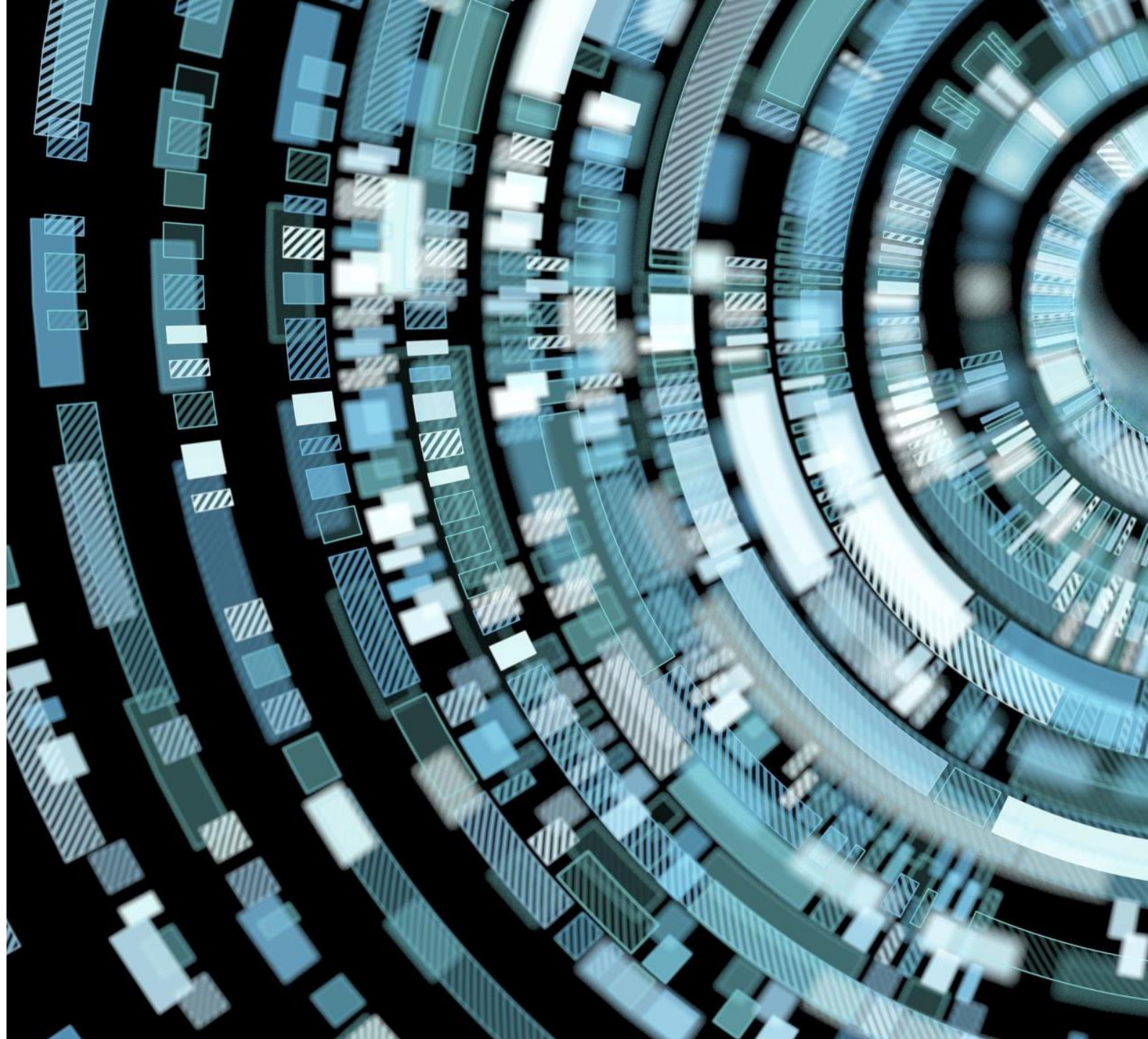


DATABRICKS SPARK HIVE DEMO

<https://advancedsqlpuzzles.com>

Scott Peters



OBJECTIVE

For this demo, we will perform the following...



Connect
Databricks to an
Azure Key Vault



Connect
Databricks to an
Azure Data
Lake



Create an ETL
process to
import csv files
from an Azure
Data Lake



Merge the data
into Hive tables



Insert the data
into a SQL
Server
database



Automate the
ETL via an Azure
Data Factory
pipeline

OBJECTIVE

The Git repository for this demo is located here:

<https://Github.com/smpetersGithub/AdvancedSQLPuzzles/tree/main/Suppliers%20and%20Parts%20Hive%20Demo>

Included in the repository are the following:

- A PDF version of this presentation (Databricks Hive Demo.pdf)
- Databricks Workspace (SuppliersAndParts.dbc)
- CSV Files to Import
- SQL Server Database DDL Scripts

If you are viewing on YouTube, each slide is set to 20 seconds.

You can pause using the spacebar, and the keys “J” and “L” will rewind and fast forward 10 seconds.

OBJECTIVE

For this demo we will be using the Suppliers and Parts database which consists of three csv files.

- 1) Suppliers.csv
- 2) Parts.csv
- 3) Shipments.csv

It's a popular database that has its own Wikipedia page.

The Suppliers and Parts Wikipedia article is located here:

https://en.wikipedia.org/wiki/Suppliers_and_Parts_database



AZURE **RESOURCES**

First, we will need to provision the following resources:

- Azure Key Vault
- Azure Data Lake
- Azure Data Factory
- SQL Server
- SQL Server Database
- Databricks Workspace







AZURE RESOURCES

Few reminders when provisioning your resources.

- When provisioning Databricks, select the premium version as this allows for creating secret scopes.
- If using an Azure free trial version, you will need to upgrade to a subscription plan to access Databricks.
- The Azure Storage Account needs to be provisioned as a data lake.
- A basic SQL Server database is will be sufficient for this tutorial. You will need to enable the database server to connect to other Azure resources.

AZURE RESOURCES

For this demo I created a resource group “rg-databricks-hive-demo” and provisioned the following resources:

<input type="checkbox"/>	 adf-databricks-hive-demo	Data factory (V2)	East US
<input type="checkbox"/>	 akv-databricks-hive-demo	Key vault	East US
<input type="checkbox"/>	 db-databricks-hive-demo	Azure Databricks Service	East US
<input type="checkbox"/>	 dlsdatabrickshivedemo	Storage account	East US
<input type="checkbox"/>	 sql-databricks-hive-demo	SQL server	East US
<input type="checkbox"/>	 sqladb-databricks-hive-demo (sql-databricks...	SQL database	East US




The documentation for Azure naming conventions is located here:

<https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-abbreviations>

AZURE RESOURCES




When you provision the resources, additional resource groups and resources are created that you do not have control of.

When you have finished provisioning the resources, you will see additional resource group for Databricks, and another for the Network Watcher.

<input type="checkbox"/>	 databricks-rg-db-databricks-hive-demo-spyhoadxx4z2g
<input type="checkbox"/>	 NetworkWatcherRG
<input type="checkbox"/>	 rg-databricks-hive-demo-001

AZURE RESOURCES

The “databricks-rg-db-databricks-hive-demo-spyheadxx4z2g” resource group has the following resources:

<input type="checkbox"/>	 dbstorage5vlna5c4osdds	Storage account
<input type="checkbox"/>	 workers-sg	Network security group
<input type="checkbox"/>	 workers-vnet	Virtual network

The “NetworkWatcherRG” resource group has the following resources:

<input type="checkbox"/>	 NetworkWatcher_eastus	Network Watcher
--------------------------	--	-----------------

AZURE RESOURCES

Here are the Azure resource names if you want to copy and paste.

- db-databricks-hive-demo (Databricks)
- dlslatabrickshivedemo (Azure Data Lake)
- akv-databricks-hive-demo (Azure Key Vault)
- sqlldb-databricks-hive-demo (SQL Database)
- sql-databricks-hive-demo (SQL Server)

DATABRICKS

Next, we will connect our resources together.

In Databricks, we will need to understand two concepts:

- 1) Secret Scopes
- 2) Databricks Tokens

First, we will begin with secret scopes.

Secret scopes allow you to securely connect to your Azure Key Vault service, where we will store our database and storage account credentials safely.

There are two different ways of creating a scope:

- 1) Azure Key Vault backed
- 2) Databricks backed

The Databricks secret scope documentation is located here:

<https://docs.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes>

DATABRICKS

Open the Databricks Workspace

The screenshot displays the Microsoft Azure Databricks Workspace interface. The top navigation bar includes 'Microsoft Azure | Databricks' on the left and 'Portal | scottpeters1188@outlook.com' on the right. A left-hand sidebar contains the Databricks logo and a menu with options: 'Data Science & E...', 'Create', 'Workspace' (highlighted), 'Repos', 'Recents', 'Search', 'Data', 'Compute', 'Jobs', 'Help', 'Settings', and a user profile section for 'adb-databricks-hive-...' with email 'scottpeters1188@outlook...'. Below the sidebar is a 'Menu options' icon. The main workspace area features a 'Workspace' header with a 'Home' button and a dropdown menu showing 'Workspace', 'Shared', 'Users', and 'SuppliersAndParts'. The central content area is titled 'Azure Databricks' and includes three primary action cards: 'Explore the Quickstart Tutorial' (with a document icon and a lightbulb), 'Import & Explore Data' (with a dashed box icon and a cloud upload icon), and 'Create a Blank Notebook' (with a document icon and a plus sign). Each card has a brief description of the action. Below these cards are three sections: 'Common Tasks' (listing 'New Notebook', 'Create Table', 'New Cluster', 'New Job', 'New MLflow Experiment', 'Import Library', and 'Read Documentation'), 'Recents' (listing 'Master_Execution', 'Connect_DataLake', and 'Initialize_Create_Database'), and 'Documentation' (listing 'Documentation', 'Release Notes', and 'Getting Started').

DATABRICKS

Next, in the Databricks workspace, access the secret scope window.

To access the secret scope window in Databricks, attach the string “#secrets\createScope” after your Databrick’s instance.

A Databrick’s instance will have the following URL:

`https://<databricks-instance>#secrets/createScope`.

This URL is case sensitive; scope in createScope must be uppercase.

The documentation for Databricks secret scopes is located here:

<https://docs.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes>

DATABRICKS SECRET SCOPE

The “Create Secret Scope” window will look like the following.

We will cover the “DNS Name” and “Resource ID” in next slides.

HomePage / Create Secret Scope

Create Secret Scope

[Cancel](#) [Create](#)

A store for secrets that is identified by a name and backed by a specific store type. [Learn more](#)

Scope Name [?](#)

databricks-hive-demo-secret-scope

Manage Principal [?](#)

Creator [v](#)

Azure Key Vault [?](#)

DNS Name

https://akv-databricks-hive-demo.vault.azure.net/

Resource ID

/subscriptions/98c00000-1000-1000-1000-100000000000/resourceGroups/rg-databrickl

DATABRICKS SECRET SCOPE

Azure Key Vault Properties Page



The DNS Name (Vault URI) and the Resource ID are in your Azure Key Vault properties page.



Copy these values from your Azure Key Vault properties page and insert the values into your “Create Secret Scope” window.



Name	akv-databricks-hive-demo
Sku (Pricing tier)	Standard
Location	eastus
Vault URI	https://akv-databricks-hive-demo.vault.azure.net/
Resource ID	/subscriptions/99306f6c-200f-48d8-abf4-7c5cbddfd78/resourceGro...
Subscription ID	99306f6c-200f-48d8-abf4-7c5cbddfd78
Subscription Name	Azure Subscription 1
Directory ID	e5a62d55-c2db-41de-b112-4aebc06a30a4
Directory Name	Default Directory


DATABRICKS SECRET SCOPE

Once completed, you will get the following confirmation.

The secret scope named databricks-hive-demo-secret-scope has been added.

Manage secrets in this scope in Azure KeyVault with manage principal = creator

And you will see the following access policy added to the key vault

	Name	Email	Key Permissions	Secret Permissions
APPLICATION				
	AzureDatabricks		0 selected ▼	2 selected ▼

The documentation for the Azure Key Vault access policies is located here:

<https://docs.microsoft.com/en-us/azure/key-vault/general/assign-access-policy?tabs=azure-portal>

DATABRICKS TOKEN

Next, we will create a Databricks token.

This token will allow our Azure Data Factory to access our Databricks notebook. Tokens replace passwords in an authentication flow and should be protected like passwords.

We will use this token when we setup a linked service in our Azure Data Factory that connects to Databricks.

The documentation for Databricks tokens is located here:

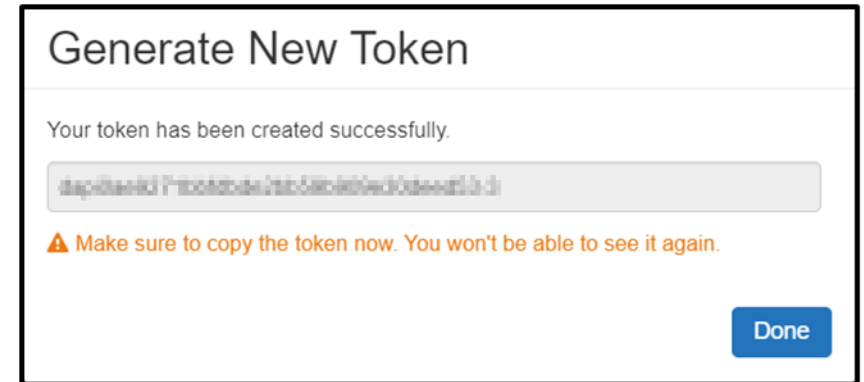
<https://docs.databricks.com/dev-tools/api/latest/authentication.html>

This token will be stored in the Azure Key Vault, which we will setup later.

DATABRICKS TOKEN

Create a token in Azure Databricks by navigating to the “User Settings” in your Databricks workspace and selecting “Generate New Token”.

Save the token, as directed by the yellow warning message.



I created the token “databricks-hive-demo”, but it is the token value that is important, not the name. This token will be stored in the Azure Key Vault, which we will setup later.

Once created, you will see the following in your Databricks “User Settings”.

Comment	Creation ↑	Expiration
databricks-hive-demo	2021-11-24 14:29:47 CST	Never

AZURE DATA LAKE

Next, we will setup or Azure Data Lake.

Create two containers:

- 1) source
- 2) hive

Within the “hive” container create a directory called “database”.

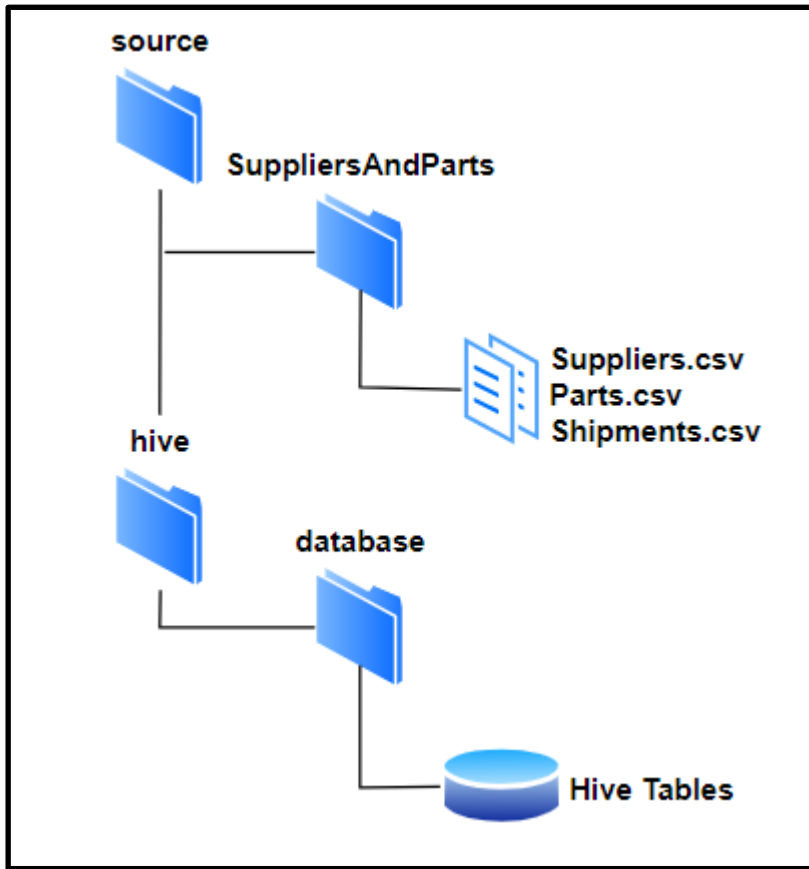
Within the “source” container, create a directory called “SuppliersAndParts”.

The directory “source > SuppliersAndParts” is where we will store the csv files to import, and “hive > Database” will contain the hive tables (which we will create later).

Upload the text files from the Git repository into the “source > SuppliersAndParts” container.

AZURE DATA LAKE

Your file structure in the storage account will look like the following:



Copy the files “Suppliers.csv”, “Parts.csv”, and “Shipments.csv” files into the “SuppliersAndParts” folder.

The “database” folder will contain the Hive tables, which we will provision later in this demo.

The Git repository for this demo is located here:

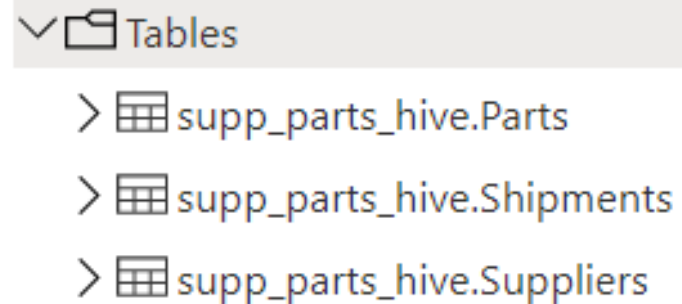
<https://Github.com/smpetersGithub/AdvancedSQLPuzzles/tree/main/Suppliers%20and%20Parts%20Hive%20Demo>

SQL SERVER DATABASE

Next, we will create the schema and tables in the SQL Server database.

The script to create the schema and tables are provided in the Git directory.

We will create a schema named “supp_parts_hive” and the following tables:



The Git repository for this demo is located here:

<https://Github.com/smpetersGithub/AdvancedSQLPuzzles/tree/main/Suppliers%20and%20Parts%20Hive%20Demo>

AZURE KEY VAULT

Next, we will create the secrets inside of the Azure Key Vault.

Here are the secrets you will need to setup in the Azure Key Vault.

Databricks will be able to access these values via the secret scope we created earlier.

Name	Type	Status
DataLakeAccessKey		✓ Enabled
DatabricksToken		✓ Enabled
DataLakeName		✓ Enabled
sqldbName		✓ Enabled
sqljdbcPort		✓ Enabled
sqlPassword		✓ Enabled
sqlServerName		✓ Enabled
sqlUserName		✓ Enabled

These secrets will be used within the Databricks notebooks, except for “DatabricksToken”, which will be used by our Azure Data Factory linked service.

AZURE KEY VAULT

Secret Name	Value	Description
sqljdbcPort	1433	This should always be 1433
sqldbName	sqldb-databricks-hive-demo	The name of the database
sqlPassword	MyPassword	The password of the database
sqlUserName	MyUserName	The admin user of the database
sqlServerName	sql-databricks-hive-demo.database.windows.net	The database server connection string
DataLakeAccessKey	Review the next few slides....	Copied from the Azure Storage Account properties
DataLakeName	dlsdatabrickshivedemo	The name of the storage account
DatabricksToken	Created in previous slides...	Used by Data Factory linked service

The two services we need Databricks to authenticate to are:

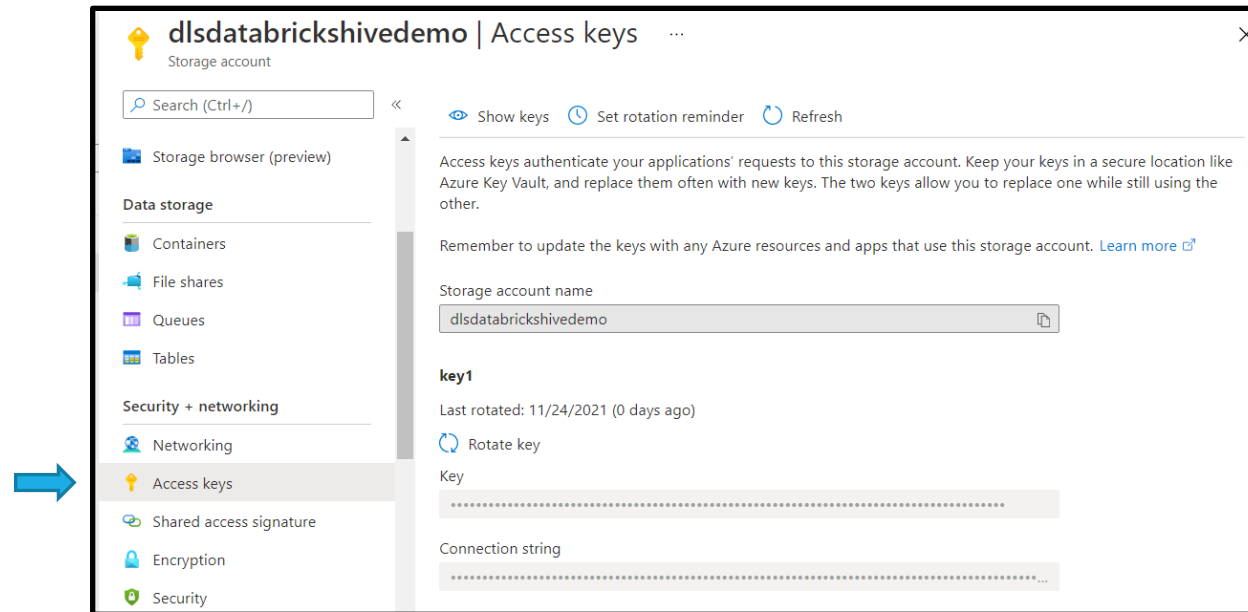
- 1) SQL Server Database
- 2) Azure Storage Account

The secret “DatabricksToken” will be used by our Data Factory linked service.

We will create the Azure Storage Account access key in the next slides.

AZURE KEY VAULT

The secret “DataLakeAccessKey” is created by navigating to your storage account and copying the access key.



The documentation for Azure Storage Account access keys is located here:

<https://docs.microsoft.com/en-us/azure/storage/common/storage-account-keys-manage?tabs=azure-portal>

RECAP

Let's do a quick recap of everything we have accomplished so far:

- We have provisioned our Azure Resources
- We created a Databricks secret scope
- We created a Databricks token
- We setup our Azure Storage Account as a data lake with the needed directories and imported the text files
- We created the schema and tables in our SQL Server Database
- We created the secret keys in our Azure Key Vault

RECAP

Next, we will perform the following:

- Import our Databricks workspace
- Create a Databricks cluster
- Run the database setup scripts
- Review the code in the workspace
- Test our Databricks workspace
- Create an Azure Data Factory pipeline
- Execute the Azure Data Factory pipeline

DATABRICKS

The following documentation gives a good overview of what we will be accomplishing for the remainder of these slides. Take a few moments to review the following documentation.

A Microsoft tutorial for incorporating Databricks into Data Factory is located here:
<https://docs.microsoft.com/en-us/azure/data-factory/transform-data-using-databricks-notebook>

If you are unfamiliar with creating clusters, running notebooks, or navigating the workspace, I recommend working through a basic tutorial of Databricks before proceeding.

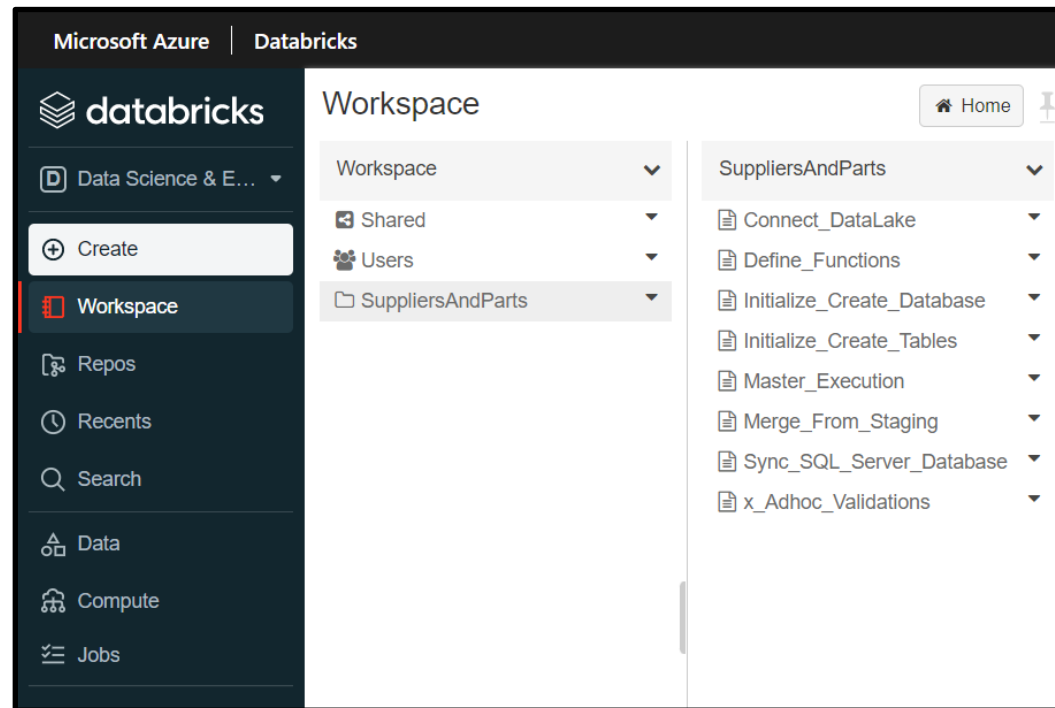
DATABRICKS

Begin by importing the workspace from the Git repository.

First, we will begin by importing the “SuppliersAndParts.dbc” workspace from the Git repository.

This workspace contains the notebooks needed to create our ETL.

Once you import the dbc file, you will see the following “SuppliersAndParts” folder in your workspace.



DATABRICKS

The workbook consists of the following notebooks:

Included Notebooks

- Connect_DataLake
- Define_Functions
- Initialize_Create_Database
- Initialize_Create_Tables
- Master_Execution
- Merge_From_Staging
- Sync_SQL_Server_Database
- x_Adhoc_Validations

DATABRICKS

Here is a brief description of each notebook.

We will cover each of these notebooks in more detail.

Notebook	Description
Connect_DataLake	Reads variables from the Azure Key Vault and connects to the Azure Data Lake
Define_Functions	Creates the functions used in the ETL process
Initialize_Create_Database	Script to create the Hive database
Initialize_Create_Tables	Script to create the Hive tables
Master_Execution	The master notebook called from Data Factory
Sync_SQL_Server_Database	Syncs the Hive tables to the SQL Server Database
x_Adhoc_Validations	General script used for auditing

DATABRICKS

Next, create a Databricks cluster. We will need to create a cluster to run our notebooks.

The documentation for configuring Databricks clusters is located here:

<https://docs.microsoft.com/en-us/azure/databricks/clusters/configure>

Remember to put a termination activity time on the cluster!

Here are the details of the cluster I have created for this demo.

A simple standard cluster will suffice. Feel free to experiment with different cluster sizes.

The screenshot shows the Databricks Clusters configuration interface. At the top, there's a breadcrumb 'Clusters /' and a cluster name 'Standard' with a star icon. Action buttons include 'Edit', 'Permissions', 'Start', 'Clone', 'Restart', 'Terminate', and 'Delete'. Below these are tabs for 'Configuration', 'Notebooks', 'Libraries', 'Event Log', 'Spark UI', 'Driver Logs', 'Metrics', 'Apps', and 'Spark Cluster UI - Master'. The 'Configuration' tab is active, showing settings for 'Policy' (Unrestricted), 'Cluster Mode' (Standard), 'Databricks Runtime Version' (8.3), and 'Autopilot Options' (Enable autoscaling and Terminate after 10 minutes). The 'Worker Type' section shows 'Standard_DS3_v2' with 14 GB Memory and 4 Cores, and a table for 'Min Workers' (2), 'Max Workers' (8), and 'Current' (0). The 'Driver Type' is also 'Standard_DS3_v2'. At the bottom, the cost is listed as 'DBU / hour: 2.25 - 6.75' and the cluster type 'Standard_DS3_v2' is highlighted.

Worker Type	14 GB Memory, 4 Cores	Min Workers	Max Workers	Current	Spot instances
Standard_DS3_v2		2	8	0	<input type="checkbox"/>

DATABRICKS

Next, create the
“demo” database.

Run the following notebook first:

Initialize_Create_Database

The following SQL statement creates a Hive database named “demo”.

Create Hive database

```
1 DROP DATABASE if exists demo CASCADE;  
2 CREATE DATABASE IF NOT EXISTS demo COMMENT 'This is demo database' LOCATION '/demo/database';
```


DATABRICKS

Second, create the tables for the Suppliers, Parts and Shipments data.

Run the following notebook next:

Initialize_Create_Tables

This will create the three tables for 1) Suppliers, 2) Parts and 3) Shipments.

Create table statements

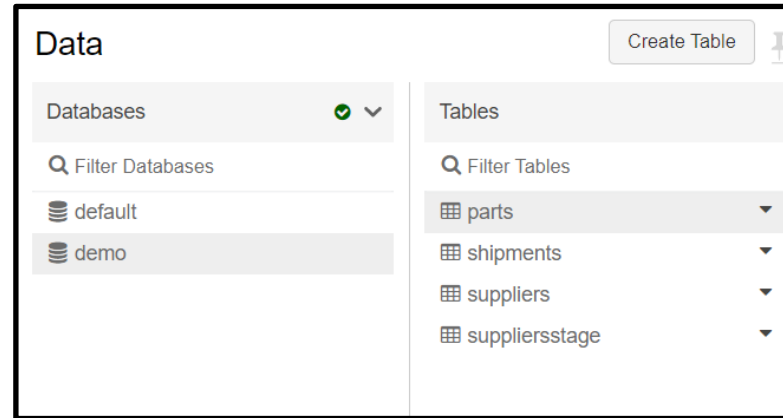
```
1 CREATE TABLE IF NOT EXISTS demo.Suppliers
2 (
3     SupplierId numeric(4,0),
4     SupplierName string,
5     Status numeric(4,0),
6     City string,
7     InsertDate timestamp
8 )
9 USING DELTA
10 LOCATION "abfss://hive@dlsdatabrickshivedemo.dfs.core.windows.net/database/Suppliers/delta/";
```

You will need to change the location only if you named the Azure storage account directory differently than this demo.

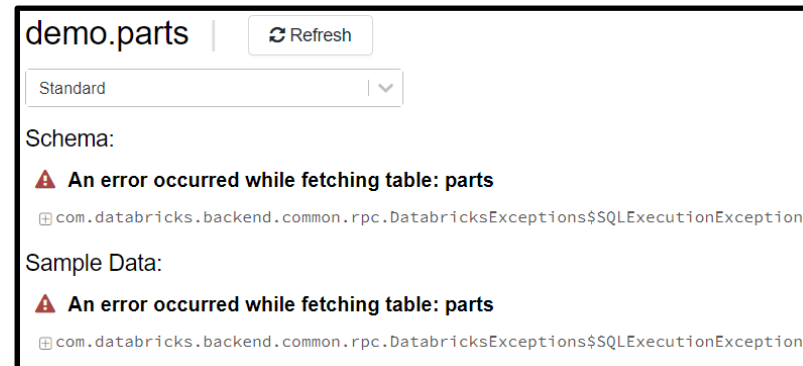
DATABRICKS

Second, create the tables for the Suppliers, Parts and Shipments data.

After creating the tables, you will see the tables in the “Data” pane of the workspace.



You will get the following error when you select a table. This is perfectly normal.



DATABRICKS

Next, we will test our connection to the Azure Key Vault and the Azure Storage Account.

Run the following notebook to test the connectivity to the Azure Key Vault and the Azure Data Lake services.

Connect_DataLake

The following code connects to the Azure Key Vault.

Set variables

```
1 scopename="databricks-hive-demo-secret-scope"
2 datalakename=dbutils.secrets.get(scope=scopename,key="DataLakeName")
3 TablePathHive = "abfss://hive@"+datalakename+".dfs.core.windows.net/database/";
4 print(TablePathHive)
5
6 #-----Database Connection-----
7 DBServer = dbutils.secrets.get(scope=scopename,key="sqlServerName")
8 dbServerUserName = dbutils.secrets.get(scope=scopename,key="sqlUserName")
9 dbServerPassword=dbutils.secrets.get(scope=scopename,key="sqlPassword")
10 DBName = dbutils.secrets.get(scope=scopename,key="sqldbName")
11 jdbcPort = dbutils.secrets.get(scope=scopename,key="sqljdbcPort")
12 url= "jdbc:sqlserver://{0}:{1};database={2}".format(DBServer,jdbcPort,DBName)
13 properties = {"user" : dbServerUserName,"password" : dbServerPassword }
14
```

If the Azure Key Vault secret names are different than this demo, you will need to modify these values here. Note the only string embedded in the notebook is the secret scope that we created earlier.

DATABRICKS

Next, let's test if we can run the master notebook.

Master_Execution

The “Master_Execution” notebook will run each time for the 1) Suppliers, 2) Parts, and 3) Shipments csv files.

This notebook inserts into the staging tables by calling the function “ProcessData” located in the “Define_Functions” notebook.

This notebook also calls the “Merge_From_Staging” and the “Synch_SQL_Server_Database”.

You may wish to remove the merge and synch notebooks and run these as separate Data Factory pipelines.

Note, all three of these notebooks use the “dbutils.widgets” to determine the file name and the process name from the data factory.

DATABRICKS

Next, let's test if we can run the master notebook.

Master_Execution

The “Master_Execution” notebook will run each time for the 1) Suppliers, 2) Parts, and 3) Shipments csv files.

The “dbutils.widect” code reads the variable “ProcessName” from the Azure Data Factory.

To test the code in Databricks, manually set the value of “ProcessName”.

Reads the variable(s) from the Azure Data Factory pipeline

```
1  #Determines the ProcessName and FileName from the Azure Data Factory
2  dbutils.widgets.text("ProcessName", "", "")
3  ProcessName = dbutils.widgets.get("ProcessName")
4  dbutils.widgets.text("FileName", "", "")
5  FileName = dbutils.widgets.get("FileName")
6
7  #Manually sets the ProcessName and FileName if testing in Databricks
8  if ProcessName == "":
9      ProcessName = 'Suppliers'
10     FileName = '/SuppliersAndParts/Suppliers.csv'
```

DATABRICKS

Define_Functions

In the below code, the variable “TableName” is the concatenation of the variable “ProcessName” and the string “Stage”.

Process file data to Hive

```
1 def ProcessData(FilePath, ProcessName):
2     print("wasbs://source@" + dataLakename + ".blob.core.windows.net" + FilePath)
3     df = spark.read.csv("wasbs://source@" + dataLakename + ".blob.core.windows.net" + FilePath, sep=",", mode="DROPMALFORMED", schema=GetSchema(ProcessName))
4
5     if (len(df.head(1)) == 0):
6         raise ValueError("Please review the schema. Possible new columns were added.")
7     else:
8         new_df = GenerateIncomingDataHeader(df.limit(1))
9
10    #Verify the header
11    VerifyHeader(df, new_df)
12
13    #Fetch data in file
14    df = df.filter(~col(df.schema.fields[0].name).contains(df.schema.fields[0].name))
15
16    #Concatenates ProcessName and the string "Stage" to derive the staging table name.
17    TableName = ProcessName + "Stage"
18    InsertDataToHive(df, TableName)
19
20    print('End of Function: ProcessData')
```

DATABRICKS

Define_Functions

The insert into the staging table has the option of “overwrite” and “truncate”.

On each execution the staging table is truncated and then inserted into.

Insert data to Hive

```
1  def InsertDataToHive(df, TableName):
2      print('Table being inserted is',TableName)
3      print('The table path is',TablePathHive)
4
5      #Truncate and overwrite options are set to True
6      df.write.mode("overwrite").option("truncate", True).option("overwriteSchema", "true").format("delta").option("path",
7      TablePathHive+"/"+TableName+"/delta/").saveAsTable("demo."+TableName)
8
9      print('End of Function: InsertDataToHive')
```

DATABRICKS

Now we will review some of the specifics in the code.

Merge_From_Staging

This notebook merges the data from the staging to the production tables.

This notebook also uses the “dbutils.widgets”, which we have discussed in previous slides.

To run this notebook in Databricks, manually set the variable “ProcessName” to “Suppliers” as shown below.

Reads the variable(s) from the Azure Data Factory pipeline

```
1 dbutils.widgets.text("ProcessName", "", "")
2 ProcessName = dbutils.widgets.get("ProcessName")
3
4 #Manually sets the ProcessName if testing in Databricks
5 if ProcessName == "":
6     ProcessName = 'Suppliers'
```


DATABRICKS

Sync_SQL_Server_Database

The synchronization to the SQL Server database has the same overwrite and truncate options as the inserts into the Hive staging tables.

If you receive connection errors to the database, check that the SQL Server has the option set to connect to other Azure services.

Truncate and insert into SQL Server database

```
1  if ProcessName=="Suppliers":
2      df = spark.sql("""
3          Select
4              SupplierId,SupplierName,Status,City
5          from
6              demo.Suppliers
7          """)
8      )
9      df.write.mode("overwrite").option("truncate", True).jdbc(url=url,table="supp_parts_hive.Suppliers",properties=properties);
```

DATABRICKS

Now we will review
some of the specifics in
the code.

x Adhoc Validations

This notebook is for creating and saving any ad-hoc statements that you wish to run.

After you have ran an ETL, I recommend using this notebook to explore your data.

Remember you may have only tested on your Suppliers data. The Parts and Shipments tables may be blank if you have not tested these processes.

DATA FACTORY

Next, we will setup the Azure Data Factory.

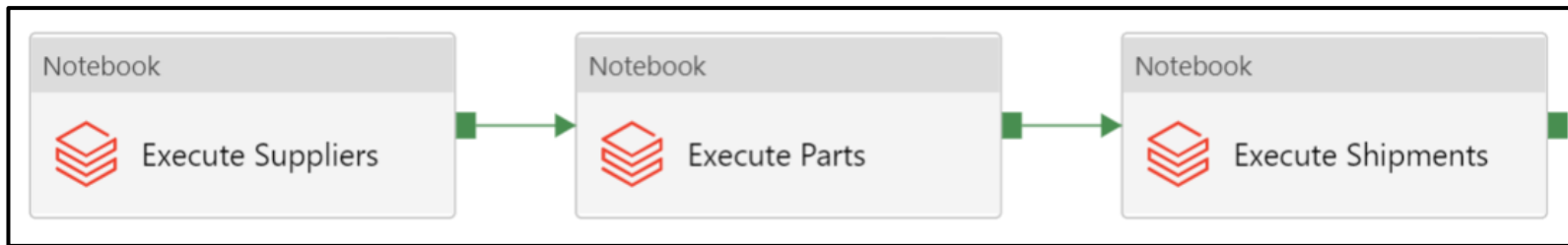
We will need to create two Azure Linked Services.

One to link to our 1) Azure Key Vault, and the second to link to the 2) Databricks Workspace.

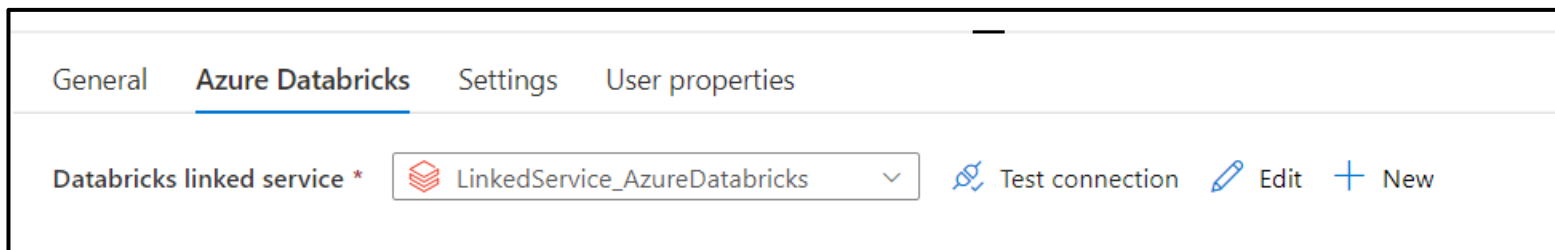
The Databricks Linked Service will utilize the Azure Key Vault Linked Service and access the Database token we setup earlier in the demo.

DATA FACTORY

Before we create the linked services, create a pipeline and add three Databricks notebooks activities. Chain them together to avoid getting “unexpected failure while waiting for the cluster” errors.




Each of these activities will utilize the Databricks linked service we will create in the next slides.



DATA FACTORY

First, the Data Factory will need access to the Azure Key Vault. Navigate to your Azure Key Vault and add an access policy that allows the Data Factory secret permissions.

The Azure Key Vault you provisioned will now have an access policy for your Databricks and the DataFactory.

Name	Email	Key Permissions	Secret Permissions
APPLICATION			
 adf-databricks-hive-d...		0 selected ▼	2 selected ▼

DATA FACTORY

AKV Linked Service

To link to the Azure Key Vault, create the AKV linked service using your Azure subscription.

You can also enter the AKV manually by providing the URL of the linked service, which is located in the properties page of the AKV.

New linked service (Azure Key Vault)

Name *

Description

Authentication method

Managed Identity

Azure key vault selection method ⓘ

☒ From Azure subscription ☐ Enter manually

Azure subscription ⓘ

Azure Subscription 1 (98306f8c-2001-48d8-ab44-7c5cbdd1de78)

Azure key vault name *

akv-databricks-hive-demo

Edit key vault

Managed identity name: **adf-databricks-hive-demo**
Managed identity object ID: **b846bd98-b226-4cd0-b9c6-9deaa3770093**
[Grant Data Factory service managed identity access to your Azure Key Vault.](#)

DATA FACTORY

Databricks Linked Service

To link to the Databricks workspace, fill in the following dropdowns with your subscription and workspace information.

You can also enter the information in manually by selecting “Enter Manually” from the dropdown box.

New linked service (Azure Databricks)

Name *

Description

Connect via integration runtime * ⓘ

Account selection method *

Azure subscription * ⓘ

Databricks workspace * ⓘ

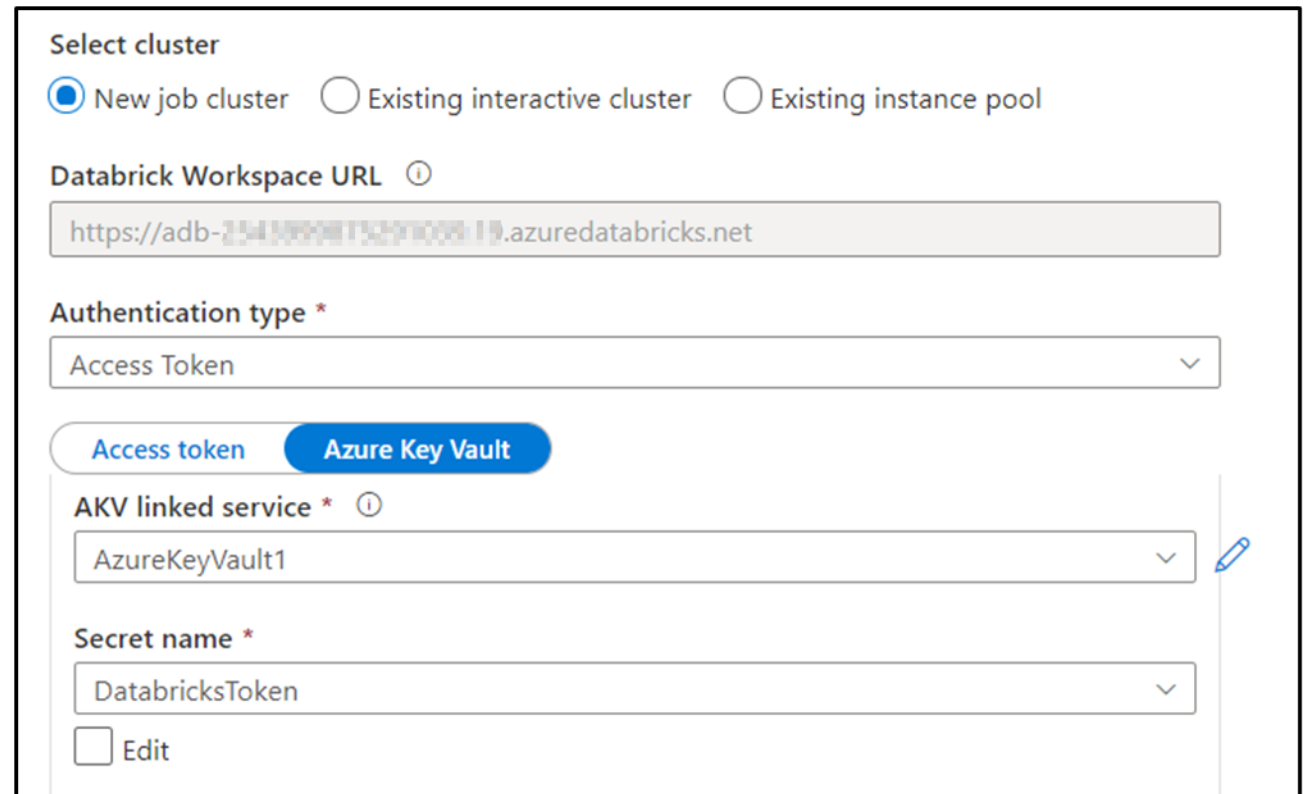
DATA FACTORY

Databricks Linked Service

Next, select “Access Token” as the authentication type.

Link to the AKV by using the linked service we created in a prior slide.

Then select the secret name you used to store the Databricks token.



The screenshot shows the 'Databricks Linked Service' configuration window. At the top, under 'Select cluster', the 'New job cluster' radio button is selected. Below this, the 'Databricks Workspace URL' is entered as 'https://adb-2543000001529100081@.azuredatabricks.net'. The 'Authentication type' dropdown is set to 'Access Token'. Below the dropdown, there are two tabs: 'Access token' and 'Azure Key Vault', with 'Azure Key Vault' being the active tab. Under the 'Azure Key Vault' tab, the 'AKV linked service' dropdown is set to 'AzureKeyVault1'. The 'Secret name' dropdown is set to 'DatabricksToken'. At the bottom left, there is an 'Edit' checkbox which is currently unchecked.

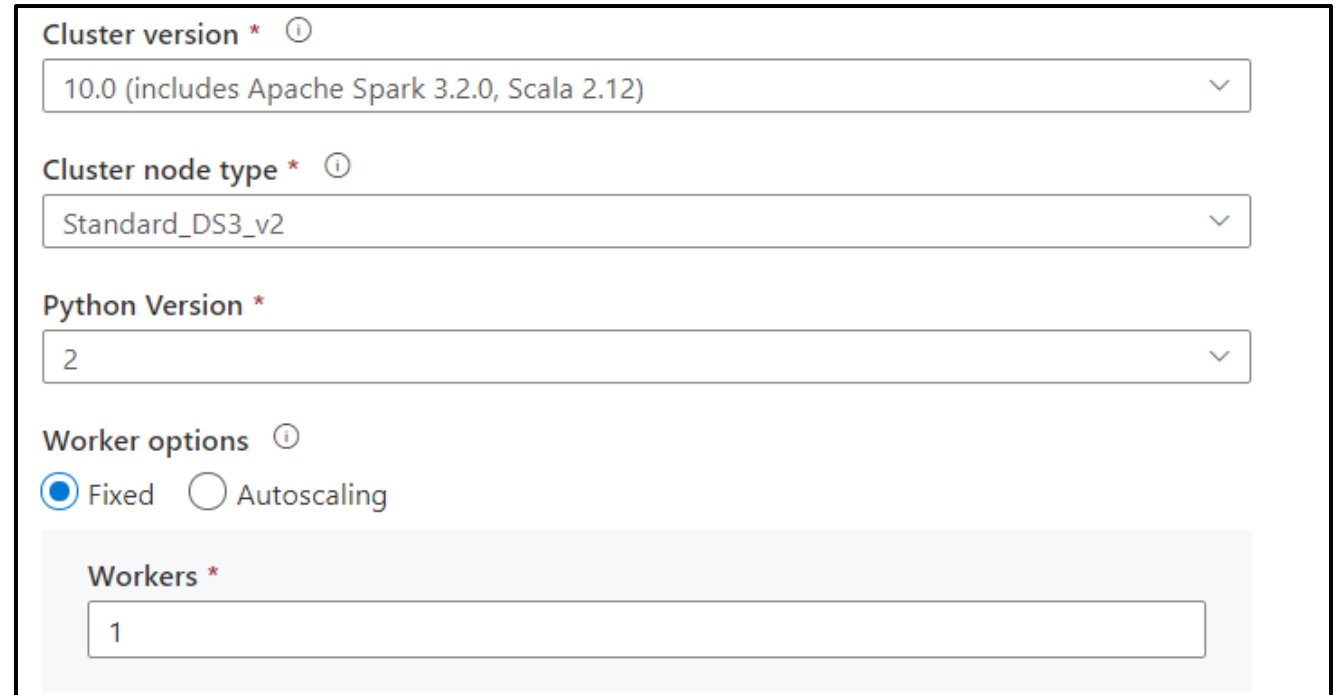
DATA FACTORY

Databricks Linked Service

Finish the linked service by filling in the rest of the fields.

Here you can see the options that I chose for cluster version, node type, etc.....

Selecting a new job cluster is the best option for cost savings, but it does add a few minutes of processing time to the pipeline to allow for the cluster to start up.



The image shows a screenshot of the Databricks cluster configuration interface. It includes the following fields and options:

- Cluster version *** (with an information icon): A dropdown menu showing "10.0 (includes Apache Spark 3.2.0, Scala 2.12)".
- Cluster node type *** (with an information icon): A dropdown menu showing "Standard_DS3_v2".
- Python Version ***: A dropdown menu showing "2".
- Worker options** (with an information icon): Two radio buttons, "Fixed" (selected) and "Autoscaling".
- Workers ***: A text input field containing the number "1".

DATA FACTORY

For the settings tab, select the notebook path and create two parameters 1) “FileName” and 2) “ProcessName”.

Fill in the appropriate information.

You will need to do this for the 1) Suppliers, 2) Parts and 3) Shipments activities.

Here is the screen shot for the Suppliers activity.
Note the use of forward slashes in the parameter “FileName”.

The screenshot shows the 'Settings' tab in the Azure Data Factory interface. At the top, there are four tabs: 'General', 'Azure Databricks', 'Settings' (which is selected), and 'User properties'. Below the tabs, the 'Notebook path *' is set to '/SuppliersAndParts/Master_Execution', with 'Browse' and 'Open' buttons to its right. Under the 'Base parameters' section, there are '+ New' and 'Delete' icons. A table lists two parameters:

<input type="checkbox"/>	Name	Value
<input type="checkbox"/>	FileName	/SuppliersAndParts/Suppliers.csv Add dynamic content [Alt+Shift+D]
<input type="checkbox"/>	ProcessName	Suppliers

DATAFACTORY

Execute the pipeline by selecting “Debug”.

Ensure there is no running cluster in your databricks before executing, else you will get “unexpected failure while waiting for the cluster” an error.

The total execution time for my pipeline is around 11 minutes.

Pipeline run ID: 855cf3a2-746f-4470-972a-e1594792a843 [🔗] 🔄 ⓘ

Name	Type	Run start	Duration	Status	Integration runtime
Shipments	Notebook	2021-11-24T23:24:32.057919'	00:02:50	✓ Succeeded	DefaultIntegrationRuntime (East US)
Parts	Notebook	2021-11-24T23:21:56.970290'	00:02:34	✓ Succeeded	DefaultIntegrationRuntime (East US)
Suppliers	Notebook	2021-11-24T23:17:03.082412	00:04:52	✓ Succeeded	DefaultIntegrationRuntime (East US)

TESTING

To test your ETL, review the data in your Hive tables with the statements in the “x_Adhoc_Validations” notebook. Also, review the data in the SQL Server tables as well.

Then alter the CSV files by adding new records and modify the current records. Run the ETL and verify the output.

To best learn the specifics of the Databricks notebooks, I recommend changing names of your AKV, storage account, secret names, etc... research where the notebooks break and fix accordingly.

CONGRATULATIONS

You now have a template for using Databricks as a transformation service!!!

Check out my Git repository and SQL blog for all sort of puzzles, tips and tricks.

The Git repository for this demo is located here:

<https://Github.com/smpetersGithub/AdvancedSQLPuzzles/tree/main/Suppliers%20and%20Parts%20Hive%20Demo>