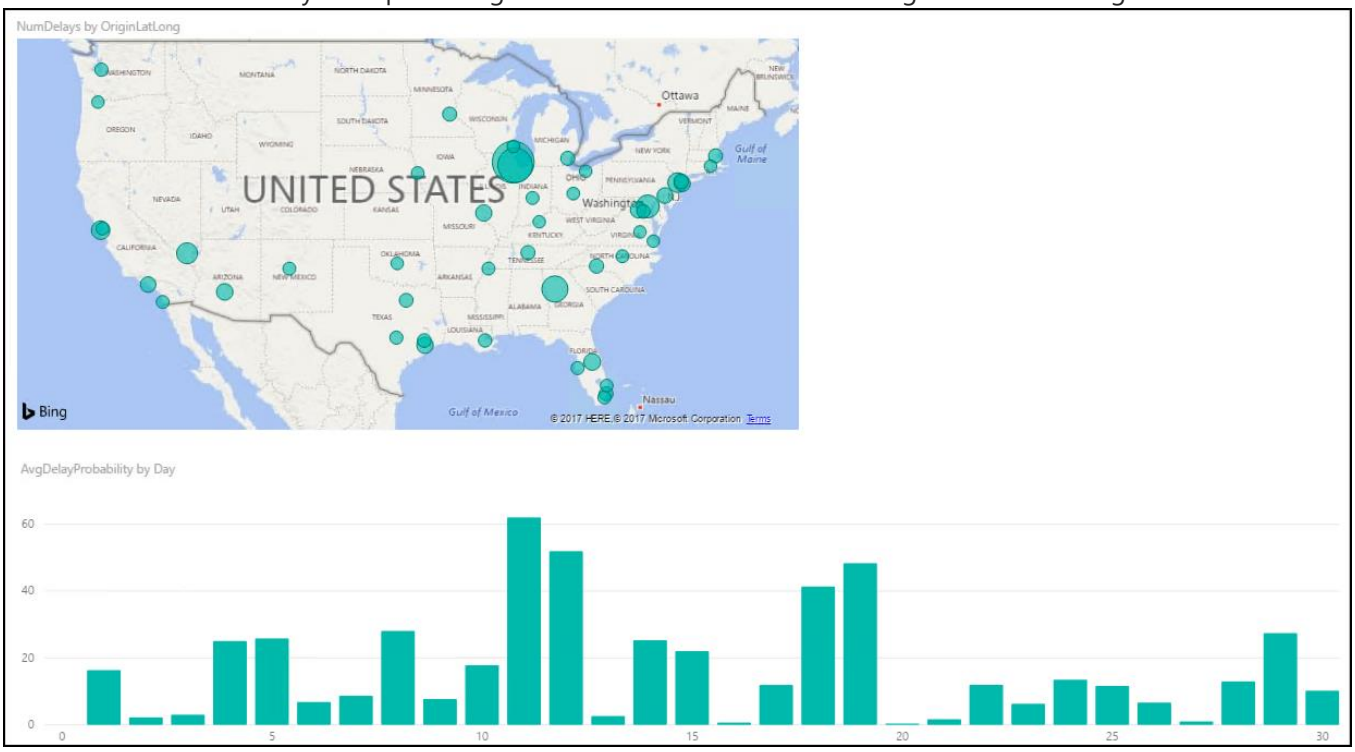


8. With the Stacked Column Chart still selected, drag the **Day** field and drop it into the **Axis** field located under Visualizations.

9. Next, drag the **AvgDelayProbability** field over, and drop it into the **Value** field.

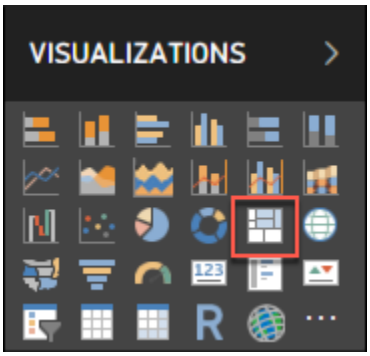


10. Grab the corner of the new Stacked Column Chart visual on the report design surface, and drag it out to make it as wide as the bottom of your report design surface. It should look something like the following.

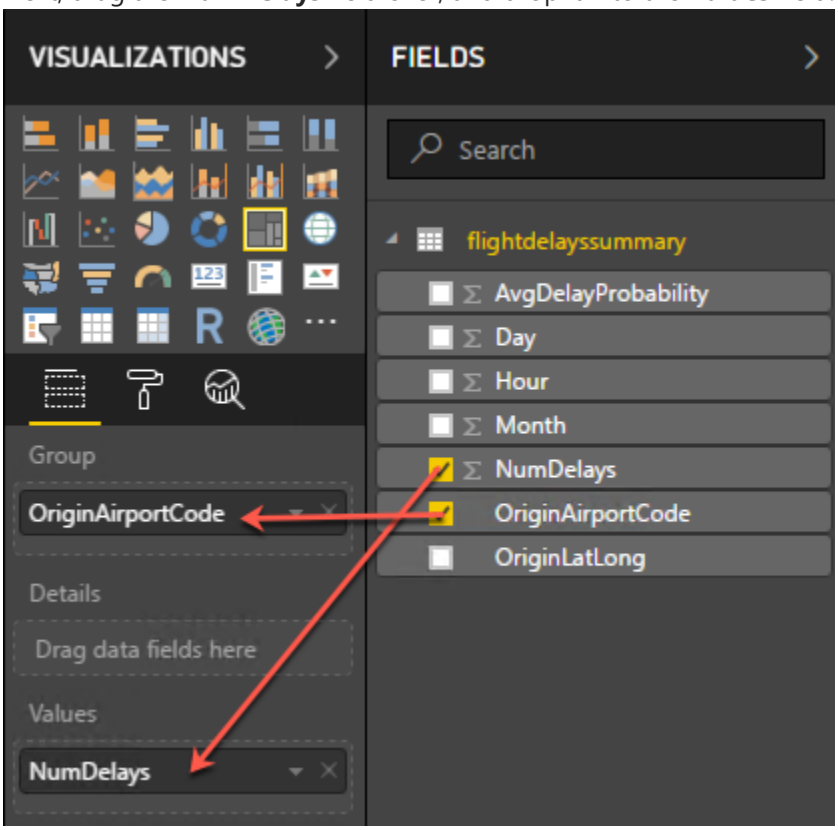


11. Unselect the Stacked Column Chart visual by clicking on the white space next to the map on the design surface.

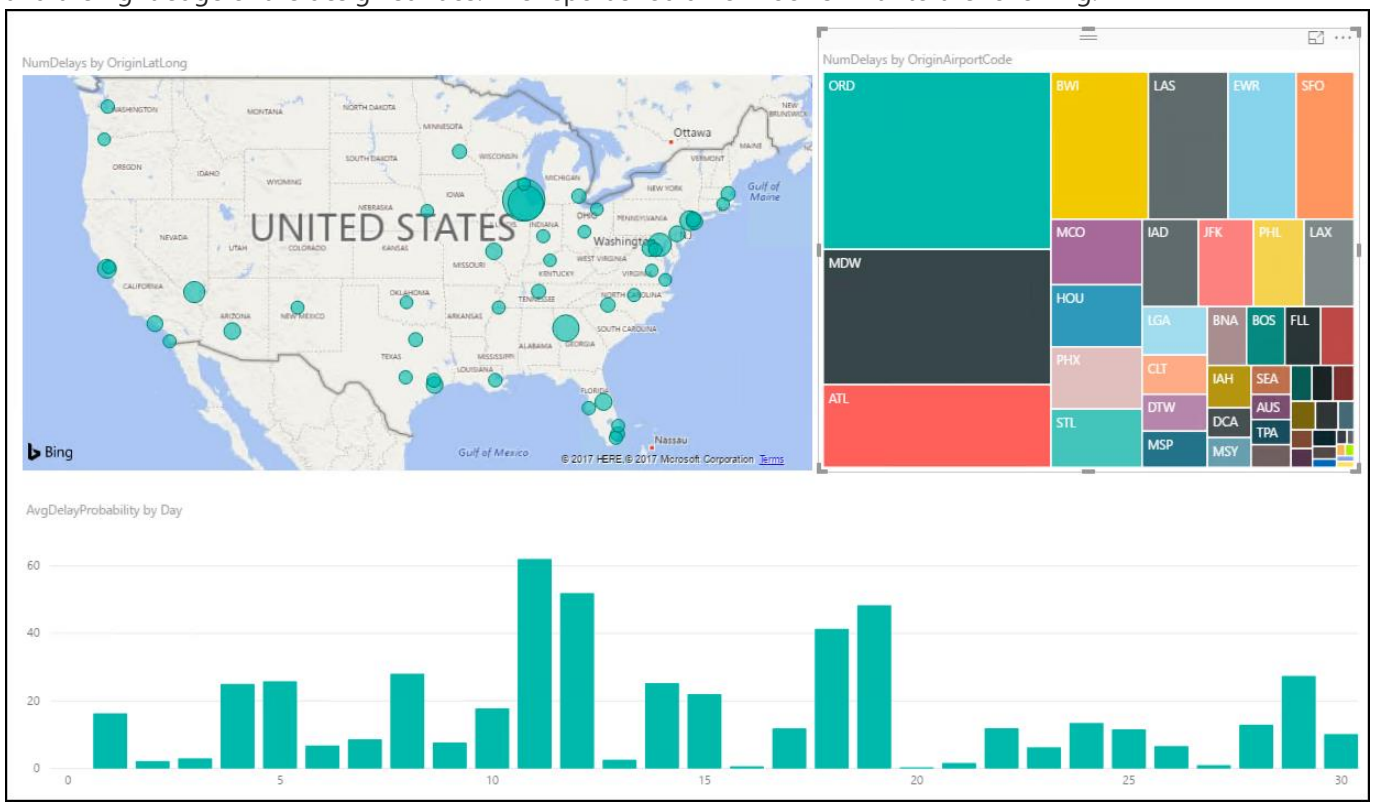
12. From the Visualizations area, select the Treemap icon to add this visualization to the report.



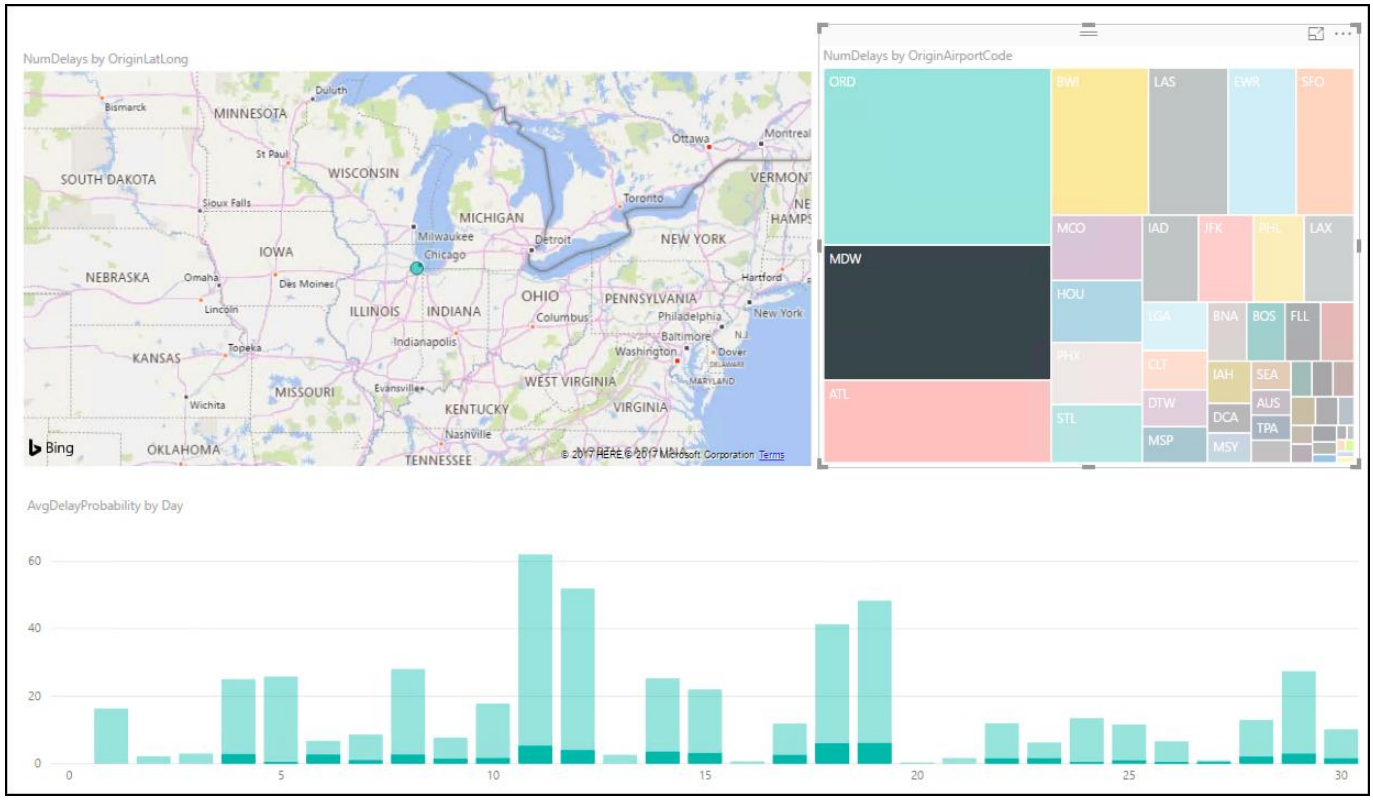
13. With the Treemap visualization selected, drag the **OriginAirportCode** field into the **Group** field under Visualizations.
14. Next, drag the **NumDelays** field over, and drop it into the **Values** field.



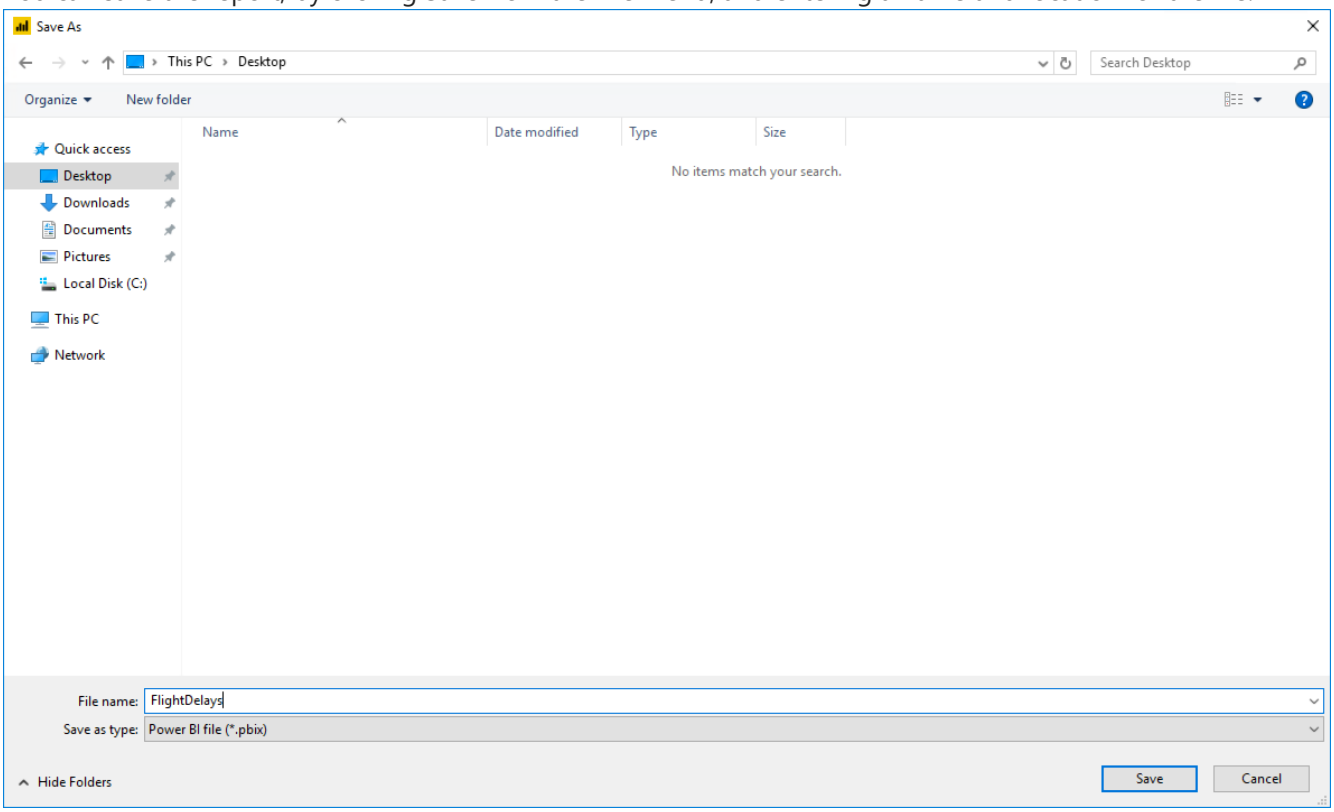
15. Grab the corner of the Treemap visual on the report design surface, and expand it to fill the area between the map and the right edge of the design surface. The report should now look similar to the following.



16. You can cross filter any of the visualizations on the report by clicking on one of the other visuals within the report, as shown below.



17. You can save the report, by clicking Save from the File menu, and entering a name and location for the file.



Exercise 7: Deploy intelligent web app

Duration: 20 minutes

In this exercise, you will deploy an intelligent web application to Azure from GitHub. This application leverages the operationalized machine learning model that was deployed in Exercise 1 to bring action-oriented insight to an already existing business process.

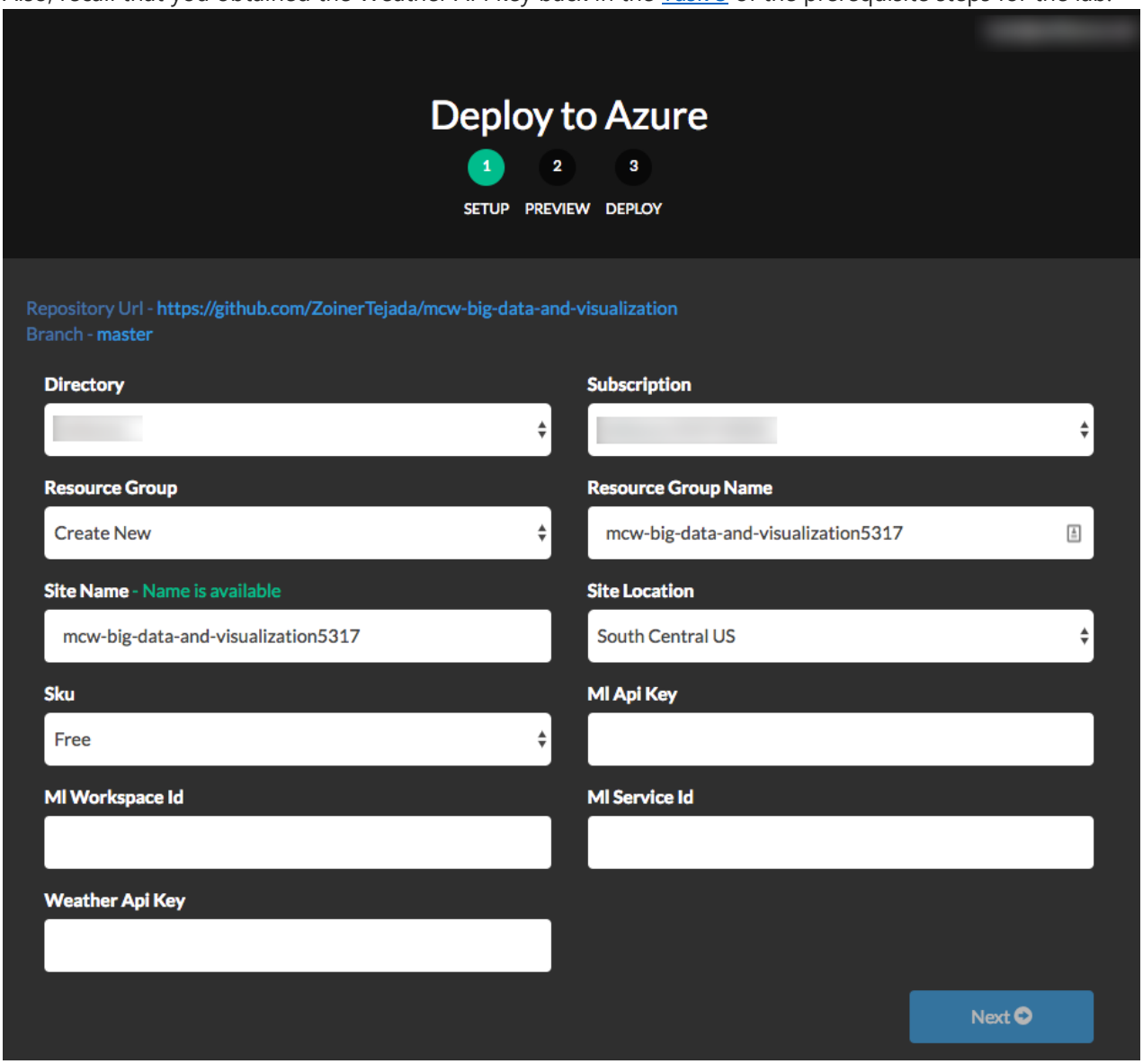
Task 1: Deploy web app from GitHub

1. Navigate to https://github.com/ZoinerTejada/mcw-big-data-and-visualization/blob/master/AdventureWorksTravel/README.md in your browser of choice, but where you are already authenticated to the Azure portal.
2. Read through the README information on the GitHub page.
3. Click the **Deploy to Azure** button.

4. On the following page, ensure the fields are populated correctly.
    a. Ensure the correct Directory and Subscription are selected.
    b. Select the Resource Group that you have been using throughout this lab.
    c. Either keep the default Site name, or provide one that is globally unique, and then choose a Site Location.
    d. Finally, enter the ML API and Weather API information.
        i. Recall that you recorded the ML API information back in Exercise 1, Task 9.
            1. This information can be obtained on your Machine Learning web service page (https://services.azureml.net, then go to the Consume tab.
            2. The Primary Key listed is your ML API key

3. In the Request-Response URL, the GUID after subscriptions/ is your ML Workspace Id
4. In the Request-Response URL, the GUID after services/ is your ML Service Id

e. Also, recall that you obtained the Weather API key back in the Task 3 of the prerequisite steps for the lab.



5. Select **Next**, and on the following screen, select **Deploy**.
6. The page should begin deploying your application while showing you a status of what is currently happening.

NOTE: If you run into errors during the deployment that indicate a bad request or unauthorized, verify that the user you are logged into the portal with is either a Service Administrator or a Co-Administrator.

7. After a short time, the deployment will complete, and you will be presented with a link to your newly deployed web application. CTRL+Click to open it in a new tab.
8. Try a few different combinations of origin, destination, date, and time in the application. The information you are shown is the result of both the ML API you published, as well as information retrieved from the Weather Underground API.
9. Congratulations! You have built and deployed an intelligent system to Azure.

After the hands-on lab

Duration: 10 mins

In this exercise, attendees will deprovision any Azure resources that were created in support of the lab.

Task 1: Delete resource group

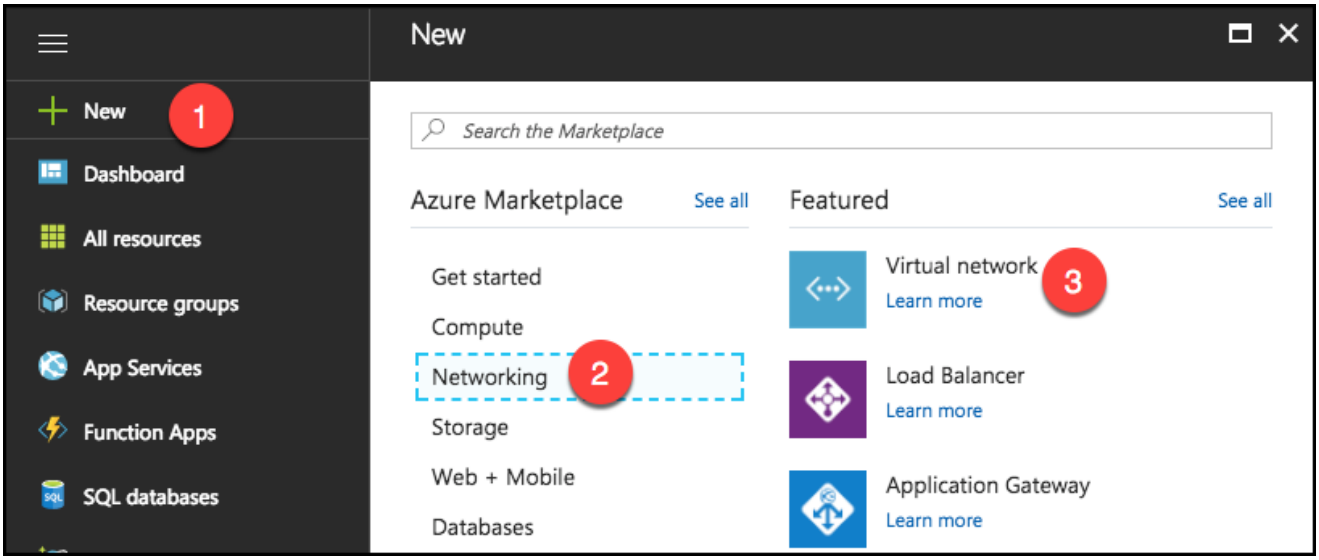
1. Using the Azure portal, navigate to the Resource group you used throughout this hands-on lab by selecting **Resource groups** in the left menu.
2. Search for the name of your research group and select it from the list.
3. Select **Delete** in the command bar and confirm the deletion by re-typing the Resource group name and selecting **Delete**.

Appendix A

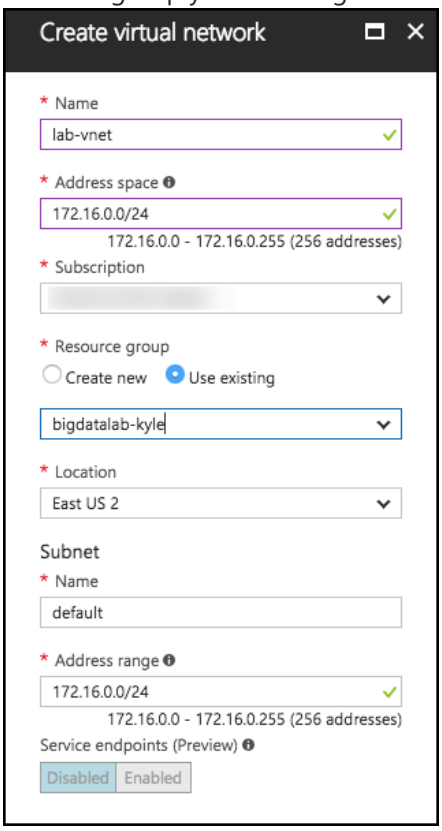
Appendix A outlines the detailed steps involved in manually creating the resources provisioned by the Lab ARM template. The ARM template creates a virtual network, multiple storage accounts, an ML Workspace, and an HDInsight Spark cluster. The workspaces will be provisioned as part of the creation of the ML Workspace and HDInsight cluster.

Create virtual network

1. To create the Virtual Network, navigate to the Azure portal, and select **+New**, **Networking**, and select **Virtual Network**.



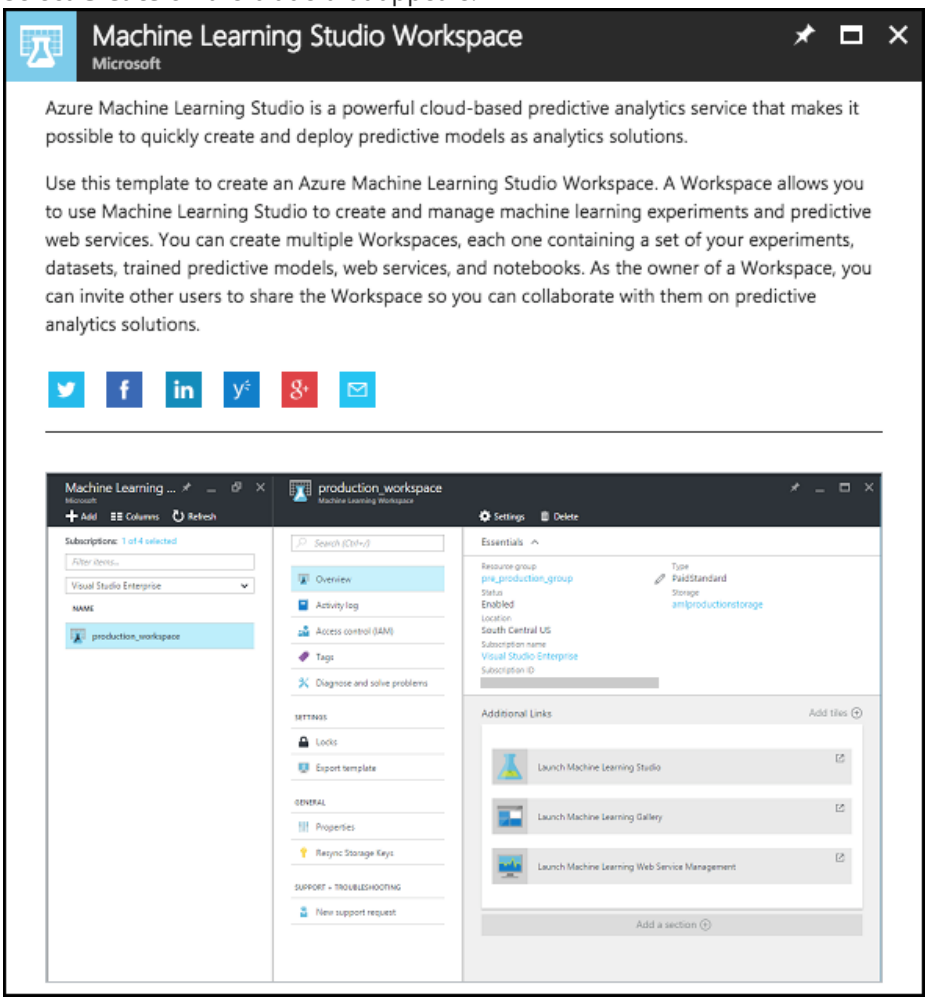
2. On the Create virtual network blade, enter a name, such as lab-vnet, select your subscription, and choose the resource group you are using for this lab.



3. Select **Create** to provision the virtual network.

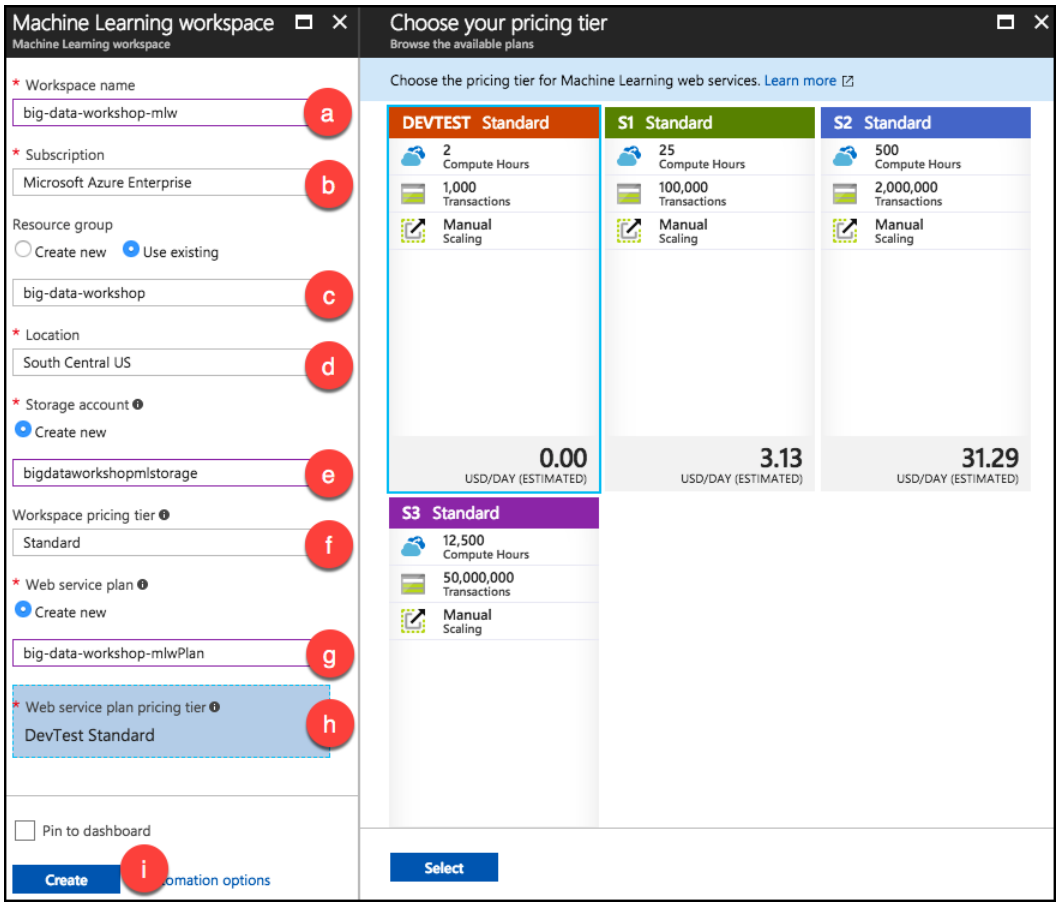
Create ML workspace & associated storage

1. Login to the Azure portal (https://portal.azure.com).
2. Select **+New**, then search for Machine Learning Studio Workspace, and press enter.
3. Select **Create** on the blade that appears.



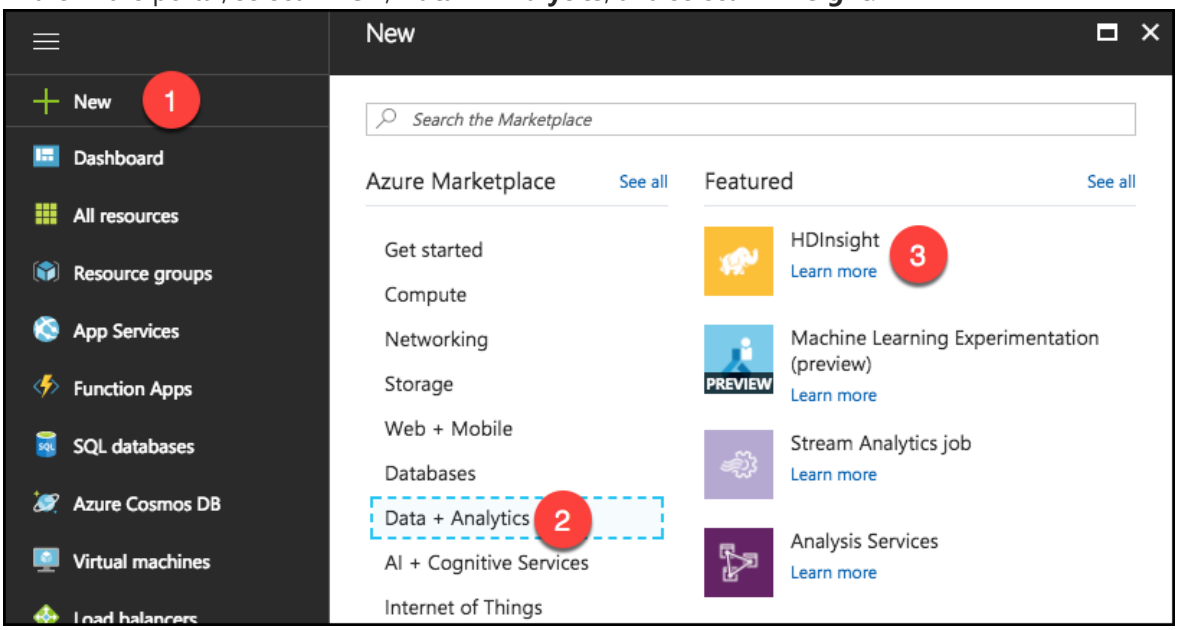
4. On the Machine Learning Workspace blade, enter the following:
    a. Workspace name: Enter a name, such as **bigdatalab-kyle**
    b. Subscription: Select your subscription
    c. Resource group: Create a new resource group for the lab, or select one you are already using.
    d. Location: Azure ML is only available in a subset of regions (Japan East, South Central US, Southeast Asiz, West Central US and West Europe), but for this hackathon it does not have to be in the same region as your other services. The ARM template uses South Central US, by default.
    e. Storage account: Create new is the only option, so enter a name, or accept the default name already provided
    f. Workspace pricing tier: Select **Standard**
    g. Web service plan: Select Create new, and enter a name for the web service plan
    h. Web service plan pricing tier: Select **DEVTEST Standard**, and click **Select**

i. Click **Create**



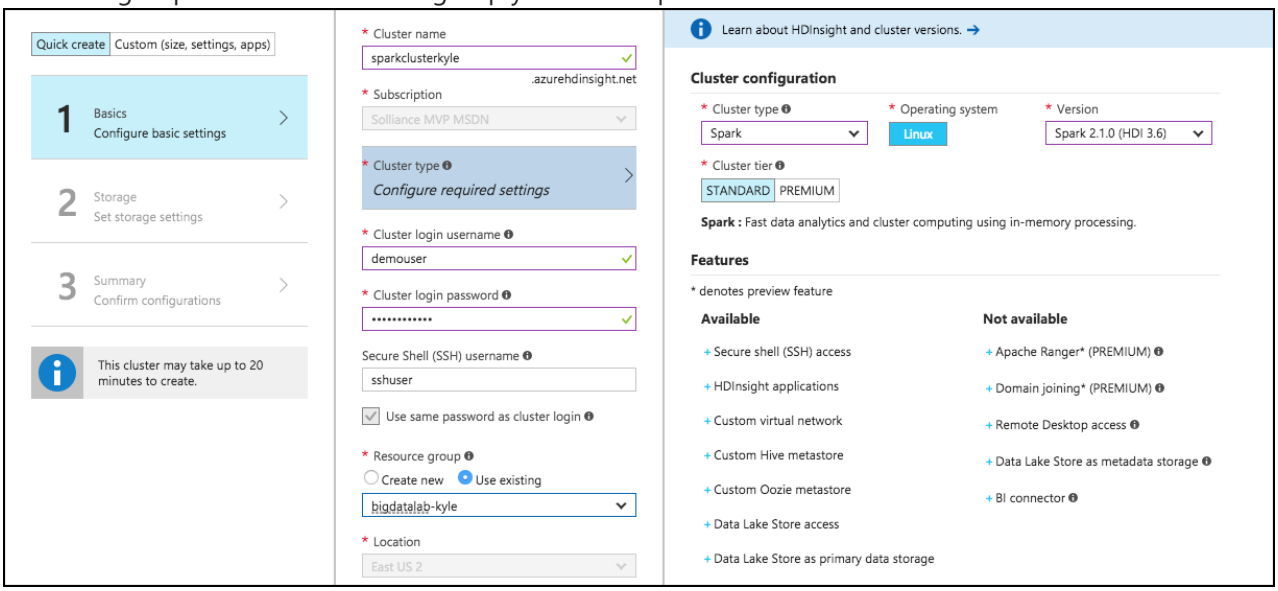
Provision HDInsights Spark Cluster & associated storage

1. In the Azure portal, select **+New**, **Data + Analytics**, and select **HDInsight**.



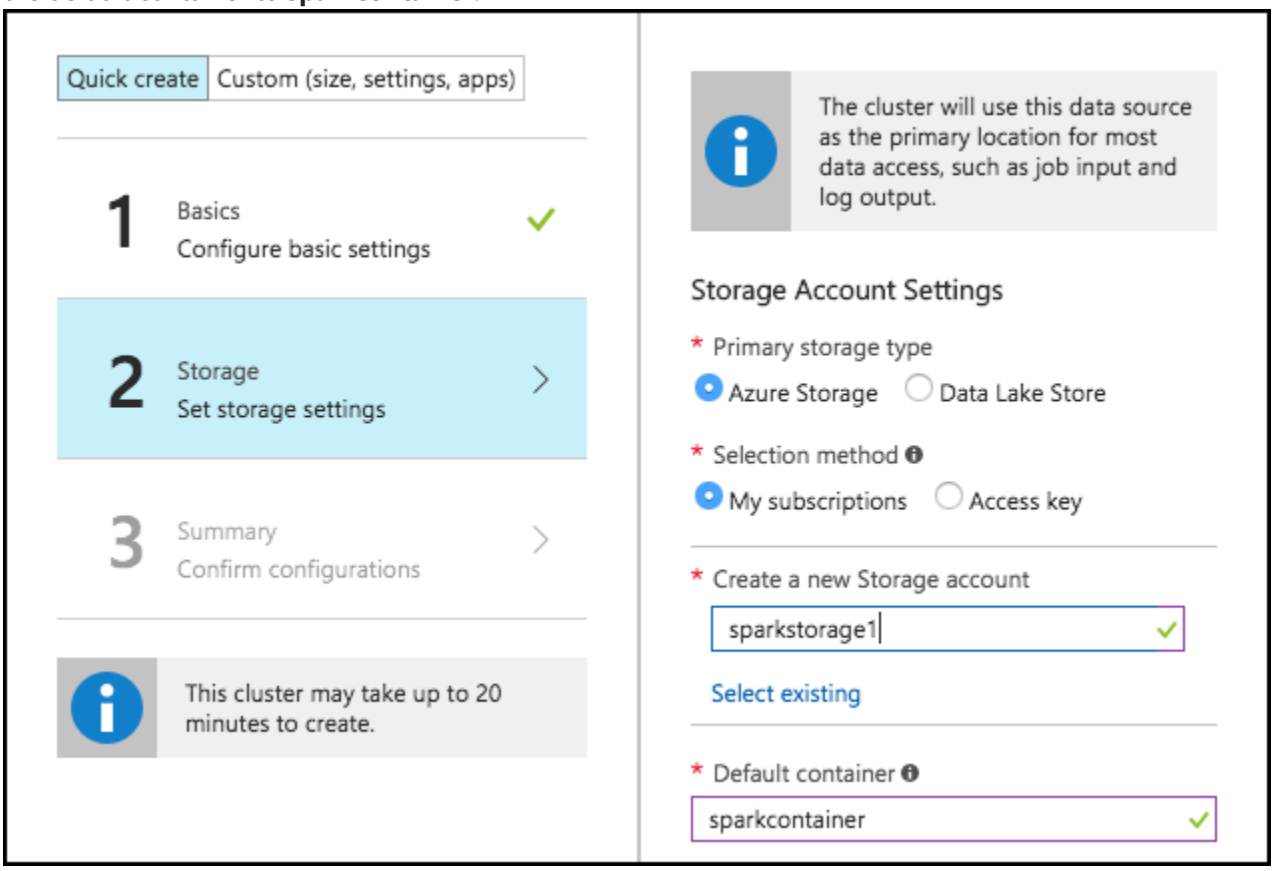
2. On the Basics blade, enter the following:
   a. Name: enter a unique cluster name, such as sparkclusterkyle. Include "sparkcluster" in the name, as it will be referred to within the lab by that name.

b. Subscription: Select your subscription.
c. Cluster type: Select Configure required settings, and select Spark for the Cluster type, and Spark 2.1.0 (HDI 3.6) for the Version. Click **Select** on the Cluster Configuration blade.
d. Cluster login username: Enter **demouser**
e. Cluster login password: Enter **Password.1!!**
f. Resource group: Select the resource group you used to provision the other resources above.



g. Select **Next** to move on to the Storage configuration.

3. On the Storage blade, create a new storage account for the cluster, named something like **sparkstorage**, and set the default container to **sparkcontainer**.



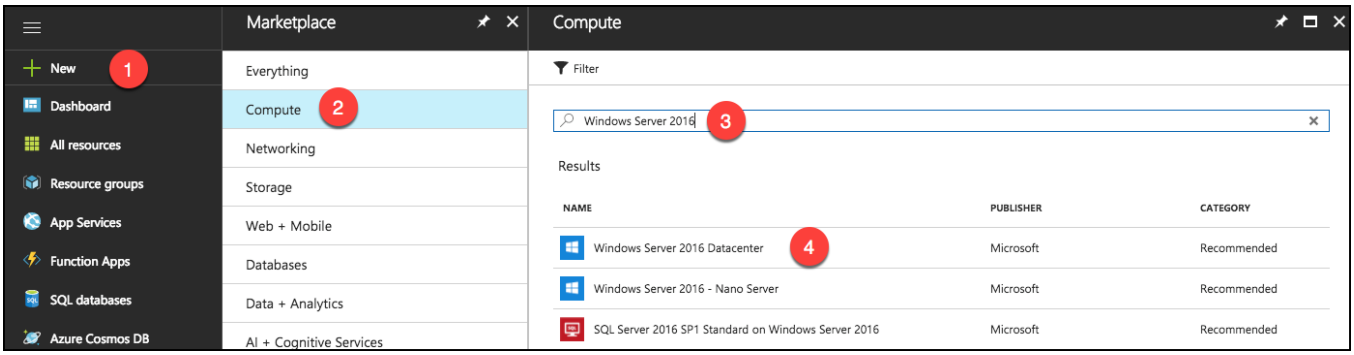
4. Select **Next** to move on to the summary blade.
5. On the summary blade, select **Create** to provision the cluster and its associated storage account.
6. Note: It may take up to 20 minutes to provision the cluster.

Appendix B

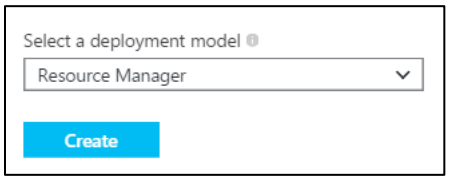
Appendix B outlines the detailed steps involved in manually creating the resources provisioned by the Lab VM ARM template.

Setup a lab virtual machine (VM)

1. In the Azure Portal, select **+NEW**, **Compute,** select **See all**, then type "Windows Server 2016" into the search bar. Select **Windows Server 2016 Datacenter** from the results**.**

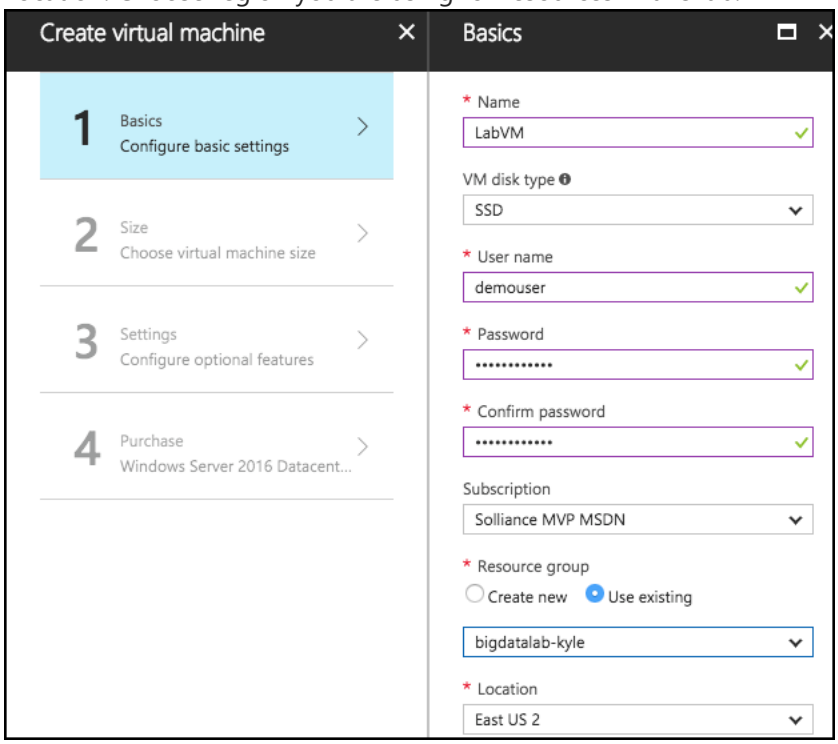


2. On the blade that comes up, at the bottom, ensure the deployment model is set to **Resource Manager** and select **Create**.

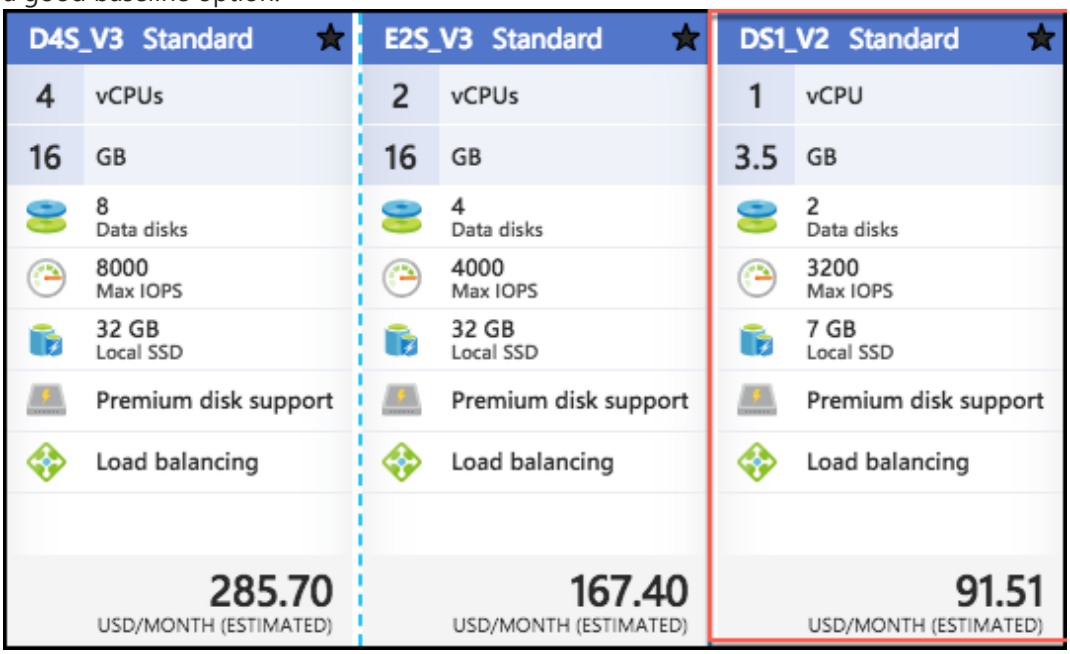


3. Set the following configuration on the Basics tab.
    - Name: LabVM
    - VM disk type: SSD
    - User name: demouser
    - Password: Password.1!!
    - Subscription: If you have multiple subscriptions choose the subscription to execute your labs in.
    - Resource Group: Select the resource group you are using for this lab.

- Location: Choose region you are using for resources in this lab.

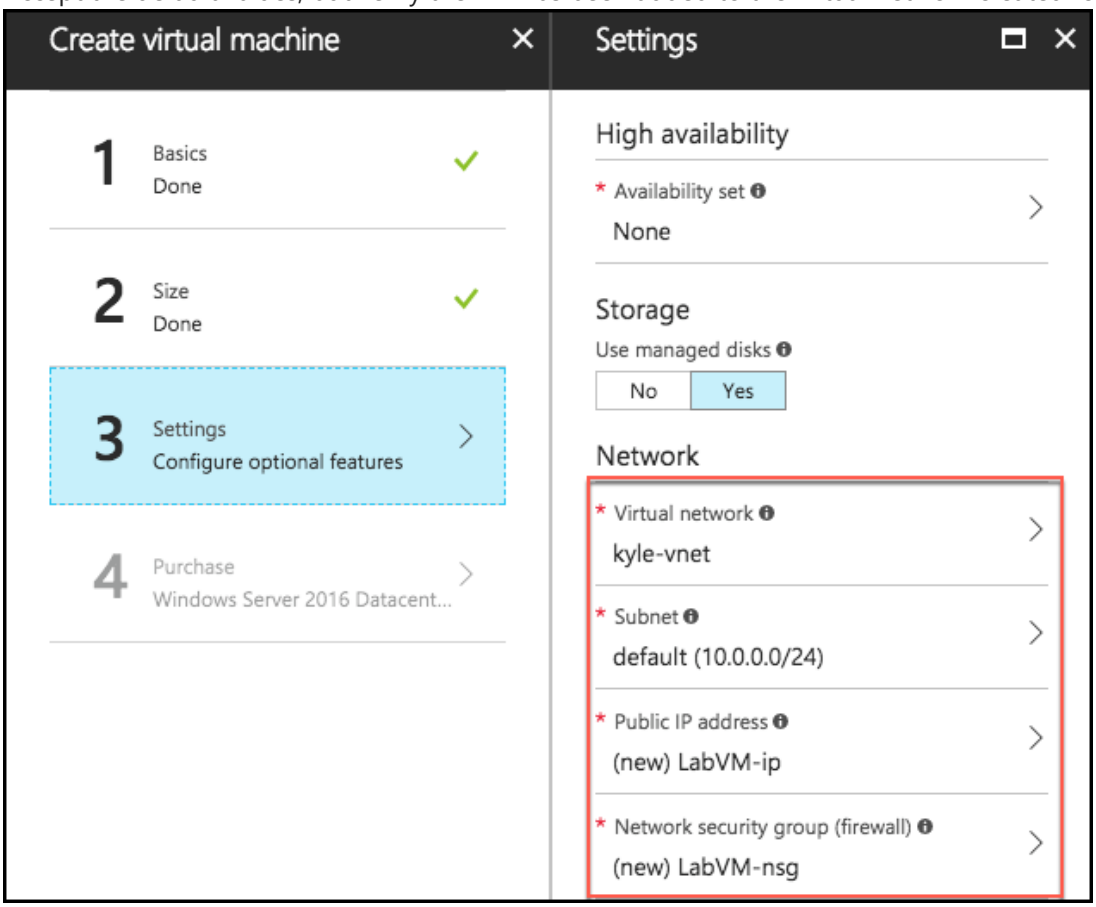


4. Select **OK** to move to the next step.
5. Select the instance size on the Size blade. This machine won't be doing much heavy lifting, so **DS1_V2 Standard** is a good baseline option.

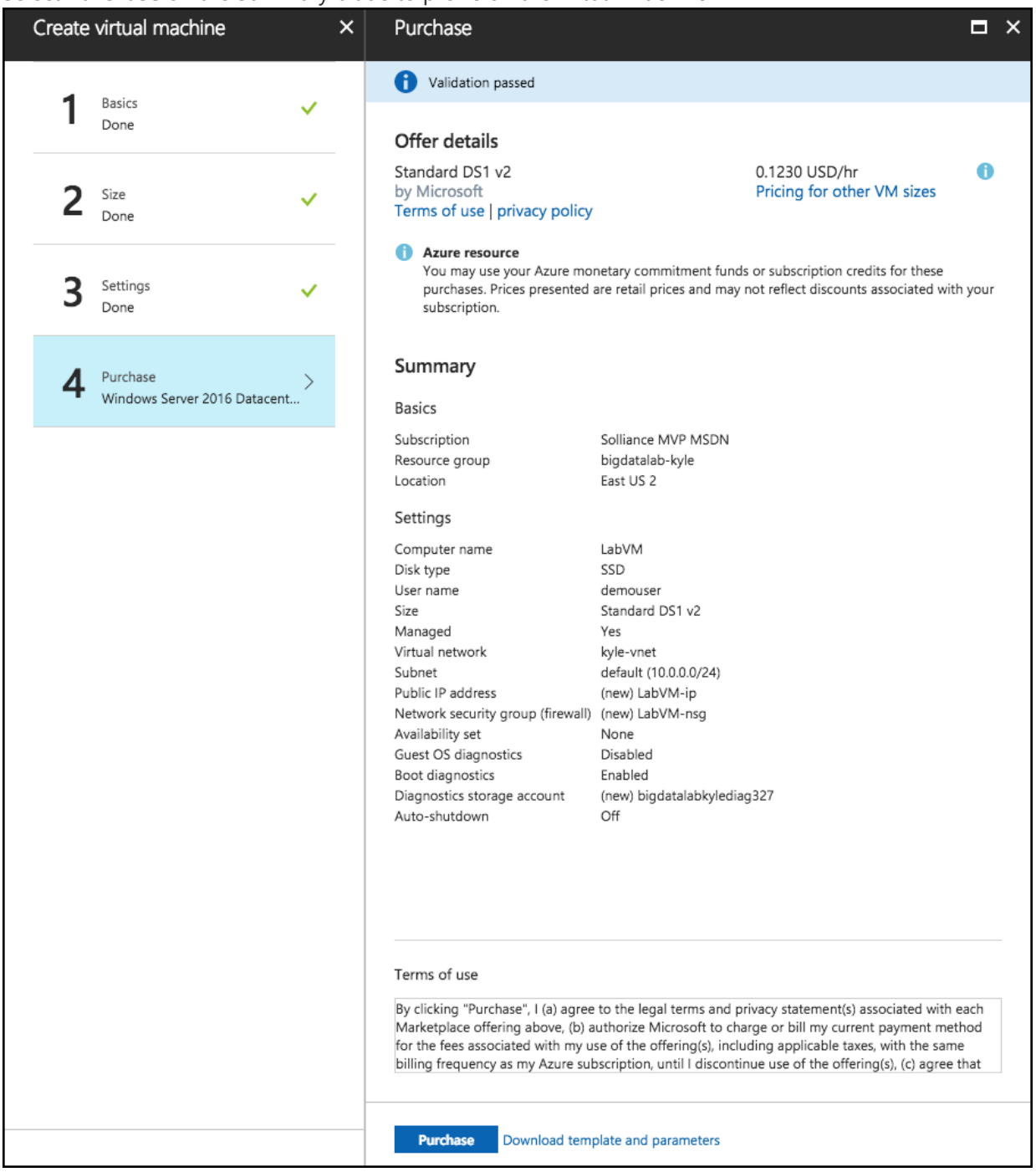


6. Click **Select** to move on to the Settings blade.

7. Accept the default values, but verify the VM has been added to the virtual network created for this lab. Select **OK**.



8. Select **Purchase** on the Summary blade to provision the virtual machine



9. It may take 10+ minutes for the virtual machine to complete provisioning.