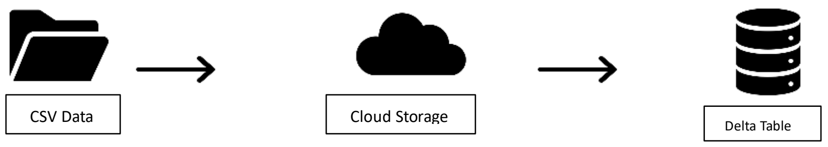
Wells Fargo – Data platform

Business Case:

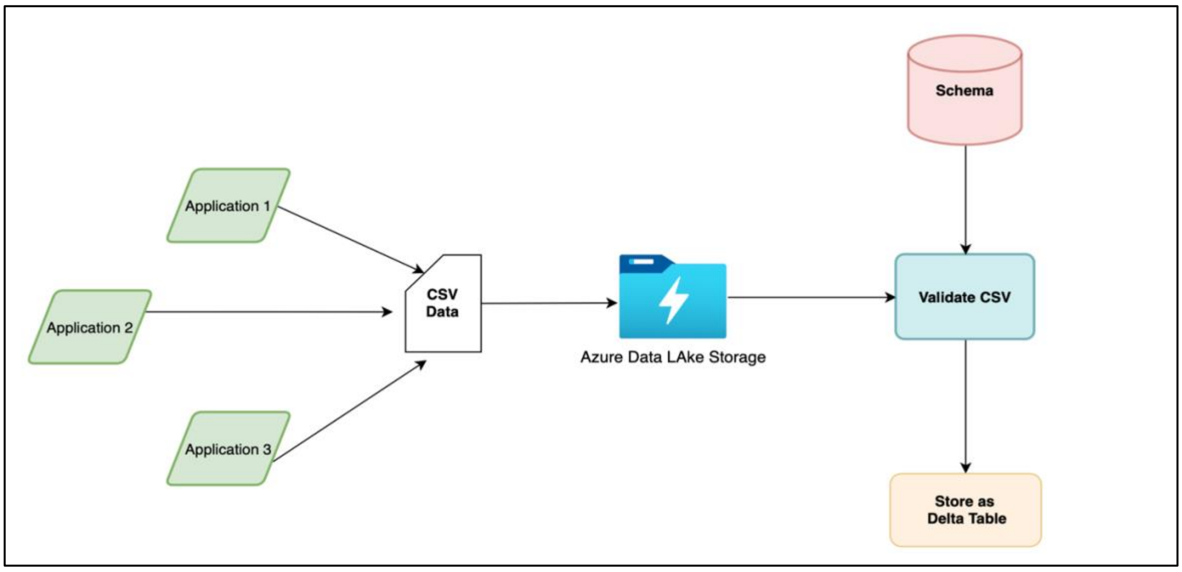
Wells Fargo has multiple internal applications that send huge amounts of data (***petabytes***) in CSV format **daily** in the company Azure cloud. We are required to perform **data/schema validation** on this incoming data and this needs to be stored in a **delta table** for downstream systems to make use for analysis.



Task:

1. Validation needs to be applied as follows:
   1.1. Need to check for **duplicate rows**. If file contains duplicate rows, it's rejected.
   1.2. Need to **validate the data format** for all the data fields, the date column names, and the date format is stored in the **Azure SQL server**, if validation fails the file is rejected.
2. Need to move all rejected files to Rejected folder.
3. Need to move all valid files to Staging folder.
4. Write all the validated files as a delta table in **Azure Databricks**

High-level Scope:

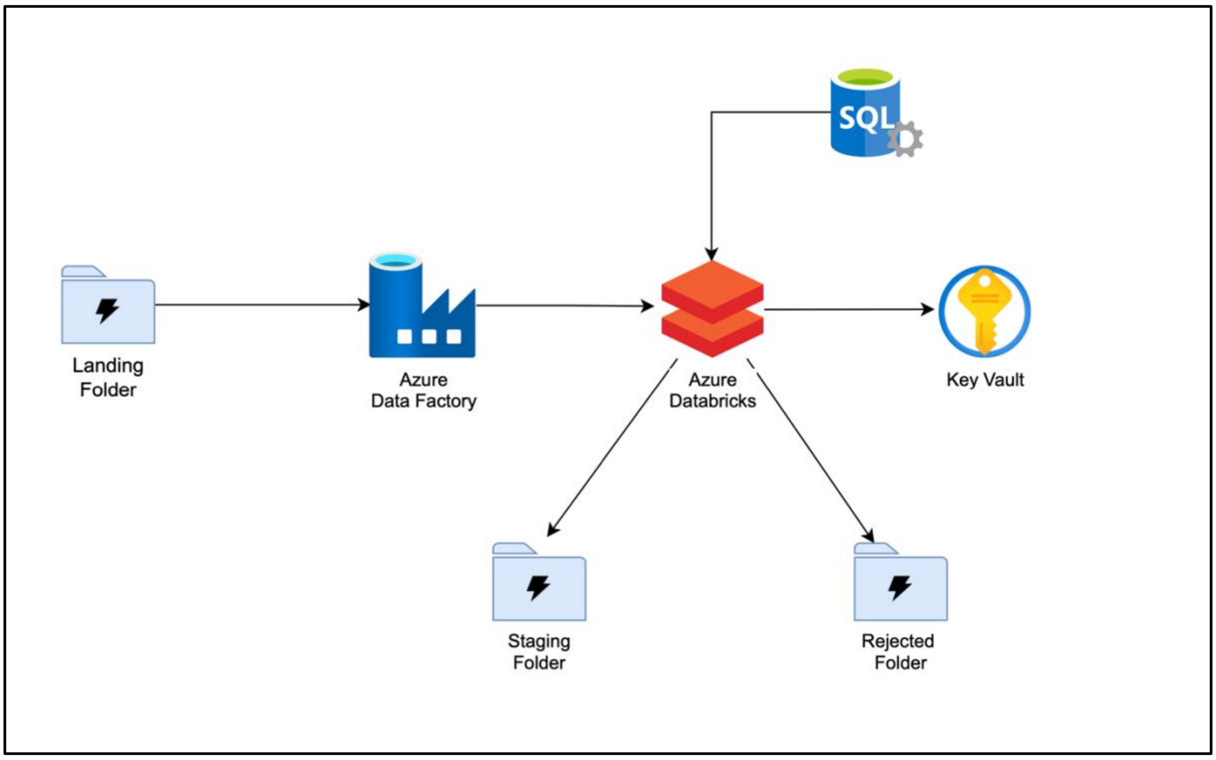


(*A diagrammatic representation of the high-level workflow of the architecture*)

Architecture Diagram:

1. Input data will come into the **Azure Data Lake Storage** account in a landing folder.
2. Then the data will be validated, by task scheduling in **azure data factory**.
3. ADF will pull the input data and invoke **Databricks** (spark) and create a delta table.

4. To apply validation rules, the **schema** will be fetched by the **SQL server**.
5. The file in landing folder will be validated against obtained schema and <u>categorized</u>.
6. credentials required to connect these services to each other will be stored in **azure key vault.**



(*A diagrammatic representation of the architecture*)

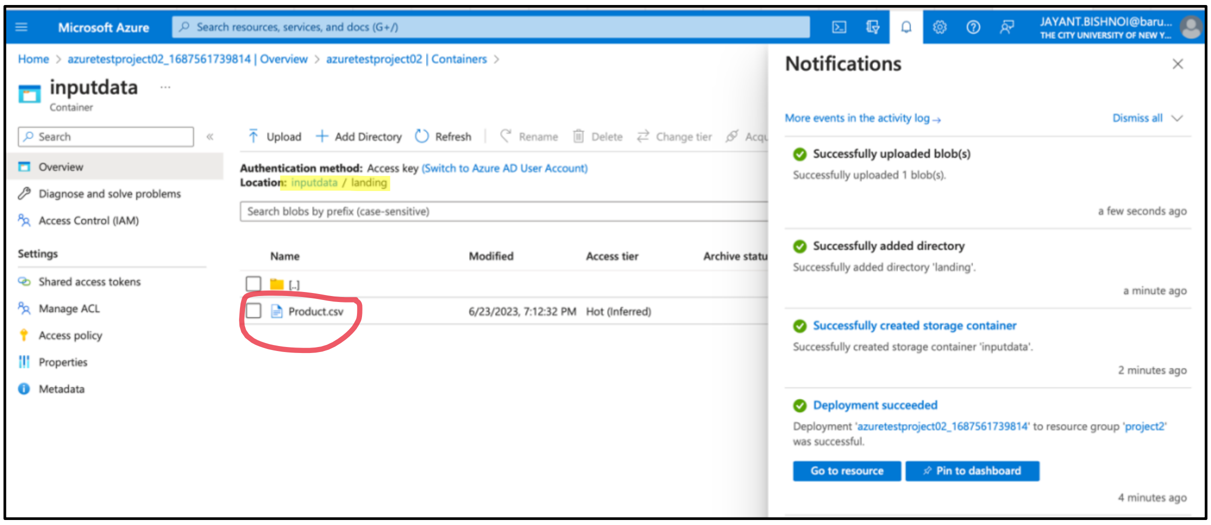
Phase 1 - Setting up Framework

Create Azure Data Lake Storage account:

(*In a real-world scenario, the files in landing folder will be provided by third party apps, but since this is a synthesized business use case, we are assuming that data is already present in the landing folder)*

Storage account & Resource Group:

1. Search for storage account in azure portal and click on create new.
2. Create new resource group called 'project 2'
3. Enable '**hierarchical namespace'**, to make it into an **Azure data lake storage gen2** account.
4. Review+Create to create the storage account.
5. Inside the storage account click in '**container**' to create a storage container and call it 'input'
6. Inside 'input' container we will create our **landing folder** where we want out s3 data to show.


Microsoft Azure | Search resources, services, and docs (G+/) | JAYANT.BISHNOI@baru... THE CITY UNIVERSITY OF NEW Y...

Home > azuretestproject02_1687561739814 | Overview > azuretestproject02 | Containers >

inputdata ...
Container

Search | Overview | Diagnose and solve problems | Access Control (IAM)

Settings: Shared access tokens | Manage ACL | Access policy | Properties | Metadata

Upload | Add Directory | Refresh | Rename | Delete | Change tier | Acqu

Authentication method: Access key (Switch to Azure AD User Account)
Location: inputdata / landing

Search blobs by prefix (case-sensitive)

Name	Modified	Access tier	Archive statu
[..]			
Product.csv	6/23/2023, 7:12:32 PM	Hot (Inferred)	

Notifications

More events in the activity log → | Dismiss all

Successfully uploaded blob(s)
Successfully uploaded 1 blob(s).
a few seconds ago

Successfully added directory
Successfully added directory 'landing'.
a minute ago

Successfully created storage container
Successfully created storage container 'inputdata'.
2 minutes ago

Deployment succeeded
Deployment 'azuretestproject02_1687561739814' to resource group 'project2' was successful.
Go to resource | Pin to dashboard
4 minutes ago

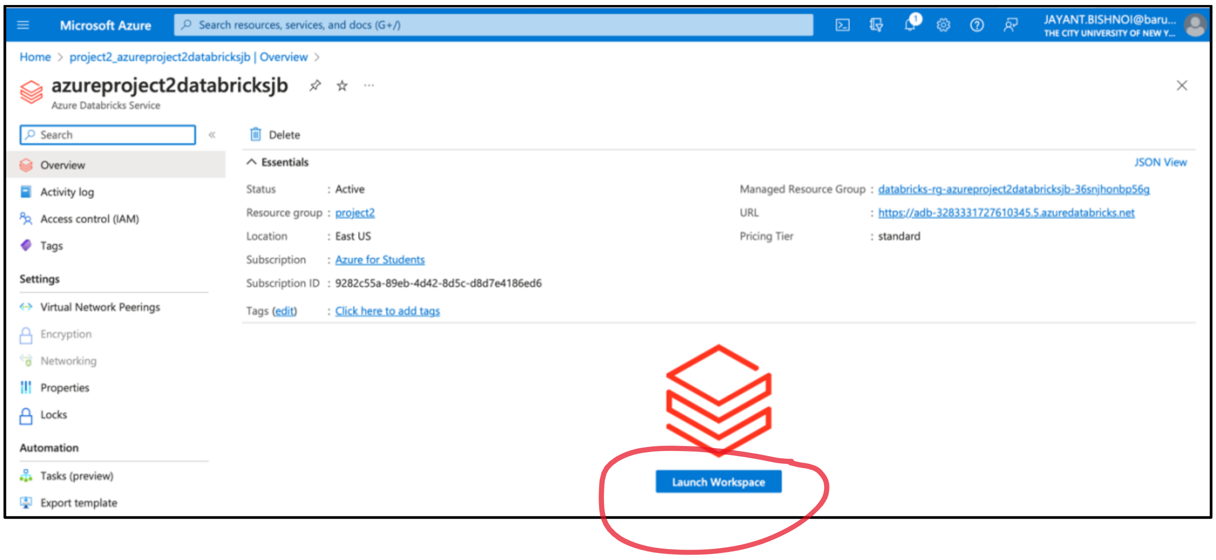
(Storage account with the input/landing folders and relevant data)

According to the architecture diagram, we need to set up an azure data factory account, <u>but without a destination</u> to send the data to it is not very practical, so we will set up the databricks account as destination first...

<u>Create Azure Databricks Account Workspace</u>:

<u>Azure Databricks:</u>

1. Search for 'databricks' in azure portal and click on create button.
2. Give a suitable name and assign a subscription and resource group (project2)
3. Keep the pricing tier as 'standard' for the purpose of the project.
4. Review and create


Microsoft Azure | Search resources, services, and docs (G+/) | JAYANT.BISHNOI@baru... THE CITY UNIVERSITY OF NEW Y...

Home > project2_azureproject2databricksjb | Overview >

azureproject2databricksjb
Azure Databricks Service

Search | Overview | Activity log | Access control (IAM) | Tags

Settings: Virtual Network Peerings | Encryption | Networking | Properties | Locks

Automation: Tasks (preview) | Export template

Delete

Essentials | JSON View

Status	: Active	Managed Resource Group	: databricks-rg-azureproject2databricksjb-36snjhonbp56g
Resource group	: project2	URL	: https://adb-3283331727610345.5.azuredatabricks.net
Location	: East US	Pricing Tier	: standard
Subscription	: Azure for Students		
Subscription ID	: 9282c55a-89eb-4d42-8d5c-d8d7e4186ed6		
Tags (edit)	: Click here to add tags		

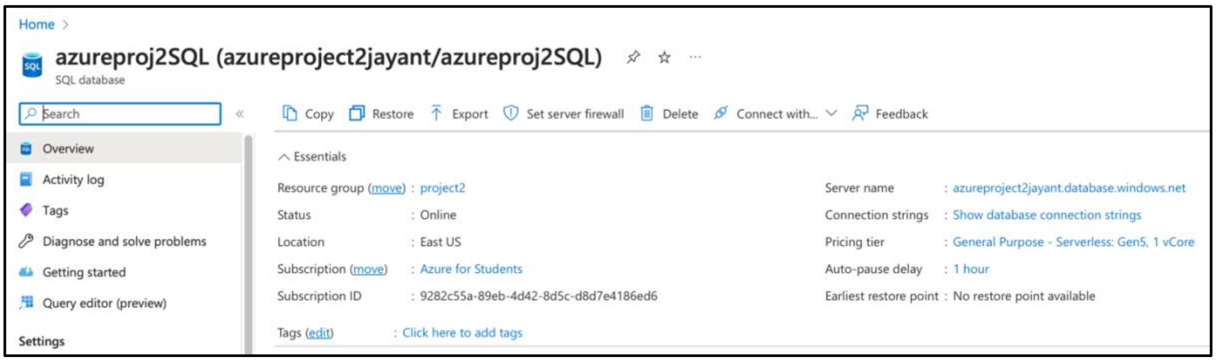
Launch Workspace

Before using ADF to call the databricks account we need to create a 'notebook' inside it, so that we can write the code for the validation and get that notebook called by ADF.

<u>Create Azure SQL Server</u>:

1. Create > subscription > resource group > database name.
2. Create new server and give name > SQL authentication > define username and password > compute + storage (basic tier)> LRS> public endpoint> allow access to DB> review +create.

(Go to resource and go to query editor, give login and password)


Home >

azureproj2SQL (azureproject2jayant/azureproj2SQL)

SQL database

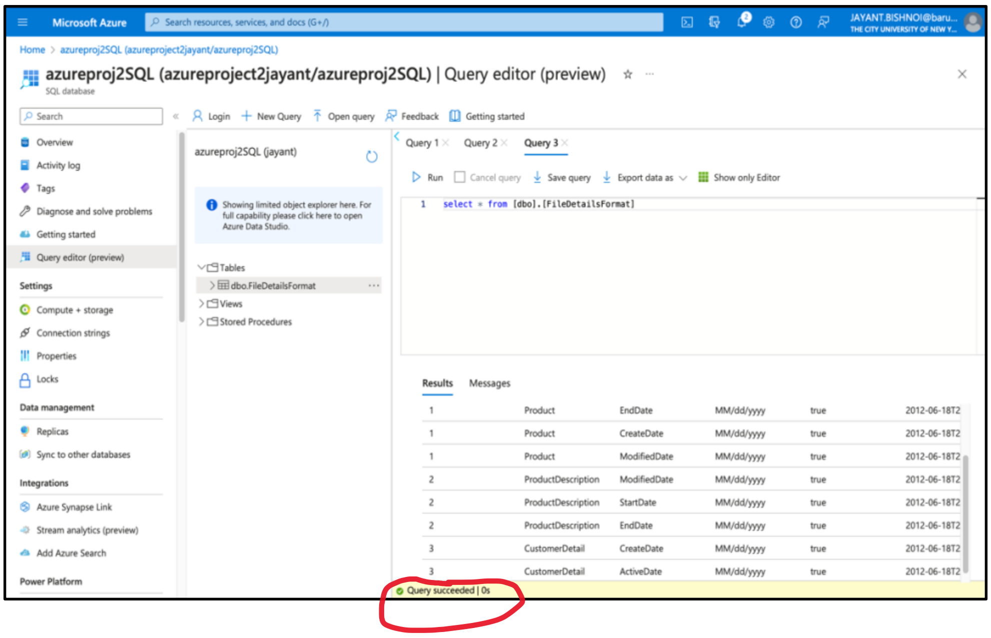
Resource group (move) : project2
Status : Online
Location : East US
Subscription (move) : Azure for Students
Subscription ID : 9282c55a-89eb-4d42-8d5c-d8d7e4186ed6
Tags (edit) : Click here to add tags
Server name : azureproject2jayant.database.windows.net
Connection strings : Show database connection strings
Pricing tier : General Purpose - Serverless: Gen5, 1 vCore
Auto-pause delay : 1 hour
Earliest restore point : No restore point available

Create schema table:

SSM – SQL Server Management System is an IDE, is an editor where you can run queries and operations related to a database.
If working on MAC, the alternative is to use AzureSQL- Edge container on docker and connect it to azure data studio.
I am choosing to use the AzureDB to make the data table and load the data.

In Azure SQl DB, go to the query editor and write the following code to make the schema table for the incoming files.

```
CREATE TABLE [FileDetailsFormat](
  [FileNo] [int] NOT NULL,
  [FileName] [nvarchar](100) NOT NULL,
  [ColumnName] [nvarchar](100) NULL,
  [ColumnDateFormat] [nvarchar](108) NULL,
  [ColumnIsNull] [nvarchar](100) NULL,
  [ModifiedDate] [datetime] NOT NULL
)
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(1, N'Product', N'StartDate', N'MM-dd-yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(1, N'Product', N'EndDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(1, N'Product', N'CreateDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(1, N'Product', N'ModifiedDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(2, N'ProductDescription', N'ModifiedDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(2, N'ProductDescription', N'StartDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(2, N'ProductDescription', N'EndDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(3, N'CustomerDetail', N'CreateDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
INSERT [dbo].[FileDetailsFormat] ([FileNo], [FileName], [ColumnName], [ColumnDateFormat], [ColumnIsNull], [ModifiedDate]) VALUES
(3, N'CustomerDetail', N'ActiveDate', N'MM/dd/yyyy', N'true', CAST(N'2012-06-18T22:34:09.000' AS DateTime))
```


Microsoft Azure
Search resources, services, and docs (G+/)
JAYANT.BISHNOI@baru...
THE CITY UNIVERSITY OF NEW Y...
Home > azureproj2SQL (azureproject2jayant/azureproj2SQL)

azureproj2SQL (azureproject2jayant/azureproj2SQL) | Query editor (preview)

SQL database

Showing limited object explorer here. For full capability please click here to open Azure Data Studio.

```
select * from [dbo].[FileDetailsFormat]
```

1	Product	EndDate	MM/dd/yyyy	true	2012-06-18T2
1	Product	CreateDate	MM/dd/yyyy	true	2012-06-18T2
1	Product	ModifiedDate	MM/dd/yyyy	true	2012-06-18T2
2	ProductDescription	ModifiedDate	MM/dd/yyyy	true	2012-06-18T2
2	ProductDescription	StartDate	MM/dd/yyyy	true	2012-06-18T2
2	ProductDescription	EndDate	MM/dd/yyyy	true	2012-06-18T2
3	CustomerDetail	CreateDate	MM/dd/yyyy	true	2012-06-18T2
3	CustomerDetail	ActiveDate	MM/dd/yyyy	true	2012-06-18T2

Query succeeded | 0s

Create Key vault and store secrets:

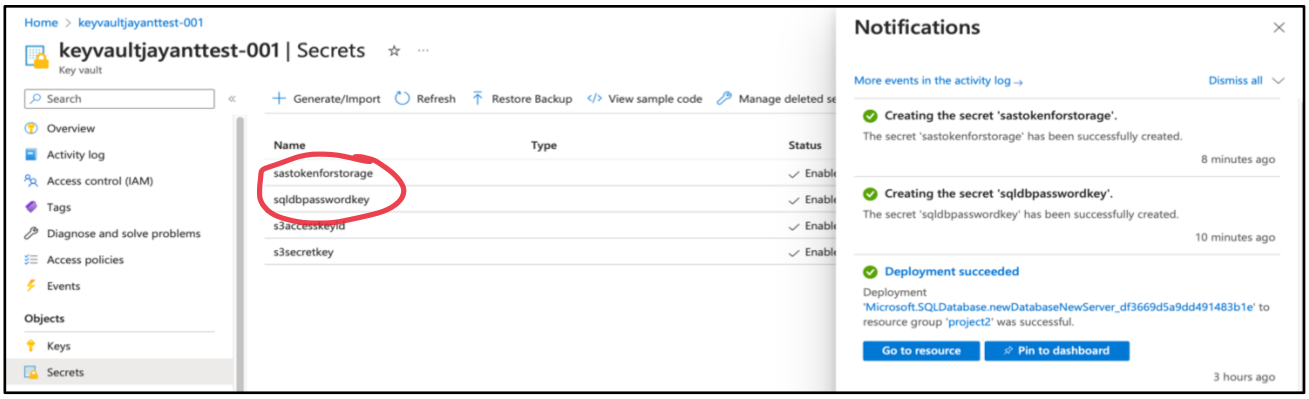
*"***We require SQL DB password & SAS token of storage account to able to connect to Databricks, we will store these secrets in key vault***"*

Azure key vault:

1. Go to azure portal, search for 'Key vault' and hit create, provide resource group (project
2. Give a suitable name, review +create.

Secrets:

1. Go to 'Secrets' and hit 'generate' to store the SQL 'DB password'
2. Create another secret for the 'shared account signature (sas) token'
3. Go to storage account and click on shared access signature and give all permissions for files and containers and objects and click create.
4. Copy the SAS token and paste it as a secret value when you are creating the secret in key vault and hit create.
5. Now Databricks will be able to access the SQL DB and the landing and staging folders with the SAS token by using these secrets in the key vault.



keyvaultjayanttest-001 | Secrets

Key vault

Name	Type	Status
sastokenforstorage		✓ Enabl
sqldbpasswordkey		✓ Enabl
s3accesskeyid		✓ Enabl
s3secretkey		✓ Enabl

Notifications

Creating the secret 'sastokenforstorage'.
The secret 'sastokenforstorage' has been successfully created.
8 minutes ago

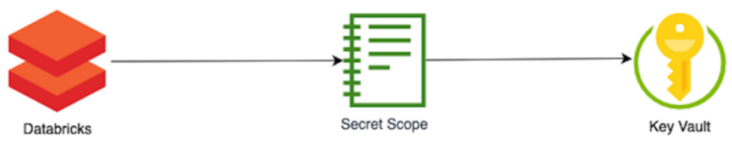
Creating the secret 'sqldbpasswordkey'.
The secret 'sqldbpasswordkey' has been successfully created.
10 minutes ago

Deployment succeeded
Deployment 'Microsoft.SQLDatabase.newDatabaseNewServer_df3669d5a9dd491483b1e' to resource group 'project2' was successful.

3 hours ago

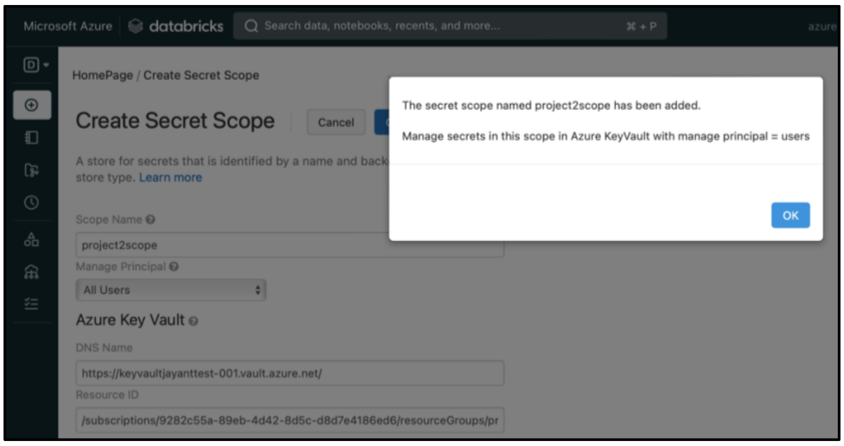
Create a secrets scope in Databricks:

"**We require Databricks to communicate with SQL DB to get schema, but it cannot go to the key vault to fetch the secrets directly. So, we define a secret scope in databricks which in turn communicates with key vault by deploying a notebook**"



Secret Scope:

1. Go to Databricks account that you have created and 'Launch Workspace'
2. At the end of your URL in search bar add **'secrets/createScope'**, get redirected to the create secret scopes page.
3. Give a suitable name < give permission to all users < paste key vault URI under DNS name & key vault resource ID under Resource ID < hit create.



HomePage / Create Secret Scope

Create Secret Scope

A store for secrets that is identified by a name and back
store type. Learn more

Scope Name: project2scope

Manage Principal: All Users

Azure Key Vault

DNS Name: https://keyvaultjayanttest-001.vault.azure.net/

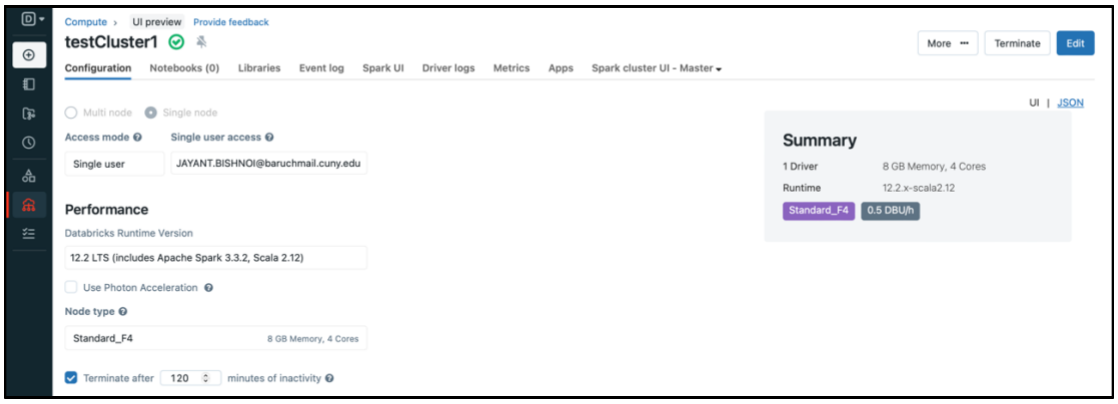
Resource ID: /subscriptions/9282c55a-89eb-4d42-8d5c-d8d7e4186ed6/resourceGroups/pr

The secret scope named project2scope has been added.

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

Create Cluster in Databricks:

1. Go to compute tab on the home page of your workspace and create cluster.
2. Give a suitable name and select runtime.
3. Choose worker type and driver type, we are selecting compute optimized standard F4.
4. Select the number of worker nodes that you need.
5. Create cluster.



testCluster1

Multi node [✓ Single node]

Access mode: Single user

Single user access: JAYANT.BISHNOI@baruchmail.cuny.edu

Performance

Databricks Runtime Version: 12.2 LTS (includes Apache Spark 3.3.2, Scala 2.12)

☐ Use Photon Acceleration

Node type: Standard_F4 — 8 GB Memory, 4 Cores

☑ Terminate after 120 minutes of inactivity

Summary

1 Driver	8 GB Memory, 4 Cores
Runtime	12.2.x-scala2.12

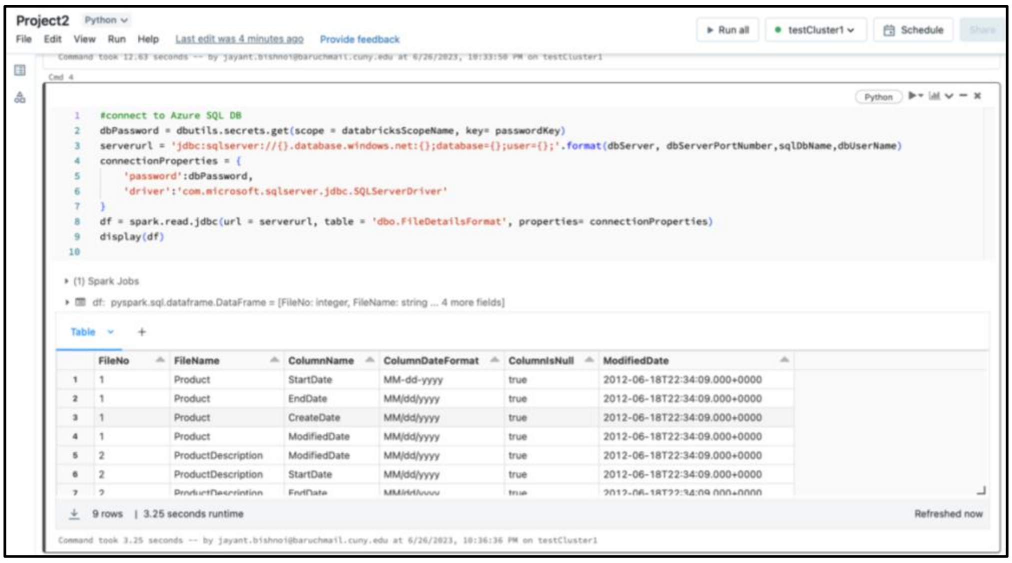
Standard_F4 0.5 DBU/h

Phase 2 – Implementing Validation

Creating Notebook in Databricks:

Inside the notebook we need to follow a series of steps:

1. Getting the **input** file
2. **Fetching the schema** from database, so we make a connection to the database, with the help of a few database properties (usernames & passwords, URL, port number etc...)



Project2 Python

File Edit View Run Help Last edit was 4 minutes ago Provide feedback

```
Command took 12.63 seconds -- by jayant.bishnoi@baruchmail.cuny.edu at 6/26/2023, 10:33:50 PM on testCluster1
```

Cmd 4

```python
#connect to Azure SQL DB
dbPassword = dbutils.secrets.get(scope = databricksScopeName, key= passwordKey)
serverurl = 'jdbc:sqlserver://{}.database.windows.net:{};database={};user={};'.format(dbServer, dbServerPortNumber,sqlDbName,dbUserName)
connectionProperties = {
    'password':dbPassword,
    'driver':'com.microsoft.sqlserver.jdbc.SQLServerDriver'
}
df = spark.read.jdbc(url = serverurl, table = 'dbo.FileDetailsFormat', properties= connectionProperties)
display(df)
```

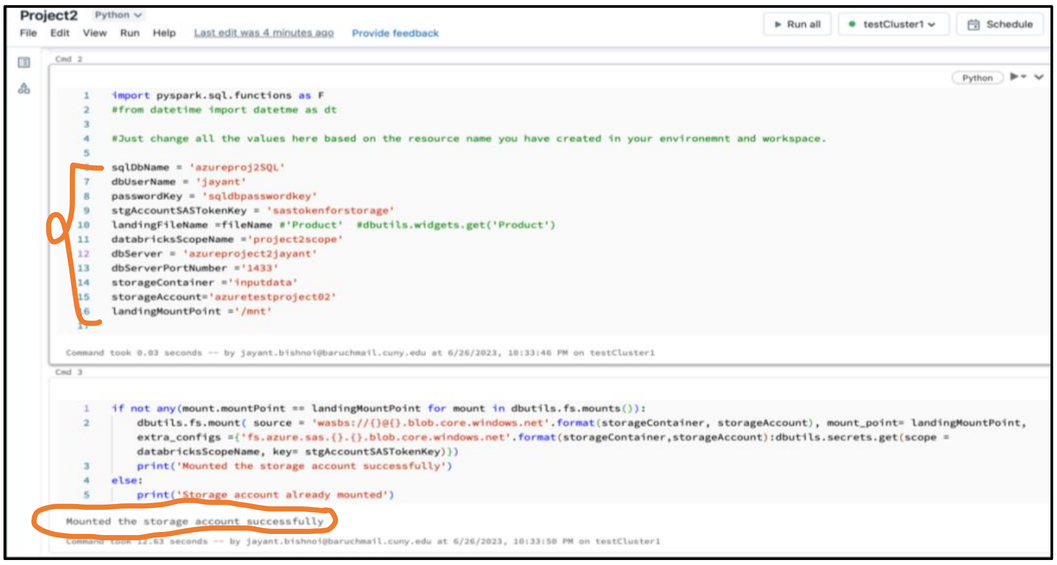
```
(1) Spark Jobs
df: pyspark.sql.dataframe.DataFrame = [FileNo: integer, FileName: string ... 4 more fields]
```

	FileNo	FileName	ColumnName	ColumnDateFormat	ColumnIsNull	ModifiedDate
1	1	Product	StartDate	MM-dd-yyyy	true	2012-06-18T22:34:09.000+0000
2	1	Product	EndDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000
3	1	Product	CreateDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000
4	1	Product	ModifiedDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000
5	2	ProductDescription	ModifiedDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000
6	2	ProductDescription	StartDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000
7	2	ProductDescription	EndDate	MM/dd/yyyy	true	2012-06-18T22:34:09.000+0000

```
9 rows | 3.25 seconds runtime                                     Refreshed now
Command took 3.25 seconds -- by jayant.bishnoi@baruchmail.cuny.edu at 6/26/2023, 10:36:36 PM on testCluster1
```

We have used the scope we defined in databricks earlier to access the password from key vault to make the connection to the database, which is successful.

3. We **mount the storage account** as ADF will only fetch the file name and the real access to the files will come from the storage account (SAS token, storage info.)



Project2 Python

File Edit View Run Help Last edit was 4 minutes ago Provide feedback

Cmd 2

```python
import pyspark.sql.functions as F
#from datetime import datetme as dt

#Just change all the values here based on the resource name you have created in your environemnt and workspace.

sqlDbName = 'azureproj2SQL'
dbUserName = 'jayant'
passwordKey = 'sqldbpasswordkey'
stgAccountSASTokenKey = 'sastokenforstorage'
landingFileName =fileName #'Product'  #dbutils.widgets.get('Product')
databricksScopeName ='project2scope'
dbServer = 'azureproject2jayant'
dbServerPortNumber ='1433'
storageContainer ='inputdata'
storageAccount='azuretestproject02'
landingMountPoint ='/mnt'
```

```
Command took 0.03 seconds -- by jayant.bishnoi@baruchmail.cuny.edu at 6/26/2023, 10:33:46 PM on testCluster1
```

Cmd 3

```python
if not any(mount.mountPoint == landingMountPoint for mount in dbutils.fs.mounts()):
    dbutils.fs.mount( source = 'wasbs://{}@{}.blob.core.windows.net'.format(storageContainer, storageAccount), mount_point= landingMountPoint,
    extra_configs ={'fs.azure.sas.{}.{}.blob.core.windows.net'.format(storageContainer,storageAccount):dbutils.secrets.get(scope =
    databricksScopeName, key= stgAccountSASTokenKey)})
    print('Mounted the storage account successfully')
else:
    print('Storage account already mounted')
```

```
Mounted the storage account successfully
Command took 12.63 seconds -- by jayant.bishnoi@baruchmail.cuny.edu at 6/26/2023, 10:33:50 PM on testCluster1
```

4. Then at last we **implement the validation** logic to check for duplication and data format.

1. For implementing the **elimination of duplicate values**, we will take the total count of rows and distinct count of rows, if both values are equal, we can concur that there is no duplication of rows.

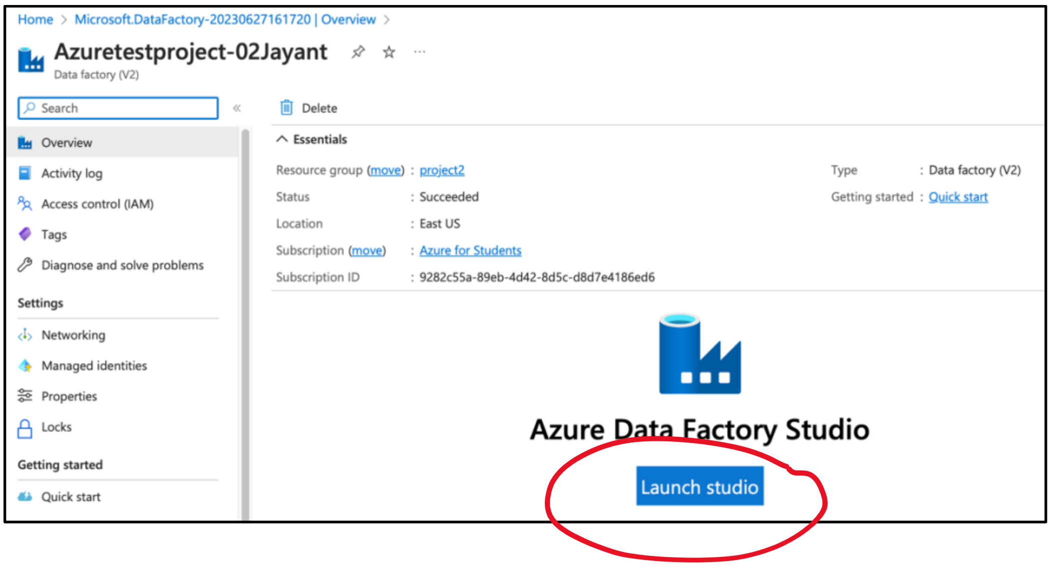
2. For finding **correct date format**, we will filter all records where filename is not what you require, in our case (product), we only need column name and date format, iterate row by row.
3. We use a **'todate'** function to pass the values in the rows and if a value is returned, we know that it's the correct format and value is <u>not null</u>.

```
df1 = spark.read.csv('/mnt/landing/'+fileName, inferSchema=True, header=True)
#display(df1)

# Rule
errorFlag=False
errorMessage = ''
totalcount = df1.count()
print(totalcount)
distinctCount = df1.distinct().count()
print(distinctCount)
if distinctCount !=totalcount:
    errorFlag = True
    errorMessage = 'Duplication Found. Rule 1 Failed'
print(errorMessage)

# Rule 2
df2 =
df.filter(df.FileName==fileNameWithoutExt).select('ColumnName','ColumnDateFormat' )
rows = df2.collect()
for r in rows:
    colName = r[0]
    colFormat =r[1]
    print(colName, colFormat)
    #display(df1.filter(F.to_date(colName, colFormat).isNull() ==True))
    formatCount =df1.filter(F.to_date(colName, colFormat).isNotNull()
==True).count()
    if formatCount == totalcount:
        errorFlag = True
        errorMessage = errorMessage +' DateFormate is incorrect for {}
'.format(colName)
    else:
        print('All rows are good for ', colName)
print(errorMessage)

if errorFlag:
    dbutils.fs.mv('/mnt/landing/'+fileName,'/mnt/rejected/'+fileName )
    dbutils.notebook.exit('{"errorFlag": "true", "errorMessage":"'+errorMessage
+'"}')
else:
    dbutils.fs.mv('/mnt/landing/'+fileName,'/mnt/staging/'+fileName )
    dbutils.notebook.exit('{"errorFlag": "false", "errorMessage":"No error"}')
```

Phase 3 – Data Migration

Creating Azure Data Factory:

1. Go to azure portal, search for data factories, and hit create, provide resource group that we created (project 1).
2. Give a suitable name, give Git configuration later and hit review + create.
3. Go to resource and launch ADF, where we will create pipelines to migrate data.



Resource group (move) : project2
Status : Succeeded
Location : East US
Subscription (move) : Azure for Students
Subscription ID : 9282c55a-89eb-4d42-8d5c-d8d7e4186ed6
Type : Data factory (V2)
Getting started : Quick start

Creating Linked Service to Databricks in Azure Data Factory:

"We need to connect to databricks account so we will create a linked service"

1. Go to ADF < launch studio, go to the manage tab and create new linked service.
2. In the compute tab search for Azure databricks, give a suitable name & description.
3. Choose 'AutoResolveIntegrationRuntime' < choose subscription < databricks workspace
4. Select 'New job cluster'

4.1. Create access token:

4.1.1. Go to user settings tab in the databricks homepage and click on < 'Generate Token' < give token description < create and copy token.

4.2. Create secret and access policy in key vault:

4.2.1. Following industry best practices, we store the databricks token as a secret in azure key vault

4.2.2. Go to your key vault < secrets < generate < give name and paste the databricks token < create

"Now we need to give Azure data Factory permission to access this key vault."

4.3. Create access Policy:

4.3.1. In the key vault < access policy < Key, Secret & Certificate management.

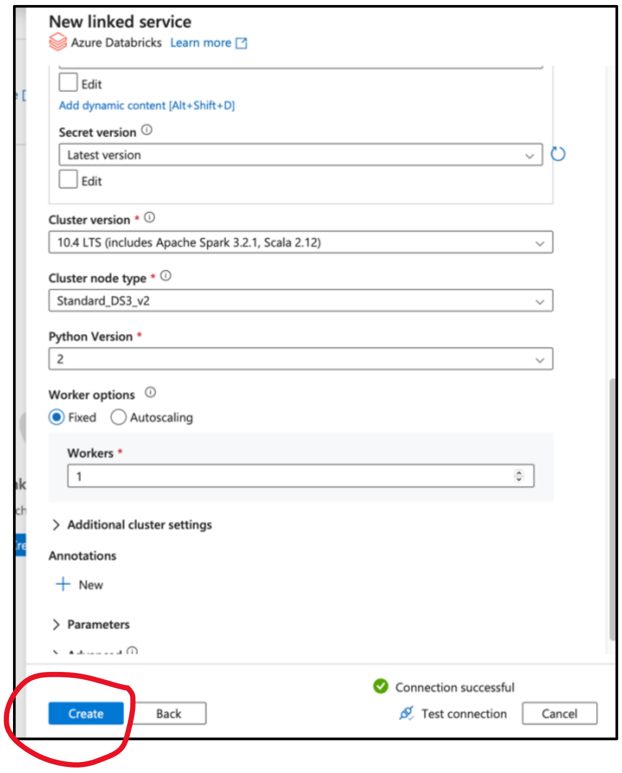
4.3.2. In the principal add the name of your Azure Data Factory account < add

"Go back to your linked service and click now on key vault instead of access token."

5. Create Linked service for key vault:
6. Create a new linked service for this key vault < select key vault name < Test connection <create.

7. Select the name of your newly created linked service.
8. Select secret name "databrickstoken"
9. Select cluster version and cluster node type.

10. Select the number of workers.
11. Test connection and create.



New linked service
Azure Databricks Learn more

Edit
Add dynamic content [Alt+Shift+D]
Secret version
Latest version
Edit

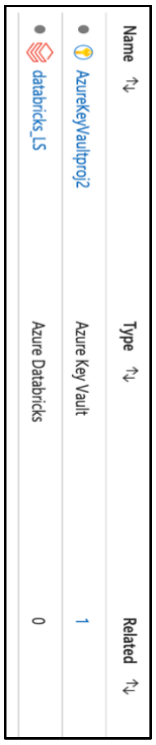
Cluster version *
10.4 LTS (includes Apache Spark 3.2.1, Scala 2.12)

Cluster node type *
Standard_DS3_v2

Python Version *
2

Worker options
Fixed (selected) Autoscaling

Workers *
1



Name	Type	Related
AzureKeyVaultproj2	Azure Key Vault	1
databricks_LS	Azure Databricks	0

<u>Creating Linked Service to Datalake storage in Azure Data Factory</u>:

"ADF need to connect to both landing folder and databricks, so we set up another linked service"

Go to ADF < click on manage < create new.
Give a suitable name and description.
Select Azure data lake gen2 and give storage account name.
Test connection and hit create.

<u>Creating Pipeline to call the Databricks notebook</u>:

1. Go to author tab in ADF < create new pipeline.
2. Search for 'notebook' < select databricks notebook and select the associated linked service.
3. Test the connection < under settings tab < provide notebook path

4. *"We want databricks to receive the name of the file from ADF, so we need to add a **base parameter**"*

5. <u>Add a Base parameter:</u>
   5.1. Create base parameter < under name write 'fileName' since that is how the code is witten

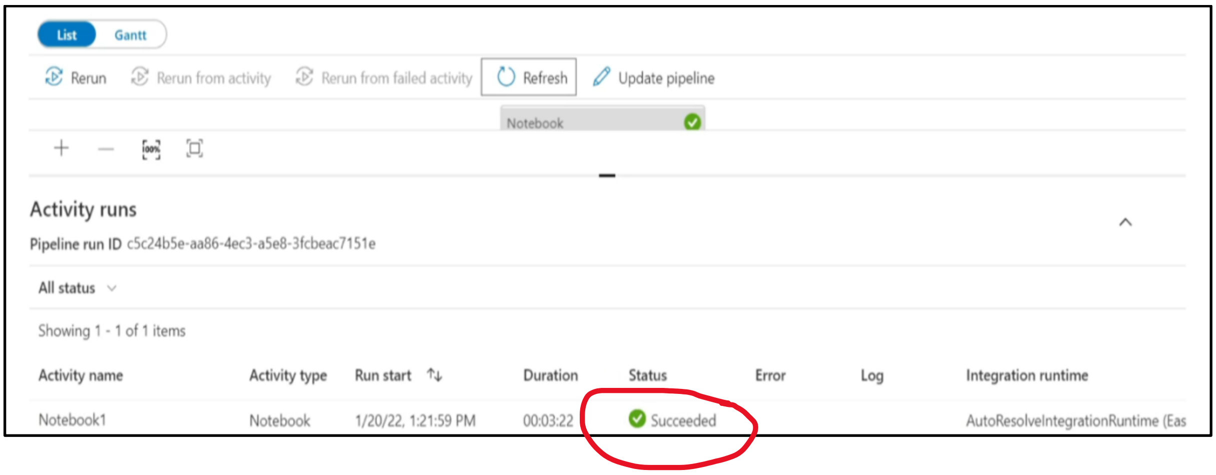
6. <u>Add a trigger</u>:
   6.1.1. Since we want the pipeline execution to happen as soon as a file comes in landing folder, we need add a storage trigger.
   6.1.2. Go to the add trigger button and add new trigger <give a suitable name.
   6.1.3. Type < storage events
   6.1.4. Select storage account name and container name (input data)

6.1.5. Blob path begins with 'landing/', so that any file in landing folder triggers pipeline.
6.1.6. Blob path ends with '.csv' to only execute for csv files.
6.1.7. Continue and create.

7. <u>Add a pipeline parameter:</u>
    7.1. Click outside notebook and click on parameters tab < create new.
    7.2. Under name write 'fileName'
    7.3. Go to trigger and edit value **'@triggerBody.fileName'** , this function will give the actual name of the file.
    7.4. Now go to the notebook parameter, we will pass the pipeline parameter in the value under the settings tab.

"We are fetching the name of the file from trigger and feeding it to pipeline parameter → Notebook parameter, so that the code in our notebook is relevant for any file that is uploaded automatically."

8. Publish.



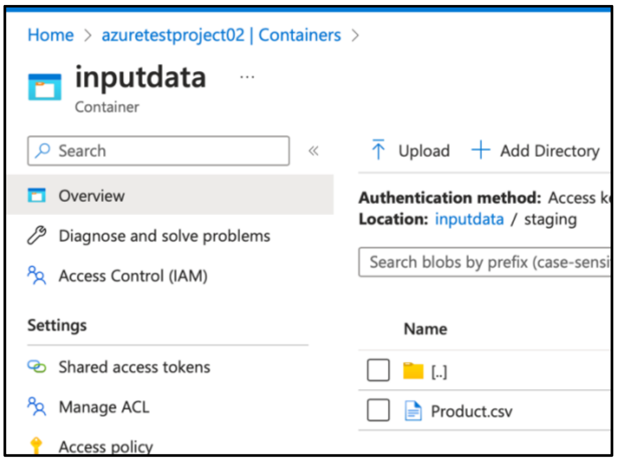
Activity runs

Pipeline run ID c5c24b5e-aa86-4ec3-a5e8-3fcbeac7151e

All status

Showing 1 - 1 of 1 items

Activity name	Activity type	Run start	Duration	Status	Error	Log	Integration runtime
Notebook1	Notebook	1/20/22, 1:21:59 PM	00:03:22	Succeeded			AutoResolveIntegrationRuntime (Eas



Authentication method: Access k
Location: inputdata / staging

Name
[..]
Product.csv

(Staging folder has been created automatically, moving the file from landing folder)

<u>THE END</u>