**AUTOMATE AN AZURE DATABRICKS NOTEBOOK IN AZURE DATA FACTORY**

Create an Azure Data Factory resource

In addition to your Azure Databricks workspace, you will need to provision an Azure Data Factory resource in your subscription.

1. In the Azure portal, close the cloud shell pane and browse to the ***msl-*xxxxxxx**\* resource group created by the setup script (or the resource group containing your existing Azure Databricks workspace).
2. In the toolbar, select **+ Create** and search for Data Factory. Then create a new **Data Factory** resource with the following settings:
   * **Subscription**: *Your subscription*
   * **Resource group**: msl-*xxxxxxx* (or the resource group containing your existing Azure Databricks workspace)
   * **Name**: *A unique name, for example****adf-xxxxxxx***
   * **Region**: *The same region as your Azure databricks workspace (or any other available region if this one is not listed)*
   * **Version**: V2
3. When the new resource is created, verify that the resource group contains both the Azure Databricks workspace and Azure Data Factory resources.

Create a notebook

You can create notebooks in your Azure Databricks workspace to run code written in a range of programming languages. In this exercise, you’ll create a simple notebook that ingests data from a file and saves it in a folder in Databricks File System (DBFS).

1. In the Azure portal, browse to the **msl-*xxxxxxx*** resource group that was created by the script (or the resource group containing your existing Azure Databricks workspace)
2. Select your Azure Databricks Service resource (named **databricks-*xxxxxxx*** if you used the setup script to create it).
3. In the **Overview** page for your workspace, use the **Launch Workspace** button to open your Azure Databricks workspace in a new browser tab; signing in if prompted.
4. View the Azure Databricks workspace portal and note that the sidebar on the left side contains icons for the various tasks you can perform.
5. In the sidebar, use the **(+) New** link to create a **Notebook**.
6. Change the default notebook name (**Untitled Notebook *[date]***) to **Process Data**.
7. In the first cell of the notebook, enter (but don’t run) the following code to set up a variable for the folder where this notebook will save data.

|  |
| --- |
| # Use dbutils.widget define a "folder" variable with a default value  dbutils.widgets.text("folder", "data")    # Now get the parameter value (if no value was passed, the default set above will be used)  folder = dbutils.widgets.get("folder") |

Under the existing code cell, use the **+** icon to add a new code cell. Then in the new cell, enter (but don’t run) the following code to download data and save it to the folder:

|  |
| --- |
| import urllib3    # Download product data from GitHub  response = urllib3.PoolManager().request('GET', 'https://raw.githubusercontent.com/MicrosoftLearning/mslearn-databricks/main/data/products.csv')  data = response.data.decode("utf-8")    # Save the product data to the specified folder  path = "dbfs:/{0}/products.csv".format(folder)  dbutils.fs.put(path, data, True) |

In the sidebar on the left, select **Workspace** and ensure your **Process Data** notebooks is listed. You’ll use Azure data Factory to run the notebook as part of a pipeline.

Enable Azure Databricks integration with Azure Data Factory

To use Azure Databricks from an Azure Data Factory pipeline, you need to create a linked service in Azure Data Factory that enables access to your Azure Databricks workspace.

Generate an access token

1. In the Azure Databricks portal, at on the top right menu bar, select the username and then select **User Settings** from the drop-down.
2. In the **User Settings** page, select **Developer**. Then next to **Access tokens** select **Manage**.
3. Select **Generate new token** and generate a new token with the comment *Data Factory* and a blank lifetime (so the token doesn’t expire). Be careful to **copy the token when it is displayed before selecting *Done***.
4. Paste the copied token to a text file so you have it handy for later in this exercise.

Create a linked service in Azure Data Factory

1. Return to the Azure portal, and in the **msl-*xxxxxxx*** resource group, select the **adf*xxxxxxx*** Azure Data Factory resource.
2. On the **Overview** page, select the **Launch studio** to open the Azure Data Factory Studio. Sign in if prompted.
3. In Azure Data Factory Studio, use the **»** icon to expand the navigation pane on the left. Then select the **Manage** page.
4. On the **Manage** page, in the **Linked services** tab, select **+ New** to add a new linked service.
5. In the **New linked service** pane, select the **Compute** tab at the top. Then select **Azure Databricks**.
6. Continue, and create the linked service with the following settings:
   * **Name**: AzureDatabricks
   * **Description**: Azure Databricks workspace
   * **Connect via integration runtime**: AutoResolveInegrationRuntime
   * **Account selection method**: From Azure subscription
   * **Azure subscription**: *Select your subscription*
   * **Databricks workspace**: *Select your****databricksxxxxxxx****workspace*
   * **Select cluster**: New job cluster
   * **Databrick Workspace URL**: *Automatically set to your Databricks workspace URL*
   * **Authentication type**: Access token
   * **Access token**: *Paste your access token*
   * **Cluster version**: 13.3 LTS (Spark 3.4.1, Scala 2.12)
   * **Cluster node type**: Standard\_DS3\_v2
   * **Python version**: 3
   * **Worker options**: Fixed
   * **Workers**: 1

Use a pipeline to run the Azure Databricks notebook

Now that you have created a linked service, you can use it in a pipeline to run the notebook you viewed previously.

Create a pipeline

1. In Azure Data Factory Studio, in the navigation pane, select **Author**.
2. On the **Author** page, in the **Factory Resources** pane, use the **+** icon to add a **Pipeline**.
3. In the **Properties** pane for the new pipeline, change its name to **Process Data with Databricks**. Then use the **Properties** button (which looks similar to **🗏\***) on the right end of the toolbar to hide the **Properties** pane.
4. In the **Activities** pane, expand **Databricks** and drag a **Notebook** activity to the pipeline designer surface.
5. With the new **Notebook1** activity selected, set the following properties in the bottom pane:
   * **General**:
     + **Name**: Process Data
   * **Azure Databricks**:
     + **Databricks linked service**: *Select the****AzureDatabricks****linked service you created previously*
   * **Settings**:
     + **Notebook path**: *Browse to the****Users/your\_user\_name****folder and select the****Process Data****notebook*
     + **Base parameters**: *Add a new parameter named folder with the value product\_data*
6. Use the **Validate** button above the pipeline designer surface to validate the pipeline. Then use the **Publish all** button to publish (save) it.

Run the pipeline

1. Above the pipeline designer surface, select **Add trigger**, and then select **Trigger now**.
2. In the **Pipeline run** pane, select **OK** to run the pipeline.
3. In the navigation pane on the left, select **Monitor** and observe the **Process Data with Databricks** pipeline on the **Pipeline runs** tab. It may take a while to run as it dynamically creates a Spark cluster and runs the notebook. You can use the **↻ Refresh** button on the **Pipeline runs** page to refresh the status.

When the run succeeds, select its name to view the run details. Then, on the **Process Data with Databricks** page, in the **Activity Runs** section, select the **Process Data** activity and use its **output** icon to view the output JSON from the activity, which should resemble this:

|  |
| --- |
| {  "runPageUrl": "https://adb-..../run/...",  "runOutput": "dbfs:/product\_data/products.csv",  "effectiveIntegrationRuntime": "AutoResolveIntegrationRuntime (East US)",  "executionDuration": 61,  "durationInQueue": {  "integrationRuntimeQueue": 0  },  "billingReference": {  "activityType": "ExternalActivity",  "billableDuration": [  {  "meterType": "AzureIR",  "duration": 0.03333333333333333,  "unit": "Hours"  }  ]  }  } |

1. Note the **runOutput** value, which is the *path* variable to which the notebook saved the data.

Clean up

If you’ve finished exploring Azure Databricks, you can delete the resources you’ve created to avoid unnecessary Azure costs and free up capacity in your subscription.