

Project 02: Customer Behavior Analysis Documentation

1. Project Overview

What is ETL?

ETL stands for Extract, Transform, Load and is a critical process in data integration and analytics pipelines. The purpose of ETL is to collect data from various sources, transform it into a usable format, and load it into a storage system for analysis and reporting. Here's a breakdown of each step:

- **Extract:** Collecting raw data from multiple sources (e.g., databases, APIs, or files).
- **Transform:** Cleaning, filtering, and formatting the extracted data to align with business requirements.
- **Load:** Storing the processed data into a data warehouse, database, or storage system for further use.

Objective:

To automate the ETL (Extract, Transform, Load) workflow for an Online Retail Analysis project using Azure Data Factory, Azure Databricks, and Azure Git. This project utilizes the Medallion Architecture, which includes Bronze, Silver, and Gold layers, to prepare data for predictive analysis.

Dataset:

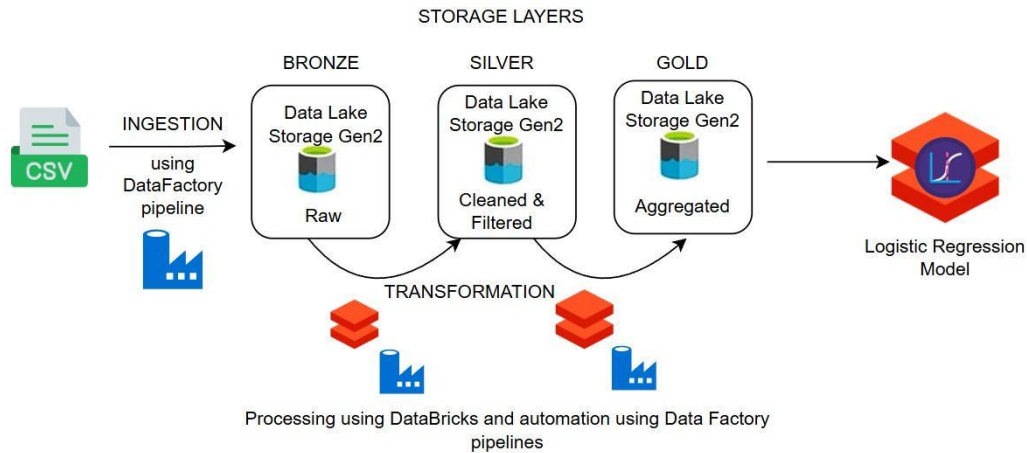
The dataset, "**Online Retail.xlsx**", contains transactional data from a UK-based online retail company. It includes the following columns:

- **InvoiceNo:** Unique identifier for each transaction.
- **StockCode:** Unique identifier for each product.
- **Description:** Product name or description.
- **Quantity:** Number of products sold per transaction.
- **InvoiceDate:** Date and time of the transaction.
- **UnitPrice:** Price per unit of the product.
- **CustomerID:** Unique identifier for each customer.
- **Country:** Country of the customer.

The dataset will be used for customer behavior analysis, focusing on metrics like purchase patterns, sessionized interactions, and churn prediction.

For a detailed walkthrough of the project, please refer to the [project explanation video](#).

Architecture:



=> Setting Up the Environment:-

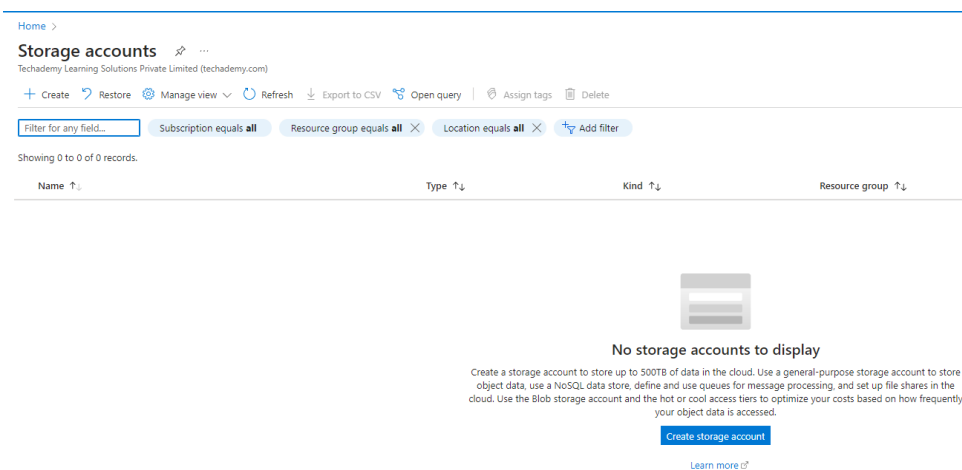
1. Create Storage Account, Containers, and Upload Raw Data

Purpose:

- The storage account serves as the Azure Data Lake Gen2, providing distributed data storage for the ETL process.
- Containers help organize data based on the Medallion Architecture.

Steps:

1. Navigate to Storage Accounts > New Storage Account.



2. Fill in required details (e.g., subscription, resource group) and click Review + Create.

[Home](#) > [Storage accounts](#) >

Create a storage account

Basics Advanced Networking Data protection Encryption Tags Review + create

Azure Storage is a Microsoft-managed service providing cloud storage that is highly available, secure, durable, scalable, and redundant. Azure Storage includes Azure Blobs (objects), Azure Data Lake Storage Gen2, Azure Files, Azure Queues, and Azure Tables. The cost of your storage account depends on the usage and the options you choose below. [Learn more about Azure storage accounts](#)

Project details

Select the subscription in which to create the new storage account. Choose a new or existing resource group to organize and manage your storage account together with other resources.

Subscription *

Resource group *
[Create new](#)

Instance details

Storage account name *

Region *
[Deploy to an Azure Extended Zone](#)

Primary service

Performance * ☒ Standard: Recommended for most scenarios (general-purpose v2 account)
☐ Premium: Recommended for scenarios that require low latency.

Redundancy *
☒ Make read access to data available in the event of regional unavailability.

[Previous](#) [Next](#) [Review + create](#)

3. Under the Basic tab, select Data Lake Storage Gen2.

[Deploy to an Azure Extended Zone](#)

Primary service

Performance *

Redundancy *

4. Enable Hierarchical namespace under the Advanced tab.

[Home](#) > [Storage accounts](#) >

Create a storage account

Basics **Advanced** Networking Data protection Encryption Tags Review + create

Security

Configure security settings that impact your storage account.

Require secure transfer for REST API operations ☒

Allow enabling anonymous access on individual containers ☐

Enable storage account key access ☒

Default to Microsoft Entra authorization in the Azure portal ☐

Minimum TLS version

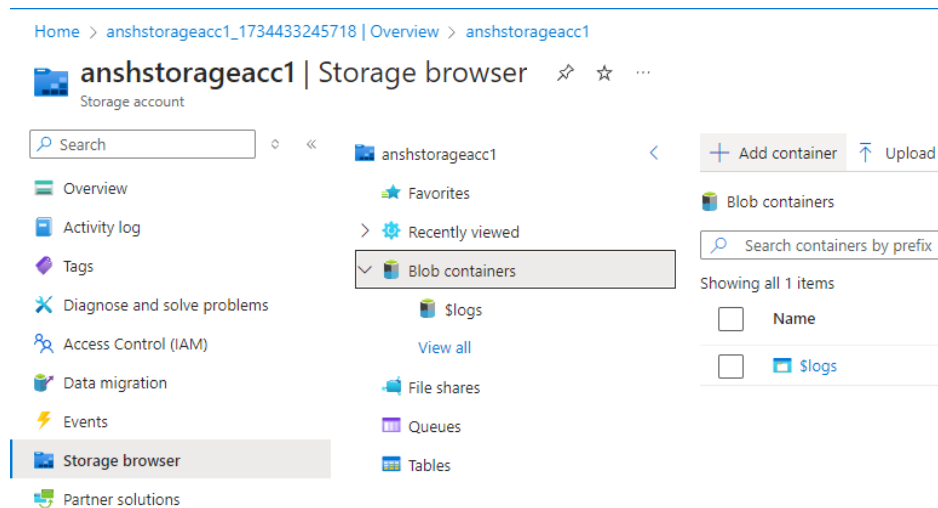
Permitted scope for copy operations (preview)

Hierarchical Namespace

Hierarchical namespace, complemented by Data Lake Storage Gen2 endpoint, enables file and directory semantics, accelerates big data analytics workloads, and enables access control lists (ACLs) [Learn more](#)

Enable hierarchical namespace ☒

5. Once deployed, go to Storage Browser > Blob Containers > Add Container.



6. Create four containers:

- One for each layer: bronze, silver, and gold.

New container

Name *

bronzelayer

Anonymous access level ⓘ

Private (no anonymous access)

The access level is set to private because anonymous access is disabled on this storage account.

Advanced

anshstorageacc1

Favorites

Recently viewed

Blob containers

- \$logs
- bronzelayer
- goldlayer
- silverlayer

View all

File shares

+ Add container

Upload

Blob containers

Search containers by prefix

Showing all 4 items

<input type="checkbox"/>	Name
<input type="checkbox"/>	\$logs
<input type="checkbox"/>	bronzelayer
<input type="checkbox"/>	goldlayer
<input type="checkbox"/>	silverlayer

- One for raw data (rawdata).

718 | Overview > anshstorageacc1

Storage browser

anshstorageacc1

Favorites

Recently viewed

Blob containers

- \$logs
- bronzelayer
- goldlayer
- rawdata
- silverlayer

View all

File shares

Queues

Tables

+ Add Directory

Upload

Refresh

Delete

Copy

Paste

Rename

Acquire lease

Release lease

Blob containers > rawdata

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive)

Showing all 0 items

<input type="checkbox"/>	Name	Last modified	Access
No items found			

Upload blob

*** Uploading on blob(s)...
Attempting to upload 1 blob(s)

Drag and drop files here
or
Browse for files

☐ Overwrite if files already exist

Advanced

Upload

Give feedback

Current uploads

Online Retail.xlsx 0 / 22.62 MiB

Dismiss: Completed All

7. Upload raw data files to the rawdata container.

anshstorageacc1

Favorites

Recently viewed

Blob containers

- \$logs
- bronzelayer
- goldlayer
- rawdata
- silverlayer

+ Add Directory

Upload

Refresh

Delete

Blob containers > rawdata

Authentication method: Access key (Switch to Microsoft Entra user account)

Search blobs by prefix (case-sensitive)

Showing all 1 items

<input type="checkbox"/>	Name
<input type="checkbox"/>	Online Retail.xlsx

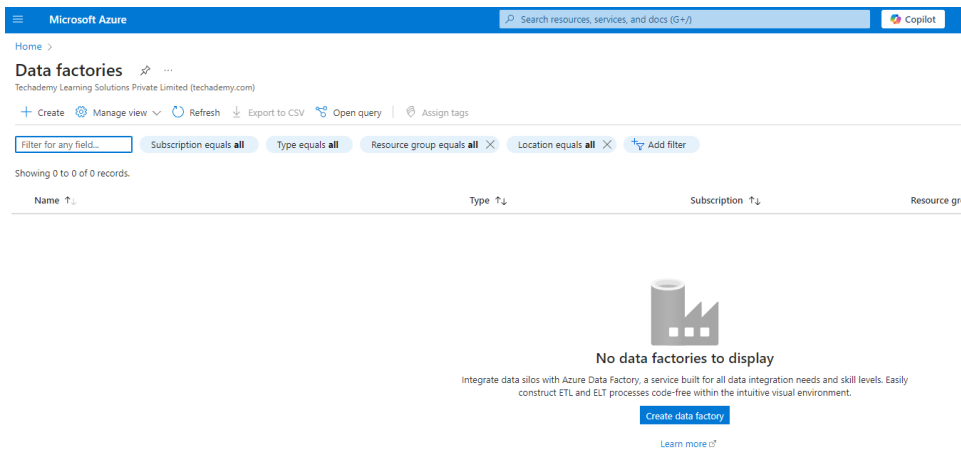
2. Create Data Factory

Purpose:

- **Azure Data Factory orchestrates the ETL workflow by automating data ingestion, transformations, and movement between layers.**

Steps:

1. **Go to Azure Data Factories > Create a data factory.**



2. **Provide a unique name for your data factory and click Review + Create.**

Home > Data factories >

Create Data Factory

Basics Git configuration Networking Advanced Tags Review + create

One-click to create data factory with sample pipeline and datasets. [Try it](#)

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ MML Learners

Resource group * ⓘ rg-azuser2415_mml.local-ga0V5 [Create new](#)

Instance details

Name * ⓘ anshdatafactory123 ✓

Region * ⓘ East US

Version * ⓘ V2

3. **After deployment, the Data Factory is ready to create pipelines for automation.**

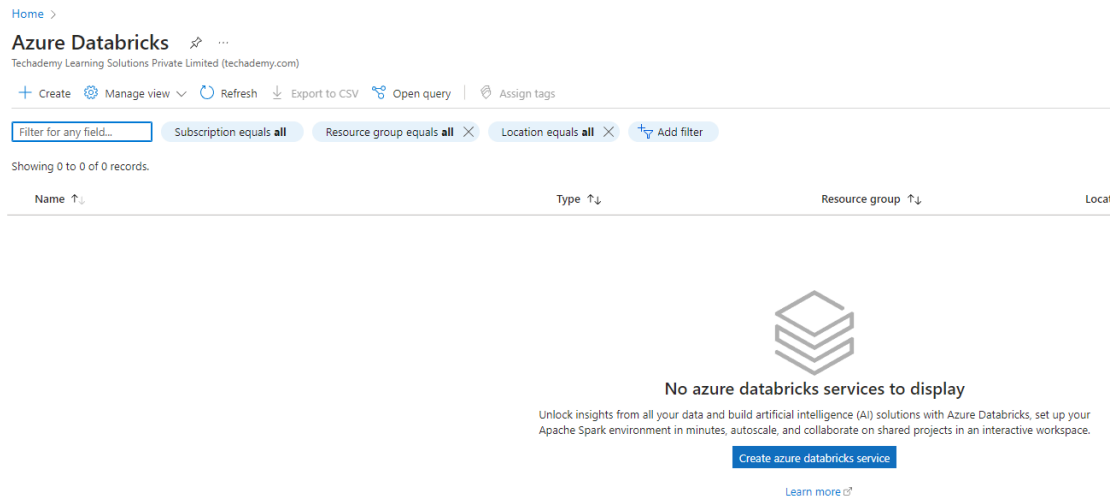
3. Create Databricks Workspace

Purpose:

- **Azure Databricks enables scalable data transformations and advanced analytics using Spark.**

Steps:

1. Navigate to Azure Databricks > Create Azure Databricks Service.



2. Enter details (e.g., subscription, resource group) and click Review + Create.

Home > Azure Databricks >

Create an Azure Databricks workspace

Basics Networking Encryption Security & compliance Tags Review + create

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * ⓘ MML Learners

Resource group * ⓘ rg-azuser2415_mml.local-ga0V5

[Create new](#)

Instance Details

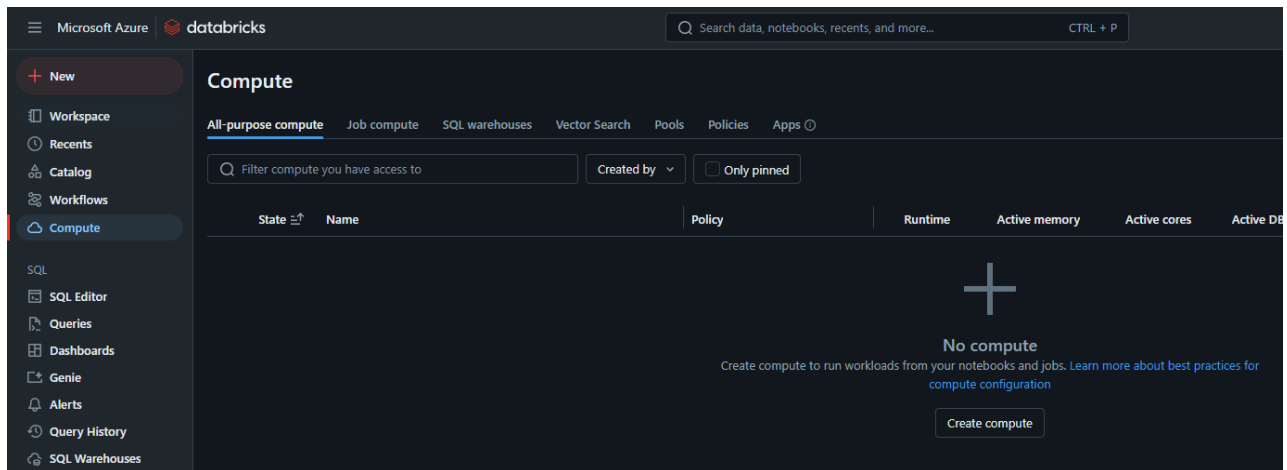
Workspace name * DataBricks1 ✓

Region * East US ✓

Pricing Tier * ⓘ Trial (Premium - 14-Days Free DBUs) ✓

Managed Resource Group name Enter name for managed resource group

3. Once deployed, launch the workspace.



4. Go to Compute > Create Compute to set up a cluster.

Compute > New compute

azuser2415_mml.local's Cluster

Policy

Unrestricted

☒ Multi node ☐ Single node

Access mode Single user access

Single user azuser2415_mml.local

Performance

Databricks runtime version

Runtime: 15.4 LTS (Scala 2.12, Spark 3.5.0)

☒ Use Photon Acceleration

Worker type Min workers Max workers

Standard_D4ds_v5 16 GB Memory, 4 Cores 1 1 ☐ Spot instances

Driver type

Same as worker 16 GB Memory, 4 Cores

☒ Enable autoscaling

☒ Terminate after 120 minutes of inactivity

Tags

Add tags

Key Value Add

> Automatically added tags

> Advanced options

Create compute Cancel

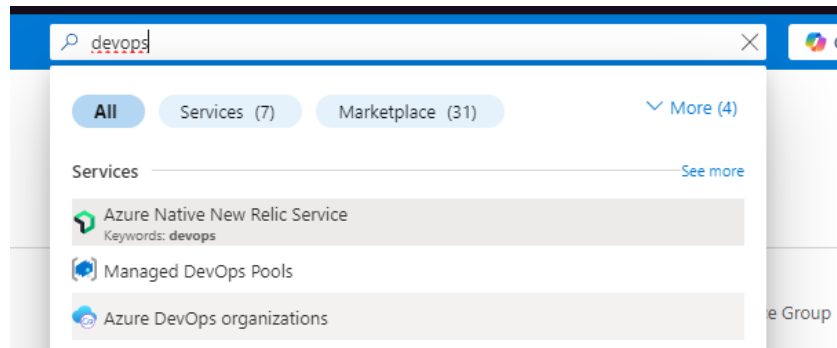
4. Set Up Azure Git Repository

Purpose:

- Azure Git repository provides version control and collaboration.
- The repository integrates with Databricks to enable seamless notebook management.

Steps:

1. Search for Azure DevOps and navigate to My Azure DevOps Organizations.



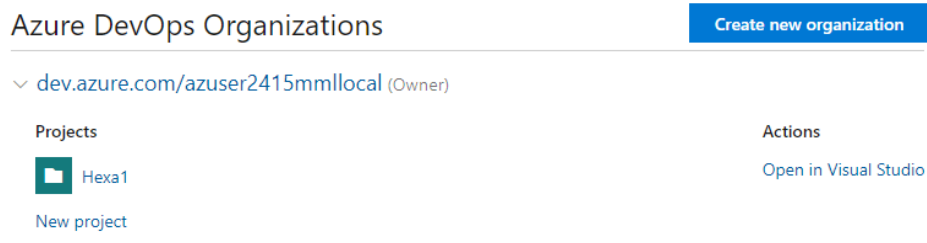
Azure DevOps

Plan smarter, collaborate better, and ship faster with a set of modern dev services

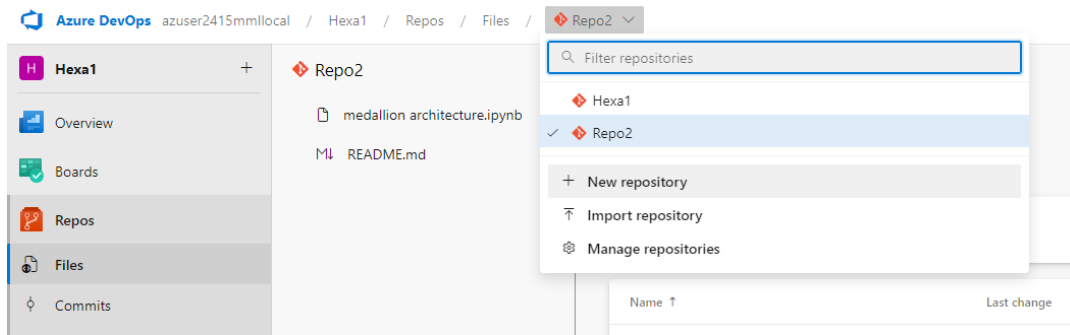
[My Azure DevOps Organizations](#)

[Get started using Azure DevOps](#)

2. Select or create an organization.



3. Go to Repos > New Repository and name it (e.g., Online Retail Analysis).



Create a repository

Repository type

Git

Repository name *

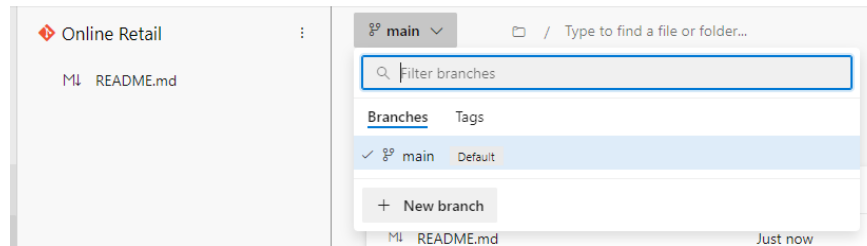
Online Retail

☒ Add a README

Add a .gitignore: None

Your repository will be initialized with a main branch.

4. Create two additional branches: development and production.



Create a branch

Name *

development

Based on

main

Work items to link

Search work items by ID or title

Cancel Create

Branches

Mine All Stale

Branch

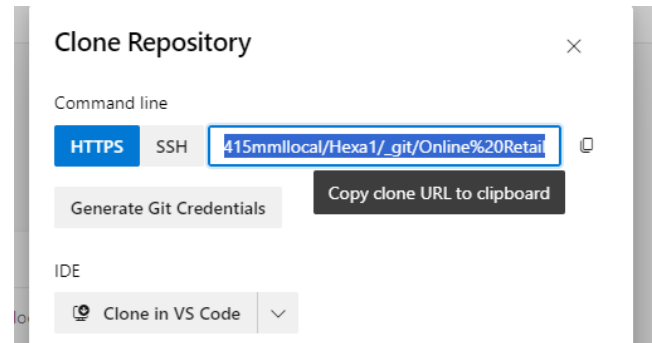
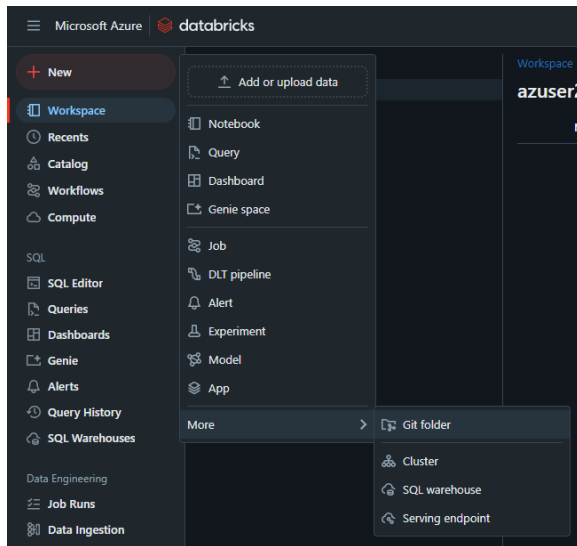
development

main

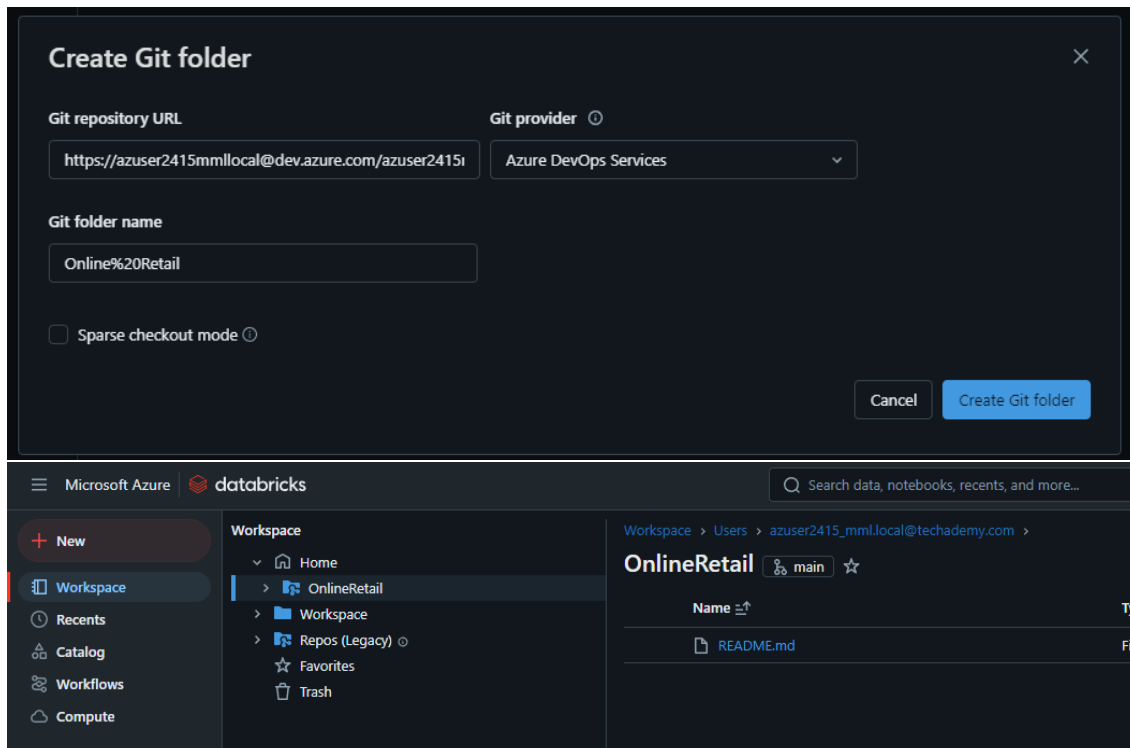
Default Compare

production

5. In Databricks, go to Workspace > More > Git Integration.



6. Copy the repository URL from Azure DevOps and paste it into the Git configuration in Databricks.



5. Configure CI/CD Pipelines

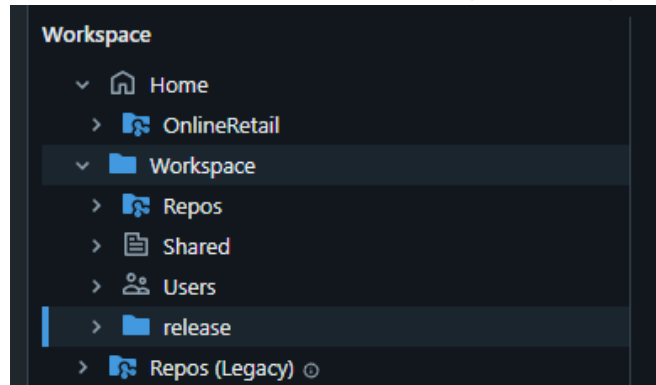
Purpose:

- Automates code deployment and ensures the latest changes are available in production and the release folder.
- Simplifies collaboration and reduces manual intervention.

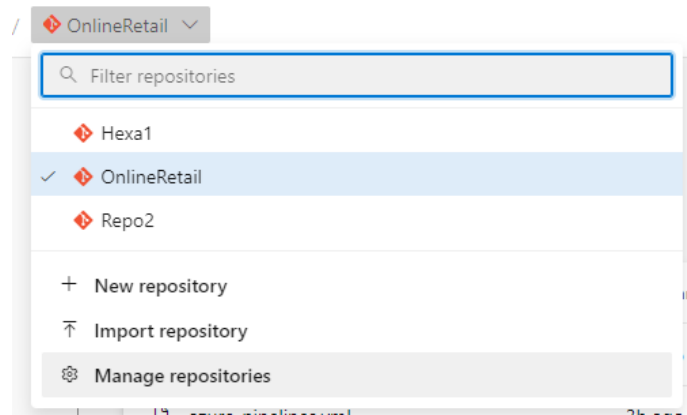
Steps:

1. Set up two CI/CD pipelines:
 - Pipeline 1: Automatically merges changes from the main branch to the production branch.
 - Pipeline 2: Updates a release folder in the Databricks workspace with the latest code from the main branch.
2. Use Azure DevOps to configure the pipelines with a YAML script that includes tasks like:
 - Checking out the repository.
 - Installing the Databricks CLI.
 - Merging branches.
 - Deploying updates to the release folder.

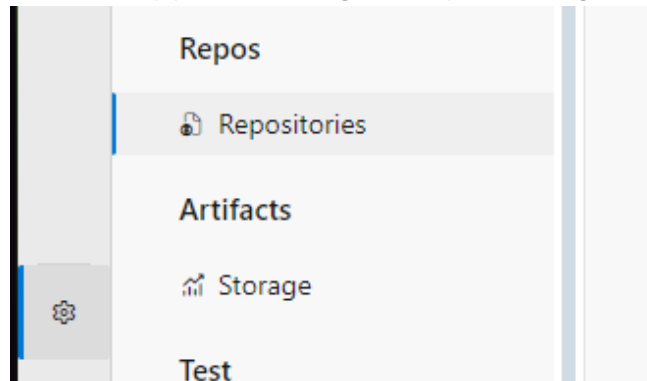
1. Create a folder called release inside your workspace



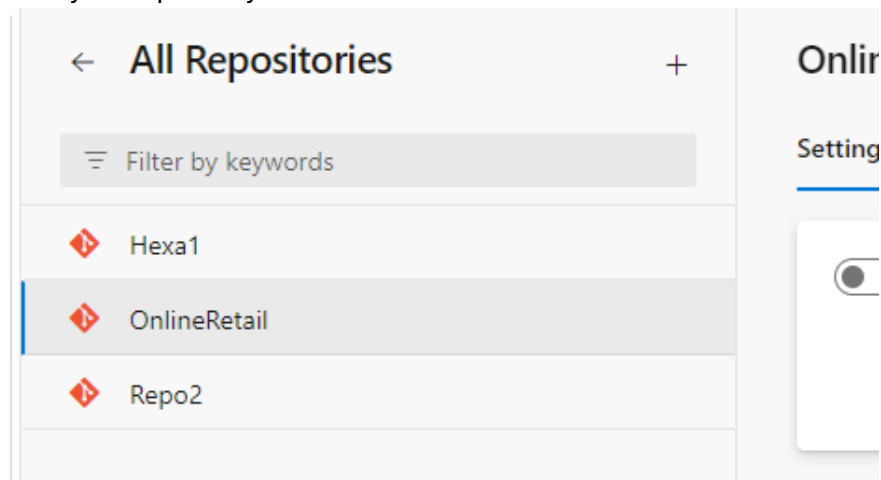
2. Inside your Repo tab go to Manage Repositories



3. Alternatively you can also go to Project Settings on bottom left > Repository side tab



4. Pick your repository from menu



5. Go to Security tab and under Users go to Build Services

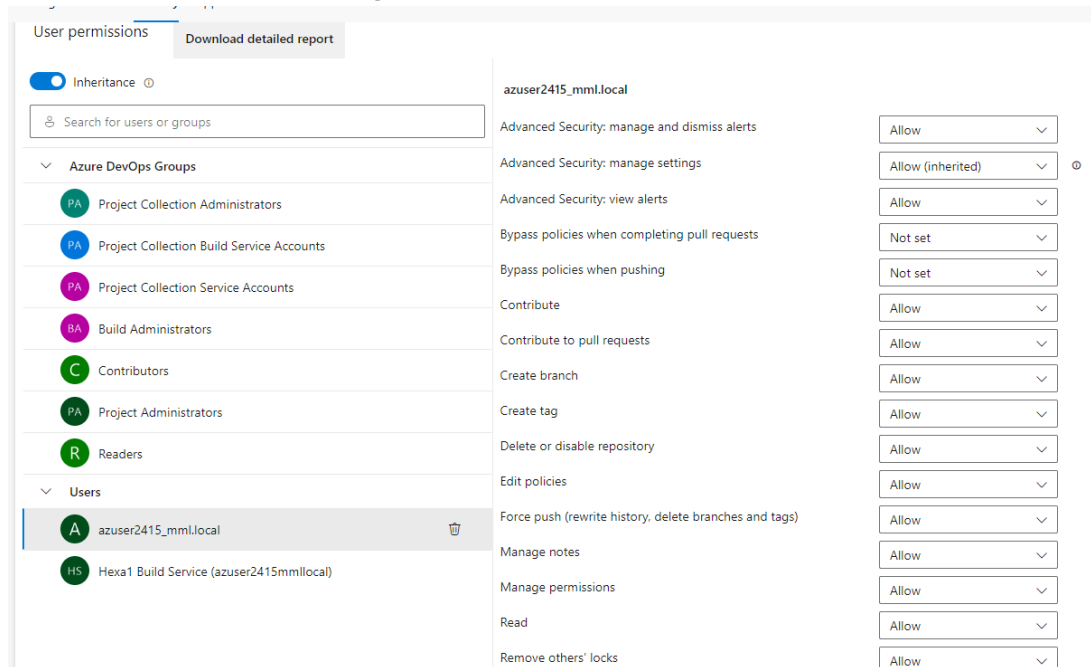
The screenshot shows the 'OnlineRetail' repository page in the Security tab. On the left, a sidebar lists repositories: 'All Repositories', 'Hexa1', 'OnlineRetail' (selected), and 'Repo2'. The main content area is titled 'OnlineRetail' and has tabs for 'Settings', 'Policies', 'Security' (active), and 'Approvals and checks'. Under the 'Security' tab, there's a 'User permissions' section with a 'Download detailed report' button. Below this, there's a search bar and a list of users and groups. The 'Azure DevOps Groups' section is expanded, showing 'Project Collection Administrators', 'Project Collection Build Service Accounts', 'Project Collection Service Accounts', 'Build Administrators', 'Contributors', 'Project Administrators', and 'Readers'. The 'Users' section is also expanded, showing 'azuser2415_mml.local' and 'Hexa1 Build Service (azuser2415mmllocal)' (selected).

the bottom most option here

6. There are 3 settings : 'Contribute' , 'Contribute to pull requests' , 'Create branch'. Make sure they are all set to Allow

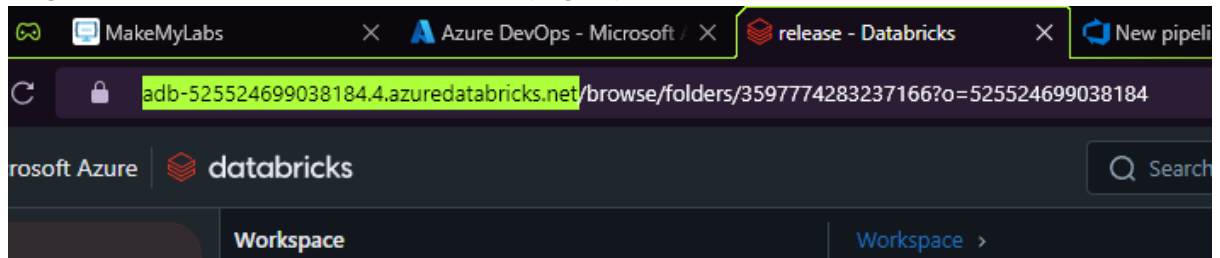
The screenshot shows the 'OnlineRetail' repository settings page. On the left, there's a list of settings: 'Contribute', 'Contribute to pull requests', 'Create branch', and 'Create tag'. On the right, there's a dropdown menu for each setting. The 'Contribute' dropdown is open, showing 'Not set', 'Allow', and 'Deny'. The 'Contribute to pull requests' dropdown is set to 'Allow'. The 'Create branch' dropdown is set to 'Allow'. Each dropdown has a green checkmark next to it.

7. Now go to under users title go to your <username> tab just above this Build Service and make sure these 3 settings are set as allow there as well

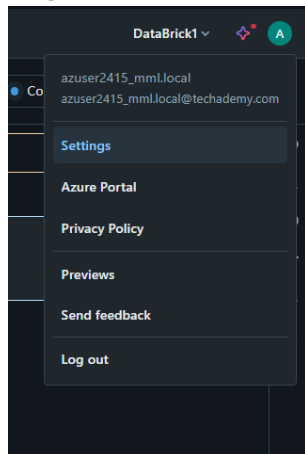


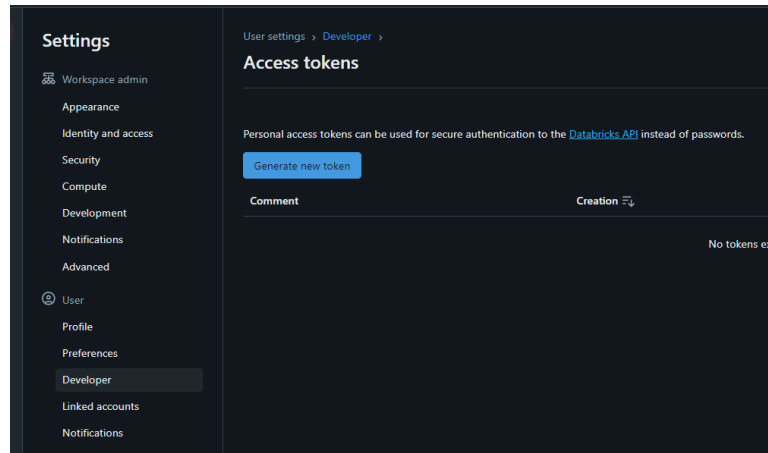
8. We need to save 2 variables in a variable group databricksHost and databricksToken. This is like username and password to log into databricks

9. To get the databricksHost > The url of our page upuntil .net is our databricksHost

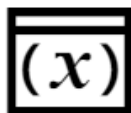
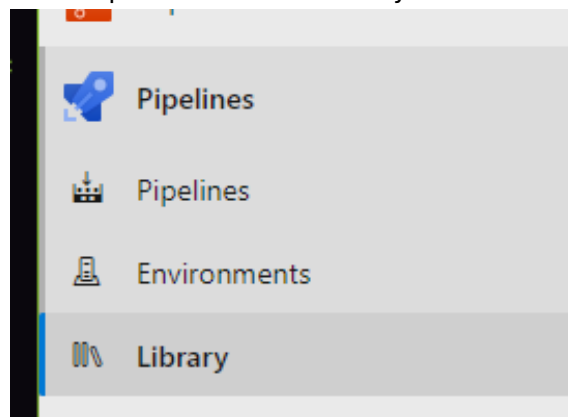


10. To get databricksToken, Go to settings > Developer > Generate new token





11. Now we need to create variable groups
Go to Pipeline side tab > Library > New Variable Group



New variable group

Create groups of variables that you can share across multiple pipelines.

[+ Variable group](#)

[Learn more about variable groups.](#)

12. Add the group name and variable names and values

Library > DEV*

Variable group | Save | Clone | Security | Pipeline permissions | Approvals and checks | Help

Properties

Variable group name

Description

☐ Link secrets from an Azure key vault as variables ⓘ

Variables

Name ↑	Value	
databricksHost	https://adb-3690888067779850,10.azuredatabricks.net	
databricksToken	dapi2493219f94a43094cfce0cfcd0a47ee1-3	

13. The name DEV is from what we defined in our yaml script

14. You can also turn them Mask /invisible

Variables

Name ↑	Value	
databricksHost	*****	
databricksToken	dapi2493219f94a43094cfce0cfcd0a47ee1-3	

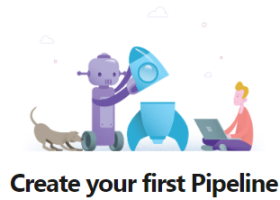
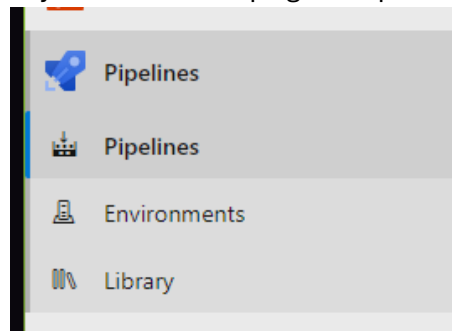
15. Save

Library > DEV*

Variable group | Save | Clo

16. Using the Variable Group we created earlier we will be creating pipeline following these steps

17. In your Azure DevOps go to Pipeline side tab

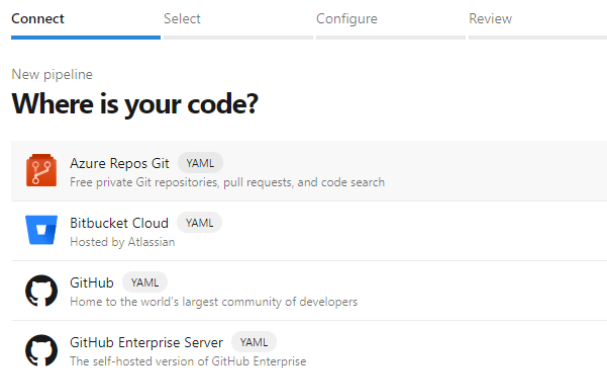


Create your first Pipeline

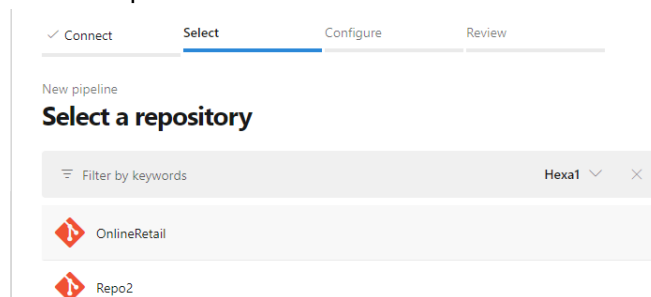
Automate your build and release processes using our wizard, and go from code to cloud-hosted within minutes.

[Create Pipeline](#) :

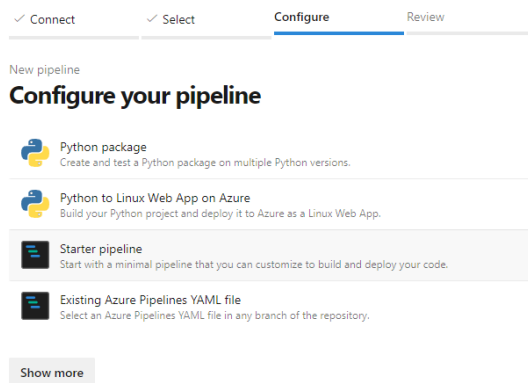
18. Choose azure repo



19. Select Repo



20. Starter Pipeline



21. In the YAML script write this

```
pool:
  vmImage: ubuntu-latest

variables:
- group: 'DEV'
- name: branchName
  value: $(Build.SourceBranch)
- name: FolderName
  value: release

steps:

# Step 1: Checkout Repository with Persisted Credentials
- checkout: self
  displayName: 'Checkout Repository'
  fetchDepth: 0
  persistCredentials: true # Ensures authentication token is used
  for git commands

# Step 2: Set Git Identity for Commit
- script: |
  git config --global user.email "ci-pipeline@yourdomain.com"
  git config --global user.name "CI Pipeline"
  displayName: 'Set Git Identity'

# Step 3: Install Databricks CLI
- script: |
  pip install databricks-cli
  displayName: "Install Databricks CLI"

# Step 4: Configure Databricks CLI
- script: |
  echo "${databricksHost}
  ${databricksToken}" | databricks configure --token
  displayName: 'Configure Databricks CLI'
```

```
# Step 5: Test Databricks CLI Connection
- script: |
    databricks workspace ls
  displayName: 'Test Databricks CLI Connection'

# Step 6: Fetch All Branches and Checkout Prod Branch
- script: |
    git fetch --all
    git checkout prod || git checkout -b prod
  displayName: 'Fetch and Checkout Prod Branch'

# Step 7: Merge Main into Prod
- script: |
    git merge origin/main --no-ff --commit -m "Merge changes from
main branch"
  displayName: 'Merge Main into Prod'

# Step 8: Push Changes to Prod Branch
- script: |
    git push origin prod
  displayName: 'Push Changes to Prod Branch'

# Step 9: Publish Updated Prod Branch Files as Pipeline Artifact
- task: PublishPipelineArtifact@1
  inputs:
    targetPath: '$(Build.Repository.LocalPath)/'
    artifact: 'Databricks'
    publishLocation: 'pipeline'
  displayName: 'Publish Prod Branch Files as Artifact'

# Step 10: Download Pipeline Artifact
- task: DownloadPipelineArtifact@2
  inputs:
    source: current
    artifact: 'Databricks'
    downloadPath: $(System.ArtifactsDirectory)/databricks
  displayName: 'Download Prod Files Artifact'

# Step 11: List Downloaded Files
- script: |
    ls $(System.ArtifactsDirectory)/databricks
  displayName: 'List Downloaded Artifacts'

# Step 12: Delete Old Release Folder in Databricks
- script: |
    FOLDER=$(FolderName)
    echo "Folder to delete: $FOLDER"
    databricks workspace rm $FOLDER --recursive
  displayName: 'Delete Old Release Folder'

# Step 13: Deploy New Release to Databricks
```

```
- script: |
  FOLDER=$(FolderName)
  echo "Folder for new release: $FOLDER"
  databricks workspace import_dir
$(System.ArtifactsDirectory)/databricks $FOLDER --exclude-hidden-
files
displayName: 'Deploy New Release to Databricks'
```

22. Now Save and Run after Reviewing your YAML

Variables Save and run

Show assistant

Save and run

Saving will commit azure-pipelines-1.yml to the repository.

Commit message

Set up CI with Azure Pipelines2

Optional extended description

Add an optional description...

☒ Commit directly to the main branch

☐ Create a new branch for this commit

23. Permit if asked

Waiting for review

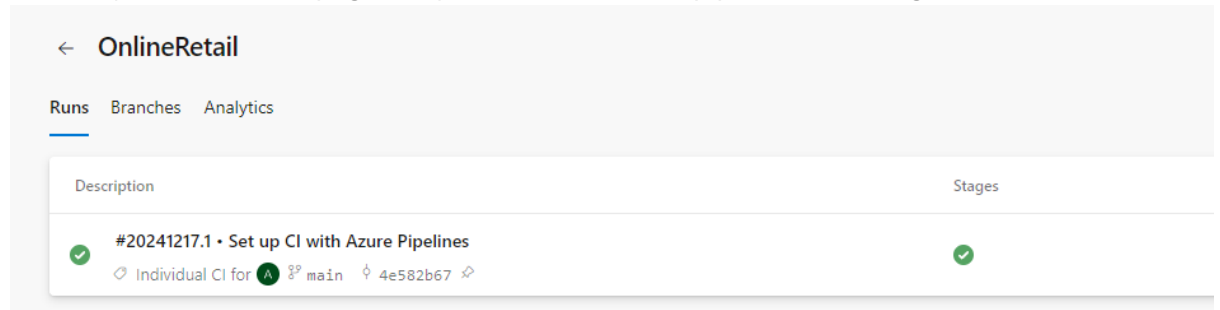
Permission Variable group DEV

Permission needed

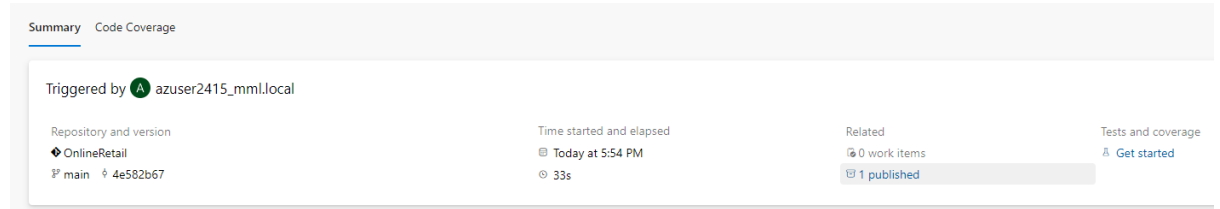
Permit

24. Now you have a pipeline to pull and merge changes from main branch to production branch whenever any changes occur in main branch and it also delivers the latest code from main branch to the release folder in our workspace

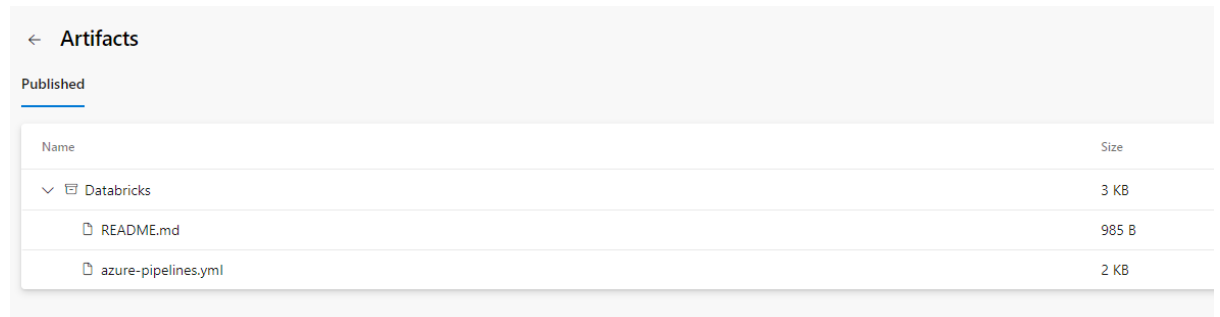
25. Refresh your databricks page and you will see that the pipeline is running



26. Now you can see it says 1 published

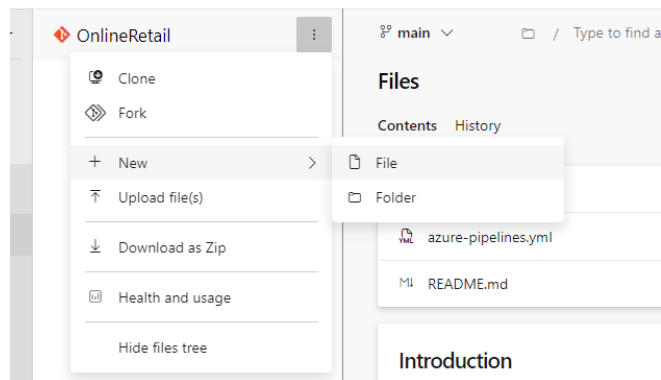


27. Click on it and see the artifacts it created



28. It has created an artifact for all items in repo even the YAML file which we do not want

So in our repo we can create .artifactignore file



New file

New file name

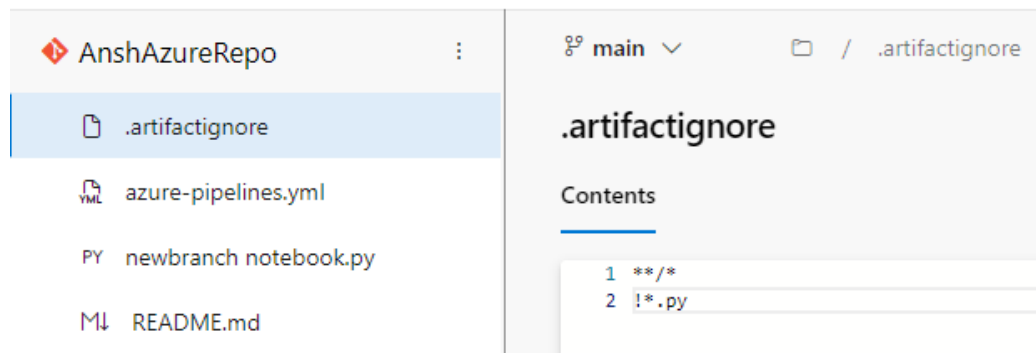
Use slashes to create multiple subfolders like "sub/folder".

Cancel Create

29. In the .artifactignore file write

****/***

!*.py



****/*** means to take everything in repo

!*.py means ignore the ones which are .py files

Essentially meaning add all files of directory which are not .py files into the .artifactignore

30. Commit

Commit ×

Comment

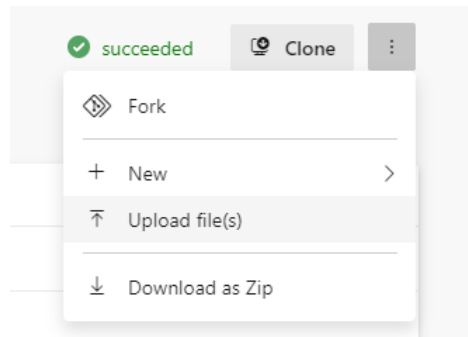
Branch name

Work items to link

3. Save and run the pipeline to verify its functionality.

As a test uploading a random notebook to main branch to test the working of pipeline

1. Uploading notebook to main branch



2. Committing changes

Commit

×

Drag and drop files here or click browse

Browse...

delta prac.ipynb

67.7 KB

remove

Comment

test

Branch name

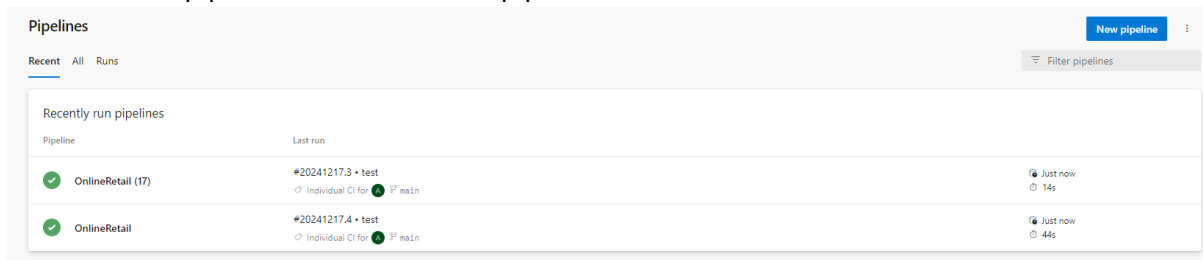
main

Work items to link

Search work items by ID or title

▼

3. Go to pipeline tab to check the pipelines

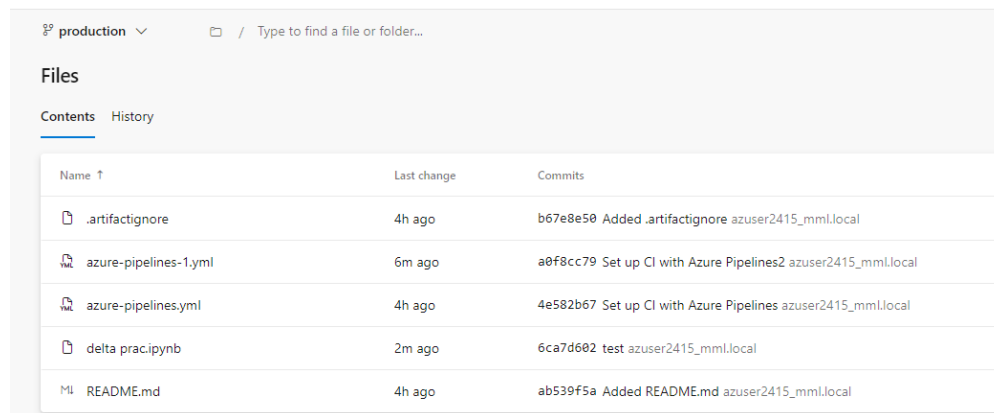


The screenshot shows the 'Pipelines' tab in the Azure DevOps interface. It lists two recently run pipelines, both of which are 'OnlineRetail'. The first pipeline is 'OnlineRetail (17)' and the second is 'OnlineRetail'. Both pipelines are marked with a green checkmark, indicating they ran successfully. The 'Last run' column shows the pipeline ID, branch, and commit. The 'Just now' status indicates the pipeline completed recently.

Pipeline	Last run
OnlineRetail (17)	#20241217.3 • test Individual CI for main
OnlineRetail	#20241217.4 • test Individual CI for main

Both have executed successfully

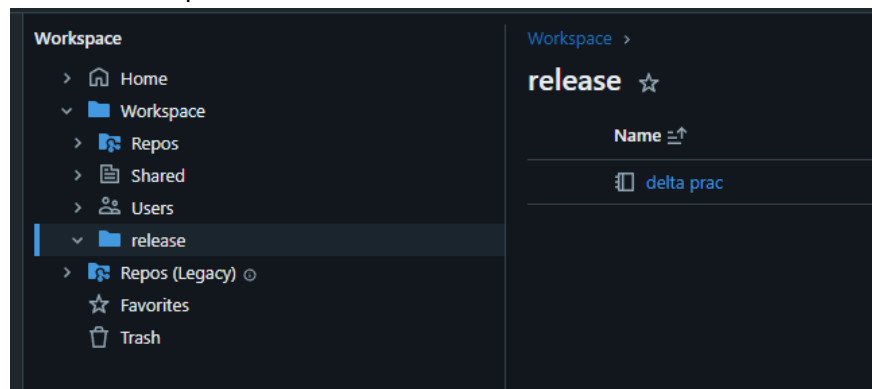
4. Check for changes to be reflected in production branch



The screenshot shows the 'Files' view in the Azure DevOps interface, specifically the 'production' branch. It lists several files and their commit history. The files are: .artifactignore, azure-pipelines-1.yml, azure-pipelines.yml, delta prac.ipynb, and README.md. The 'Last change' column shows the time since the last commit, and the 'Commits' column shows the commit hash and the user who made the change.

Name	Last change	Commits
.artifactignore	4h ago	b67e8e50 Added .artifactignore azuser2415_mml.local
azure-pipelines-1.yml	6m ago	a0f8cc79 Set up CI with Azure Pipelines2 azuser2415_mml.local
azure-pipelines.yml	4h ago	4e582b67 Set up CI with Azure Pipelines azuser2415_mml.local
delta prac.ipynb	2m ago	6ca7d602 test azuser2415_mml.local
README.md	4h ago	ab539f5a Added README.md azuser2415_mml.local

5. Check for updates to be reflected in the release folder



We have confirmed that the pipeline is running perfectly. We should go ahead and delete the test notebook we added.

Now our all our environment is setup and ready to perform tasks

=> Data Work:-

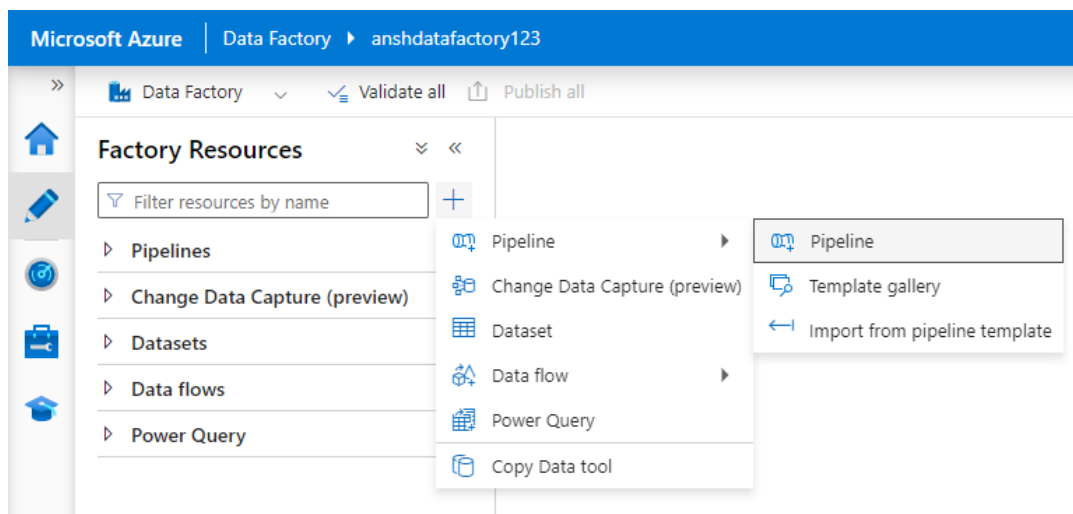
6. Bronze Pipeline (Raw Data Ingestion)

Purpose:

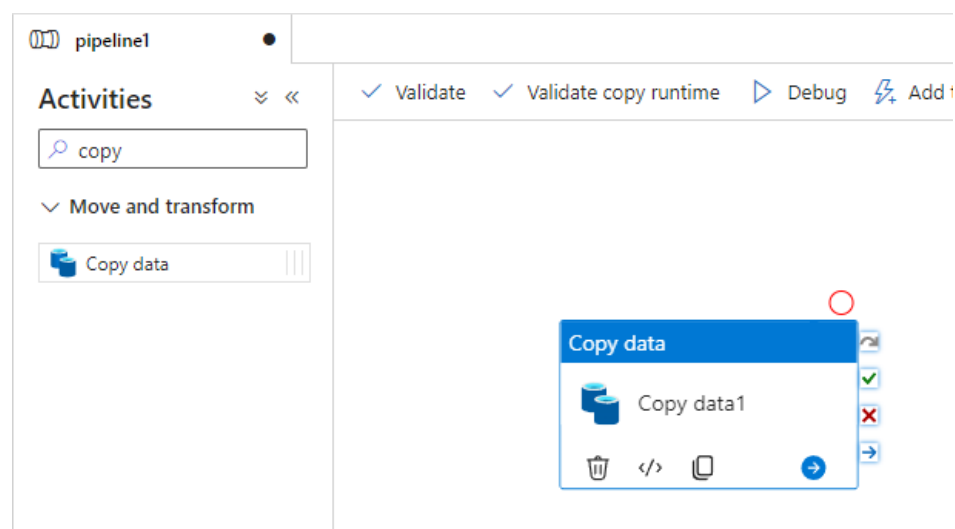
- The Bronze layer stores raw data with minimal processing, acting as the single source of truth.
- Automating this process ensures data consistency and repeatability.

Steps:

1. Open Azure Data Factory and create a new pipeline.



2. Use the Copy Data activity to transfer raw data from the rawdata container to the bronze container.



3. Configure the source as the raw data container and the sink as the bronze container.

General

Source¹

Sink¹

Mapping

Settings

User properties

Source dataset *


Select...

▼


+ New

4. Select Data Lake Storage Gen2 > DelimitedText (csv) which is our file type

New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#) 

Select a data store

 Search

<

All

Azure

Database

File

Generic protocol

>

Azure Blob Storage

Azure Cosmos DB for MongoDB

Azure Cosmos DB for NoSQL


Azure Data Explorer (Kusto)

Azure Data Lake Storage Gen2


Azure Database for MariaDB (Legacy)

Select format


Choose the format type of your data




Avro





Binary



DelimitedText







5. Now we need to create a new Linked Service

TECHADEMY LEARNING SOLUTIONS PRIVATE LIMIT

Set properties

Name

DelimitedText1

Linked service *


Select...

Filter...

+ New

6. Select your azure subscription and storage account name and test connection on bottom right. Then click on create

New linked service

 Azure Data Lake Storage Gen2 [Learn more](#)

Name *

AzureDataLakeStorage1

Description

Connect via integration runtime * ⓘ

 AutoResolveIntegrationRuntime

Authentication type

Account key

Account selection method ⓘ

☒ From Azure subscription ☐ Enter manually

Azure subscription ⓘ

MML Learners (2a3c6418-97b9-4d96-a24b-2c2d7633d375)

Storage account name *

anshstorageacc1

Test connection ⓘ


☒ To linked service ☐ To file path


Annotations

+ New

> Parameters

> Advanced ⓘ

 Connection successful

 Test connection

Create


Cancel

7. Next you need to browse for your file

Browse

Select a file or folder.

[Root folder](#) > **rawdata**

 Online Retail.xlsx

Set properties

Name

DelimitedText1

Linked service *

AzureDataLakeStorage1

File path

rawdata

/ Directory

/ Online Retail.xlsx

First row as header



Import schema

☒ From connection/store ☐ From sample file ☐ None

> Advanced

8. Now your source is done we need to select sink Go to the sink tab > New > Data Lake Storage Gen2

General Source **Sink¹** Mapping Settings User properties

Sink dataset *

Select...

+ New

New dataset

In pipeline activities and data flows, reference a dataset to specify the location and structure of your data within a data store. [Learn more](#)

Select a data store

All Azure Database File Generic protocol

Azure AI Search

Azure Blob Storage

Azure Cosmos DB for MongoDB

Azure Cosmos DB for NoSQL

Azure Data Explorer (Kusto)


Azure Data Lake Storage Gen2


9. We will not be converting our data to parquet files at this stage. It will be performed later in a silver layer. Currently our goal is to save the raw data in the Bronze Layer Container.


Select the DelimitedText option again.


Select format


Choose the format type of your data



Avro


Binary


DelimitedText


JSON


ORC


Parquet

10. Select the bronzelayer container as the sink path

Set properties

Name

DelimitedText2

Linked service *

AzureDataLakeStorage1

File path

bronzelayer / Directory / File name

First row as header

☒

Import schema

☒ From connection/store ☐ From sample file ☐ None

> Advanced

11. Test, Validate and publish the pipeline.



Your pipeline has been validated.

No errors were found.

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (3)

NAME	CHANGE	EXISTING
✓ Pipelines		
pipeline1	(New)	-
✓ Datasets		
DelimitedText1	(New)	-
DelimitedText2	(New)	-

12. Trigger the pipeline

✓ Validate ✓ Validate copy runtime ▶ Debug ⚡ Add trigger

Trigger now

New/Edit

13. You can check the pipeline in the Monitor side tab

Pipeline runs

Triggered Debug Rerun Cancel options Refresh Edit columns List Gantt

Filter by run ID or name Chennai, Kolkata, Mu... : Last 24 hours Pipeline name : All Status : All Runs : Latest runs Triggered by : All

Showing 1 - 1 items

<input type="checkbox"/>	Pipeline name ↑↓	Run start ↑↓	Run end ↑↓	Duration	Triggered by	Status ↑↓	Run
<input type="checkbox"/>	pipeline1	12/17/2024, 10:07:17 PM	12/17/2024, 10:07:39 PM	22s	Manual trigger	✓ Succeeded	Original

14. Check your bronzelayer container. The raw files are uploaded there

anshstorageeacc1 < + Add Directory ↑ Upload ↻ Refresh

★ Favorites

> ⚙ Recently viewed

▼ 📁 Blob containers

- 📁 \$logs
- 📁 **bronzelayer**
- 📁 goldlayer
- 📁 rawdata
- 📁 silverlayer

📁 Blob containers > 📁 bronzelayer

Authentication method: Access key ([Switch to Mic](#))

🔍 Search blobs by prefix (case-sensitive)

Showing all 1 items

<input type="checkbox"/>	Name
<input type="checkbox"/>	Online Retail.csv

7. Silver Layer (Data Cleaning)

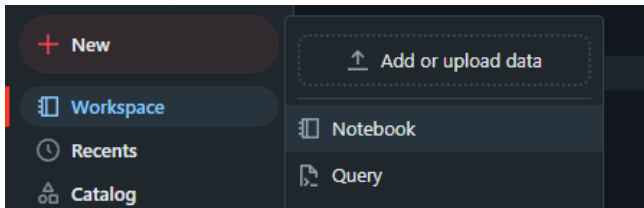
Purpose:

- The Silver layer provides cleaned and enriched data, enabling downstream processes to work with consistent datasets.

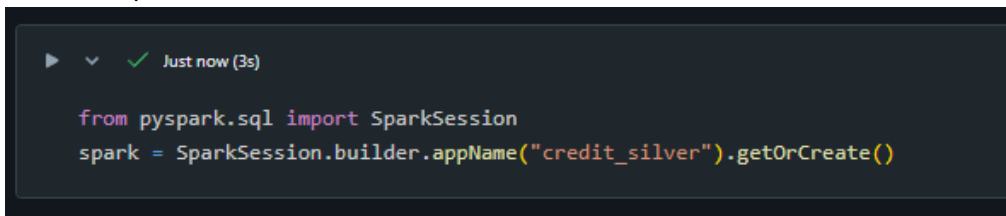
Steps:

Now that we have our containers ready, we need an Azure Databricks Workspace for ETL transformations

1. Launch the Azure Databricks workspace we created earlier
2. Go to your repo folder and switch to development branch
3. In the development branch click +New > Notebook



1. Create a spark session



EXTRACTION:

2. Mount your bronzelayer container to your workspace
define your source
`source="wasbs://{container_name}@{storage_account}.blob.core.windows.net/",`
Mounting point
`mount_point="/path "`
Configurations
`extra_configs={"fs.azure.account.key.{storage_account}.blob.core.windows.net":"{access_key_of_storage_account}"}`

To grab the access key go to your storage account > Security + Networking > Access Keys

anshstorageacc1 | Access keys

Storage account

Search

Set rotation reminder Refresh Give feedback

Overview
Activity log
Tags
Diagnose and solve problems
Access Control (IAM)
Data migration
Events
Storage browser
Storage Mover
Partner solutions
Data storage
Security + networking
Networking
Front Door and CDN
Access keys
Shared access signature

Access keys authenticate your applications' requests to this storage account. Keep your keys in a secure location Key Vault, and replace them often with new keys. The two keys allow you to replace one while still using the other.

Remember to update the keys with any Azure resources and apps that use this storage account.
[Learn more about managing storage account access keys](#)

Storage account name
anshstorageacc1

key1 Rotate key
Last rotated: 12/13/2024 (0 days ago)
Key
..... Show
Connection string
..... Show

key2 Rotate key
Last rotated: 12/13/2024 (0 days ago)
Key
..... Show
Connection string

Rotate the key once to make sure it is fresh and click on show in key 1 to copy

key1 Rotate key
Last rotated: 12/13/2024 (0 days ago)
Key
siatbyEfrZ14tOZQZDwrv1y6Ddn08e9z9yXN7AcjQN3CNfCZs9lov6DaX+sY6aovHj... Copy to clipboard Hide
Connection string

```
dbutils.fs.mount(  
  source="wasbs://bronzelayer@anshstorageacc1.blob.core.windows.net/",  
  mount_point="/mnt/bronze",  
  extra_configs={f"fs.azure.account.key.anshstorageacc1.blob.core.windows.net": "yJYkxy/4lM16tOdG895ttNjLhXoQ4ljG9n4iajm4ayIRTX8hY+hFB4pTAXYirbgAD7DXMSZ5pRCH+AST"
```

3. Read the data from the container

```
df=spark.read.csv('dbfs:/mnt/bronze/Online Retail.csv', header=True, inferSchema=True)
```

(2) Spark Jobs

Table +

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
1	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	01-12-2010 08:26	2.55	17850	United Kingdom
2	536365	71053	WHITE METAL LANTERN	6	01-12-2010 08:26	3.39	17850	United Kingdom
3	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	01-12-2010 08:26	2.75	17850	United Kingdom
4	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	01-12-2010 08:26	3.39	17850	United Kingdom
5	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	01-12-2010 08:26	3.39	17850	United Kingdom
6	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	01-12-2010 08:26	7.65	17850	United Kingdom
7	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	01-12-2010 08:26	4.25	17850	United Kingdom
8	536366	22633	HAND WARMER UNION JACK	6	01-12-2010 08:28	1.85	17850	United Kingdom
9	536366	22632	HAND WARMER RED POLKA DOT	6	01-12-2010 08:28	1.85	17850	United Kingdom
10	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	01-12-2010 08:34	1.69	13047	United Kingdom
11	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	01-12-2010 08:34	2.1	13047	United Kingdom
12	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	01-12-2010 08:34	2.1	13047	United Kingdom
13	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	01-12-2010 08:34	3.75	13047	United Kingdom
14	536367	22310	IVORY KNITTED MUG COSY	6	01-12-2010 08:34	1.65	13047	United Kingdom
15	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	01-12-2010 08:34	4.25	13047	United Kingdom

TRANSFORMATION:

4. Printing the schema of df

```
df.printSchema()

root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: string (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- Country: string (nullable = true)
```

The InvoiceDate column is of string data type

5. converting the InvoiceDate column into a date and time format

```
from pyspark.sql.functions import to_timestamp

# Convert InvoiceDate to a datetime column
df = df.withColumn("InvoiceDate", to_timestamp("InvoiceDate", "dd-MM-yyyy HH:mm"))

df.printSchema()
df.select("InvoiceDate").show(5, truncate=False)
```

```
root
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- Country: string (nullable = true)

+-----+
|InvoiceDate|
+-----+
|2010-12-01 08:26:00|
|2010-12-01 08:26:00|
|2010-12-01 08:26:00|
|2010-12-01 08:26:00|
|2010-12-01 08:26:00|
+-----+
only showing top 5 rows
```

6. Checking for number of rows with unique combinations of InvoiceNo and StockCode, taking the combination as a unique identifier

```
# Count rows with unique combinations of InvoiceNo and StockCode
unique_count = df.select("InvoiceNo", "StockCode").distinct().count()
print(f"Number of unique rows with distinct combinations of InvoiceNo and StockCode: {unique_count}")
```

▶ (3) Spark Jobs

Number of unique rows with distinct combinations of InvoiceNo and StockCode: 531225

7. Checking how many null values are present in each row

```
from pyspark.sql.functions import col, sum, when

null_counts = df1.select(
    [
        sum(when(col(c).isNull(), 1).otherwise(0)).alias(c)
        for c in df1.columns
    ]
)

null_counts.display()
```

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	0	1454	0	0	0	135080	0

8. Dropping any rows with NULL values

```
df = df.dropna()
```

▶ df: pyspark.sql.dataframe.DataFrame =

```
# Count rows with unique combinations of InvoiceNo and StockCode after dropping NULL values
unique_count = df.select("InvoiceNo", "StockCode").distinct().count()
print(f"Number of unique rows with distinct combinations of InvoiceNo and StockCode: {unique_count}")
```

▶ (3) Spark Jobs

Number of unique rows with distinct combinations of InvoiceNo and StockCode: 396681

9. **Filtering the dataset**

Removing rows where Quantity is 0 or less

```
from pyspark.sql.functions import col

# Removing rows where Quantity is 0 or less
df_filtered = df.filter(col("Quantity") > 0)
```

10. Removing rows where UnitPrice is 0 or less

```
# Removing rows where UnitPrice is 0 or less
df_filtered = df_filtered.filter(col("UnitPrice") > 0)
```

11. Filter out rows where Quantity is not an integer (or is a decimal value)

```
# Filter out rows where Quantity is not an integer (or is a decimal value)
df_filtered = df_filtered.filter(col("Quantity") % 1 == 0)
```

12. Since this dataset stores each item of an invoice in a separate row. Keeping only 1 record for each order of a customer at a given datetime, so we can later find which customers have reordered in the future

```
df_unique = df_filtered.dropDuplicates(["CustomerID", "InvoiceDate"])
df_unique.orderBy("InvoiceNo").show()
```

(2) Spark Jobs

df_unique, pyspark.sql.dataframe.DataFrame = [InvoiceNo: string, StockCode: string ... 6 more fields]

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
536365	85123A	WHITE HANGING HEA...	6	2010-12-01 08:26:00	2.55	17850	United Kingdom
536366	22633	HAND WARMER UNION...	6	2010-12-01 08:28:00	1.85	17850	United Kingdom
536367	84879	ASSORTED COLOUR B...	32	2010-12-01 08:34:00	1.69	13047	United Kingdom
536369	21756	BATH BUILDING BLO...	3	2010-12-01 08:35:00	5.95	13047	United Kingdom
536370	22728	ALARM CLOCK BAKEL...	24	2010-12-01 08:45:00	3.75	12583	France
536371	22086	PAPER CHAIN KIT 5...	80	2010-12-01 09:00:00	2.55	13748	United Kingdom
536372	22632	HAND WARMER RED P...	6	2010-12-01 09:01:00	1.85	17850	United Kingdom
536373	85123A	WHITE HANGING HEA...	6	2010-12-01 09:02:00	2.55	17850	United Kingdom
536374	21258	VICTORIAN SEWING	22	2010-12-01 09:09:00	10.95	15100	United Kingdom

13. Getting a count of number of unique orders for each customer

```
from pyspark.sql.functions import count

# Group by CustomerID and count the occurrences
df_customer_multiple = df_unique.groupBy("CustomerID") \
    .agg(count("CustomerID").alias("customer_count"))
```

```
df_customer_multiple.show(10)
```

▶ (3) Spark Jobs

CustomerID	customer_count
15727	7
17389	34
16503	4
14570	2
17420	3
13285	4
13623	5
15447	1
16386	2
14450	3

only showing top 10 rows

14. Now we can join this df to our df_filtered, this results in a new column which shows how many times a customer placed an order in our original df

```
# Perform a left join between df_filtered and df_customer_multiple on CustomerID
df_filtered_with_flag = df_filtered.join(df_customer_multiple, on="CustomerID", how="left")

df_filtered_with_flag.show()
```

▶ (4) Spark Jobs

CustomerID	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	customer_count
17850	536365	85123A	WHITE HANGING HEA...	6	2010-12-01 08:26:00	2.55	United Kingdom	33
17850	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	United Kingdom	33
17850	536365	84406B	CREAM CUPID HEART...	8	2010-12-01 08:26:00	2.75	United Kingdom	33
17850	536365	84029G	KNITTED UNION FLA...	6	2010-12-01 08:26:00	3.39	United Kingdom	33
17850	536365	84029E	RED WOOLLY HOTTIE...	6	2010-12-01 08:26:00	3.39	United Kingdom	33
17850	536365	22752	SET 7 BABUSHKA NE...	2	2010-12-01 08:26:00	7.65	United Kingdom	33
17850	536365	21730	GLASS STAR FROSTE...	6	2010-12-01 08:26:00	4.25	United Kingdom	33
17850	536366	22633	HAND WARMER UNION...	6	2010-12-01 08:28:00	1.85	United Kingdom	33
17850	536366	22632	HAND WARMER RED P...	6	2010-12-01 08:28:00	1.85	United Kingdom	33
13047	536367	84879	ASSORTED COLOUR B...	32	2010-12-01 08:34:00	1.69	United Kingdom	9
13047	536367	22745	POPPY'S PLAYHOUSE...	6	2010-12-01 08:34:00	2.1	United Kingdom	9
13047	536367	22748	POPPY'S PLAYHOUSE...	6	2010-12-01 08:34:00	2.1	United Kingdom	9
13047	536367	22749	FELTCRAFT PRINCES...	8	2010-12-01 08:34:00	3.75	United Kingdom	9
13047	536367	22310	IVORY KNITTED MUG...	6	2010-12-01 08:34:00	1.65	United Kingdom	9
13047	536367	84969	BOX OF 6 ASSORTED...	6	2010-12-01 08:34:00	4.25	United Kingdom	9
13047	536367	22623	BOX OF VINTAGE JI...	3	2010-12-01 08:34:00	4.95	United Kingdom	9

15. Making a new column Flag_reorder where the value is 1 if the customer has reordered or else 0 if the customer never reordered from the store

```
from pyspark.sql.functions import when

# Assuming customer_count is a column in df_filtered_with_flag
df_filtered_with_flag = df_filtered_with_flag.withColumn(
    "Flag_reorder", when(col("customer_count") == 1, 0).otherwise(1)
)

df_filtered_with_flag.show()
```

LOAD:

16. Saving the data into silverlayer container

Mounting the container

```
#Saving this data in our silver layer container
#mounting
dbutils.fs.mount(
    source="wasbs://silverlayer@anshstorageeacc1.blob.core.windows.net/",
    mount_point="/mnt/silver",
    extra_configs={f"fs.azure.account.key.anshstorageeacc1.blob.core.windows.net": "yJYkxy/4lM16todG895ttNjLhXoQ41jG9n4iajm4ay1RTX8hY+hFB4pTAXY1rbgAD7DXHSZ5pRCH+ASTqvkNgw=="})

True
```

17. Writing the data in form of parquet files

```
1 df_filtered_with_flag.write.format("parquet").mode("overwrite").save("/mnt/silver/online_retail_parquet")
```

The screenshot displays the Azure Storage Explorer interface. On the left sidebar, the 'Blob containers' section is expanded, showing a list of containers: \$logs, bronzelayer, goldlayer, rawdata, and silverlayer. The 'silverlayer' container is selected and highlighted. Below the list, there is a 'View all' link. The main pane on the right shows the contents of the 'silverlayer' container. At the top, there are controls for '+ Add Directory', 'Upload', 'Refresh', and 'Delete'. Below these, the 'Authentication method' is set to 'Access key' with a link to 'Switch to Microsoft Entra u'. A search bar is present with the placeholder text 'Search blobs by prefix (case-sensitive)'. Below the search bar, it says 'Showing all 2 items'. A table lists the items in the container:

<input type="checkbox"/>	Name
<input type="checkbox"/>	_\$azuretmpfolder\$
<input type="checkbox"/>	online_retail_parquet

8. Gold Layer (Data Aggregation)

Purpose:

- The Gold layer stores business-ready data for analytics and reporting.

Steps:

Gold Layer (Data Enrichment and Aggregation) In the Gold layer, the data is enriched with business-level aggregations and summaries.

Now create a new notebook for Gold Layer

1. Create your spark session

```
from pyspark.sql import SparkSession
spark = SparkSession.builder.appName("credit_gold").getOrCreate()
```

EXTRACTION:

2. Mount your silverlayer container (if required)

```
1 %fs ls dbfs:/mnt/silver/
```

	path	name	size	modificationTime
1	dbfs:/mnt/silver/online_retail_parqu...	online_retail_parqu...	0	0

LOAD:

3. Load the data from your silverlayer container

```
1 # Read the Parquet file into a DataFrame
2 df = spark.read.format("parquet").load("dbfs:/mnt/silver/online_retail_parquet/")
```

```
1 df.show(10)
```

▶ (1) Spark Jobs

CustomerID	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	customer_count	Flag_reorder
18119	537567	90178A	AMBER CHUNKY GLAS...	1	2010-12-07 11:59:00	11.95	United Kingdom	1	0
18119	537567	21864	UNION JACK FLAG P...	1	2010-12-07 11:59:00	2.1	United Kingdom	1	0
18119	537567	21916	SET 12 RETRO WHIT...	1	2010-12-07 11:59:00	0.42	United Kingdom	1	0
18119	537567	21866	UNION JACK FLAG L...	1	2010-12-07 11:59:00	1.25	United Kingdom	1	0
18119	537567	21916	SET 12 RETRO WHIT...	1	2010-12-07 11:59:00	0.42	United Kingdom	1	0
18119	537567	20751	FUNKY WASHING UP ...	1	2010-12-07 11:59:00	2.1	United Kingdom	1	0
18119	537567	21327	SKULLS WRITING SET	1	2010-12-07 11:59:00	1.65	United Kingdom	1	0
18119	537567	21328	BALLOONS WRITING...	1	2010-12-07 11:59:00	1.65	United Kingdom	1	0
18119	537567	21865	PINK UNION JACK ...	1	2010-12-07 11:59:00	2.1	United Kingdom	1	0
18119	537567	21867	PINK UNION JACK ...	1	2010-12-07 11:59:00	1.25	United Kingdom	1	0

TRANSFORMATION:

4. Performing Aggregations:

Since each item of invoices are listed as a separate row we need to group the rows by InvoiceID

```
from pyspark.sql.functions import first

df_grouped = df.groupBy("InvoiceNo").agg(
    first("CustomerID").alias("CustomerID"),
    first("Country").alias("Country"),
    first("customer_count").alias("customer_count"),
    first("Flag_reorder").alias("Flag_reorder")
)

df_grouped.show()
```

InvoiceNo	CustomerID	Country	customer_count	Flag_reorder
541783	16618	United Kingdom	5	1
544303	18116	United Kingdom	8	1
537691	13842	United Kingdom	9	1
541518	12451	Switzerland	5	1
550831	13534	United Kingdom	22	1
550617	13376	United Kingdom	3	1
562204	16324	United Kingdom	3	1
554792	12547	Spain	2	1
553390	17841	United Kingdom	124	1
552191	16832	United Kingdom	3	1
554072	13555	United Kingdom	4	1
553658	16818	United Kingdom	11	1

5. Now we also need a df with only data of individual customers

```
df_customer_only = df.groupBy("CustomerID").agg(
    first("Country").alias("Country"),
    first("customer_count").alias("customer_count"),
    first("Flag_reorder").alias("Flag_reorder")
)

df_customer_only.show()
```

CustomerID	Country	customer_count	Flag_reorder
18119	United Kingdom	1	0
13211	United Kingdom	3	1
14560	United Kingdom	22	1
16686	United Kingdom	7	1
12455	Cyprus	6	1
18138	United Kingdom	1	0
16940	United Kingdom	3	1
18210	United Kingdom	6	1

6. Finding out customer retention rate

```
1 from pyspark.sql.functions import col
2
3 #Customers who have placed only 1 order
4 customers_one = df.filter(col("Flag_reorder") == 0).count()
5 print(f"Number of customers with only 1 order placed: {customers_one}")
6
7 #Customers who have placed multiple orders
8 customers_multi = df.filter(col("Flag_reorder") == 1).count()
9 print(f"Number of customers with multiple orders placed: {customers_multi}")
```

```
Number of customers with only 1 order placed: 1495
Number of customers with multiple orders placed: 2843
```

Counting the number of customer with only 1 order and number of customers with multiple orders placed

7. Customer Retention Rate:

The customer retention rate measures the percentage of customers a business retains over a specific period. It indicates how good the business is at keeping its customers returning for future purchases or interactions.

$\text{Retention_rate} = (\text{Customers with multiple orders} / \text{Customers with single order}) * 100$

```
1 #Percentage of returning customers
2 percentage = (customers_multi / (customers_one + customers_multi)) * 100
3 print(f"Customer Retention rate : {percentage:.2f}%")
```

```
Customer Retention rate : 65.54%
```

The Customer Retention Rate for the store in the past 1 year time period is 65.54%

8. Customer Churn Rate

Customer churn rate, also known as customer attrition rate, is a metric that measures the percentage of customers who stop using a company's product or service over a given period of time. It is commonly used by businesses to evaluate customer retention and satisfaction.

The formula for calculating the churn rate is:

$\text{Churn Rate} = (\text{Number of customers lost} / \text{Total Number of customers}) * 100$

```
1 churn_rate = (customers_one / (customers_one + customers_multi)) * 100
2 print(f"The Churn Rate for the store is : {churn_rate:.2f}%")
```

```
The Churn Rate for the store is : 34.46%
```

The Churn Rate for the store in the past 1 year time period is 34.46%

LOAD:

9. Mounting the goldlayer storage

```
1 #Saving this data in our gold layer container
2 #mounting
3 dbutils.fs.mount(
4     source="wasbs://goldlayer@anshstorageacc1.blob.core.windows.net/",
5     mount_point="/mnt/gold",
6     extra_configs={f"fs.azure.account.key.anshstorageacc1.blob.core.windows.net": "b02ueNu8Q5JAFB2nDoa/PKCR7W5HsJHEAoxWP4ns0B1zZ/gpKYv3UGp10RgAtZojcW3Z1m4dn3eK+ASttqwPAA=="})
7 )
True
```

10. Saving all 3 data frames so that the which ever df is required can be used by analytics

```
1 df.write.format("parquet").mode("overwrite").save("/mnt/gold/all_data_parquet")
2 df_grouped.write.format("parquet").mode("overwrite").save("/mnt/gold/invoices_grouped_parquet")
3 df_customer_only.write.format("parquet").mode("overwrite").save("/mnt/gold/customer_only_parquet")
```

The screenshot shows the Azure Storage Explorer interface. On the left, the 'Blob containers' section is expanded, and 'goldlayer' is selected. The main pane shows the contents of the 'goldlayer' container. At the top, there are buttons for '+ Add Directory', 'Upload', and 'Refresh'. Below these, the 'Authentication method' is set to 'Access key'. A search bar is present with the text 'Search blobs by prefix (case-sensitive)'. Below the search bar, it says 'Showing all 4 items'. A table lists the items in the container:

<input type="checkbox"/>	Name
<input type="checkbox"/>	_azuretmpfolder\$
<input type="checkbox"/>	all_data_parquet
<input type="checkbox"/>	customer_only_parquet
<input type="checkbox"/>	invoices_grouped_parquet

9. Logistic Regression Model

Purpose:

- Predicts customer churn likelihood, aiding in strategic decision-making.

Steps:

Creating a logistic regression model to predict if a customer will be returning customer or a one time customer only

1. Creating spark session

```
03:53 PM (<1s)

1 from pyspark.sql import SparkSession
2 spark = SparkSession.builder.appName("logistic_regression").getOrCreate()
```

2. Loading the data

Mount the container if required, here since we already have the container mounted in our databricks workspace we can directly load

```
1 %fs ls dbfs:/mnt/gold
```

	path	name	size	modificationTime
1	dbfs:/mnt/gold/all_data_parquet/	all_data_parquet/	0	0
2	dbfs:/mnt/gold/customer_only_parquet/	customer_only_parquet/	0	0
3	dbfs:/mnt/gold/invoices_grouped_parquet/	invoices_grouped_parquet/	0	0

```
1 # Read the Parquet file into a DataFrame
2 data = spark.read.format("parquet").load("dbfs:/mnt/gold/all_data_parquet/")

(1) Spark Jobs
```

```
1 data.show(5)
```

CustomerID	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	customer_count	Flag_reorder
18119	537567	90178A	AMBER CHUNKY GLAS...	1	2010-12-07 11:59:00	11.95	United Kingdom	1	0
18119	537567	21864	UNION JACK FLAG P...	1	2010-12-07 11:59:00	2.1	United Kingdom	1	0
18119	537567	21916	SET 12 RETRO WHIT...	1	2010-12-07 11:59:00	0.42	United Kingdom	1	0
18119	537567	21866	UNION JACK FLAG L...	1	2010-12-07 11:59:00	1.25	United Kingdom	1	0
18119	537567	21916	SET 12 RETRO WHIT...	1	2010-12-07 11:59:00	0.42	United Kingdom	1	0

3. Getting schema

```
1 data.printSchema()

root
 |-- CustomerID: integer (nullable = true)
 |-- InvoiceNo: string (nullable = true)
 |-- StockCode: string (nullable = true)
 |-- Description: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- InvoiceDate: timestamp (nullable = true)
 |-- UnitPrice: double (nullable = true)
 |-- Country: string (nullable = true)
 |-- customer_count: long (nullable = true)
 |-- Flag_reorder: integer (nullable = true)
```

4. Dropping the columns "InvoiceNo", "Description", "InvoiceDate", "customer_count" as we will not be using them as features for Logistic Regression model

```
1 # Drop the unnecessary columns from the DataFrame
2 data = data.drop("InvoiceNo", "Description", "InvoiceDate", "customer_count")
3 data.show()
```

CustomerID	StockCode	Quantity	UnitPrice	Country	Flag_reorder
18119	90178A	1	11.95	United Kingdom	0
18119	21864	1	2.1	United Kingdom	0
18119	21916	1	0.42	United Kingdom	0
18119	21866	1	1.25	United Kingdom	0
18119	21916	1	0.42	United Kingdom	0
18119	20751	1	2.1	United Kingdom	0
18119	21327	1	1.65	United Kingdom	0
18119	21328	1	1.65	United Kingdom	0
18119	21865	1	2.1	United Kingdom	0

5. Getting a count for the number of distinct values in country

```
1 count = data.select("Country").distinct().count()
2 print(count)

▶ (3) Spark Jobs

37
```

Since there are 37 unique countries performing onehotencoder will massively increase the complexity of our dataset. So we will only be performing Indexing

6. Performing Indexing for the columns Country and StockCode as they are string values. We need to pass integer or numeric values for Vectorization

```

indexer_country = StringIndexer(inputCol="Country", outputCol="CountryIndex", handleInvalid="skip")
indexer_stockcode = StringIndexer(inputCol="StockCode", outputCol="StockCodeIndex", handleInvalid="skip")

assembler=VectorAssembler(inputCols=[
    'CustomerID',
    'Quantity',
    'UnitPrice',
    'CountryIndex',
    'StockCodeIndex'],outputCol='features')

```

```

from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline

```

7.

```

log_reg=LogisticRegression(featuresCol='features',labelCol='Flag_reorder')

```

8. Creating a pipeline to ensure the transformations to data happens in the correct order in a sequential manner

```

pipeline=Pipeline(stages=[indexer_country,indexer_stockcode,assembler,log_reg])

```

9. Splitting our data into train and test dataset in the ratio of 70% : 30%

```

1  train,test=data.randomSplit([0.7,.3])
• test: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, StockCode: string ... 4 more fields]
• train: pyspark.sql.dataframe.DataFrame = [CustomerID: integer, StockCode: string ... 4 more fields]

```

10. Training the model on train dataset

```

model=pipeline.fit(train)

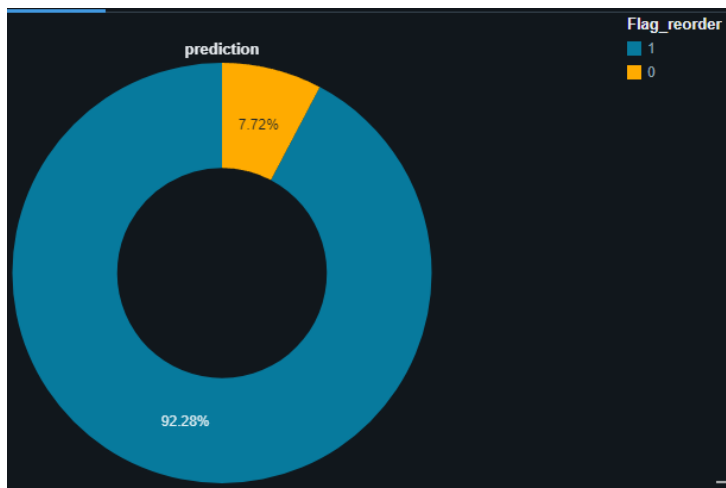
```

11. Performing a test of model on the test dataset

```

result=model.transform(test)
display(result.select('Flag_reorder','prediction'))

```



12. Getting the accuracy for the model

```
from pyspark.ml.evaluation import BinaryClassificationEvaluator  
eval_re=BinaryClassificationEvaluator(rawPredictionCol='prediction',labelCol='Flag_reorder')
```

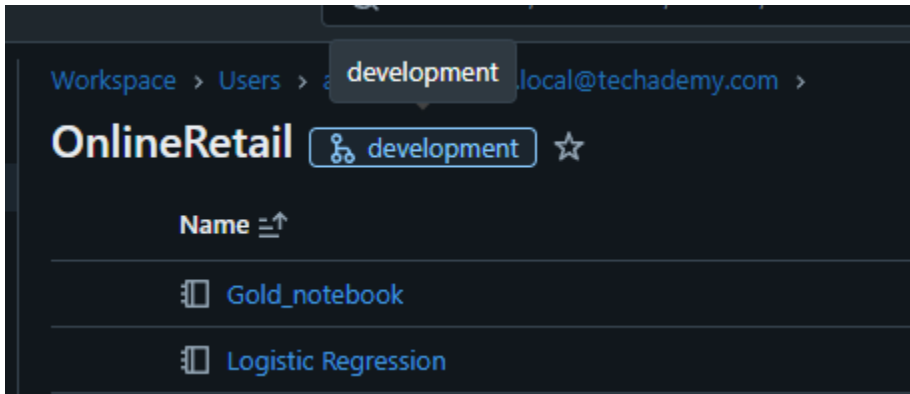
```
1 AUC=eval_re.evaluate(result)  
2 AUC
```

► (4) Spark Jobs

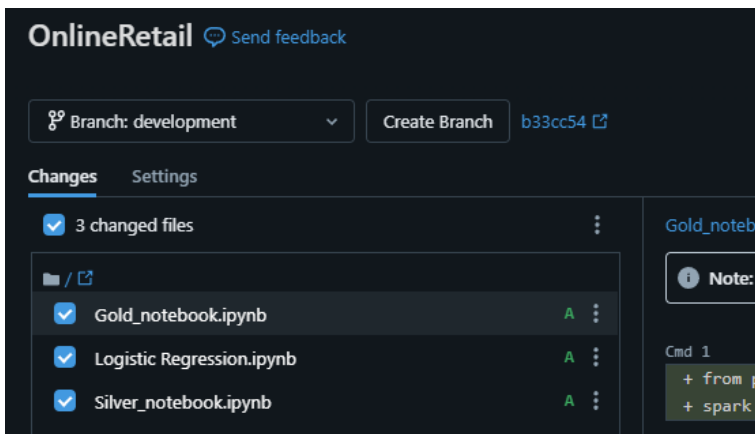
0.5

10. Pushing the changes into the Development branch

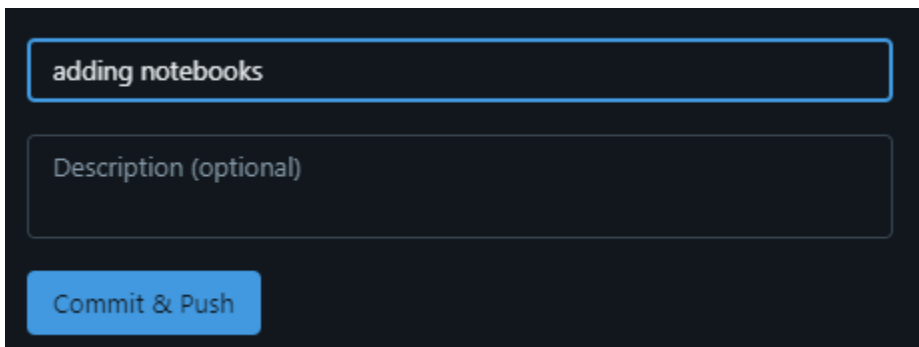
1. You should be in the development branch if our repo in your databricks workspace
2. Click on the branch icon



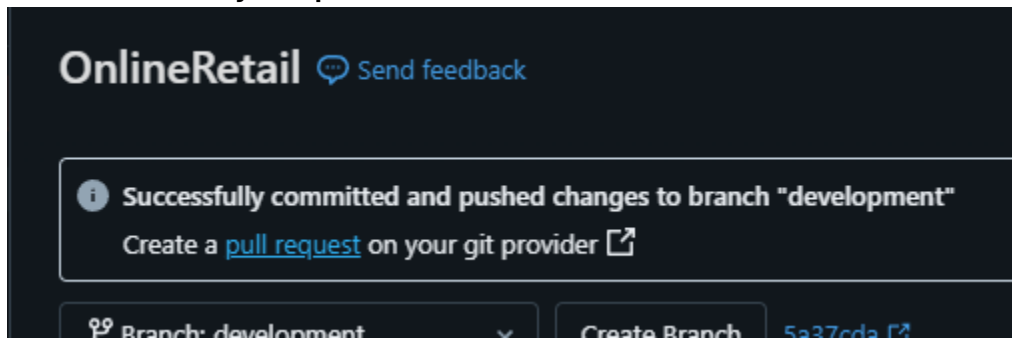
3. You can see that it notices 3 new notebook additions



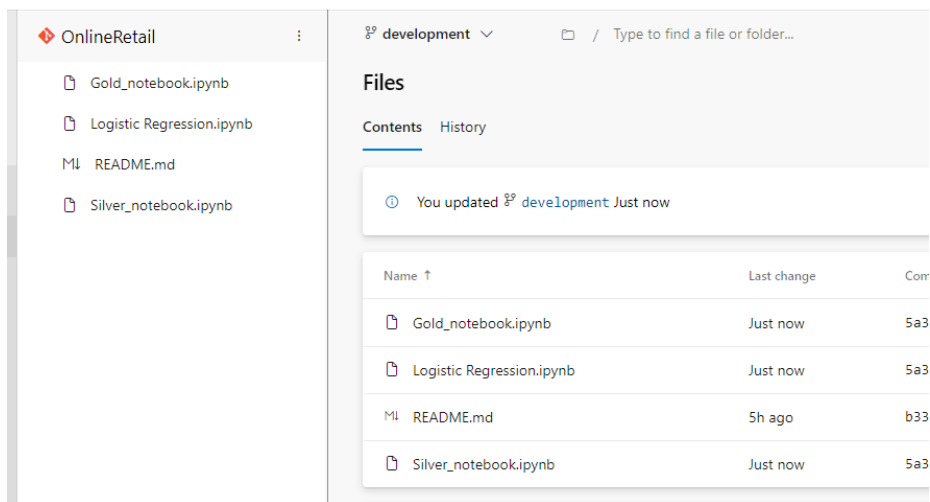
4. Select the Commit & Push button



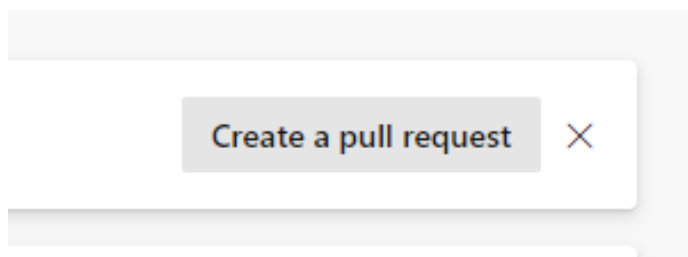
5. Push successfully completed



6. You can see the changes in the repo by selecting the development branch in Azure Git repo



7. You will not see an option to create a pull request



8. Create a pull request from development to main branch

New pull request

development into main

Overview Files 3 Commits 1

Title

adding notebooks

Description

adding notebooks

16/4000

Markdown supported. Drag & drop, paste, or select files to insert.

Link work items.

@ # 🔗 📎 ✎ ✓ **B** *I* </> ↔ ☰ ☷ ☸

adding notebooks

Reviewers

Add required reviewers

👤 Search users and groups to add as reviewers

Work items to link

Search work items by ID or title

Tags

Create

9. Review and Complete the pull request

adding notebooks

Active 16 👤 azuser2415_mmllocal proposes to merge development into main

Approve Complete

Overview Files Updates Commits

✓ No merge conflicts

Last checked just now

Reviewers

Add

Required


10. Do not check the delete development branch at the moment and complete the merge

Complete pull request

×

Merge type

Merge (no fast forward) ▾



Post-completion options

☒ Complete associated work items after merging

☐ Delete development after merging






☐ Customize merge commit message

11. Now your main branch should also contain the notebooks we created earlier

🔗 main ▾ / Type to find a file or folder...

Files

Contents History




Name ↑	Last change	Commits
 .artifactignore	5h ago	ef217b47 Added .artifactignore azuser2415_mml.local
 Gold_notebook.ipynb	3m ago	5a37cdab adding notebooks azuser2415_mml.local@techademy.com
 Logistic Regression.ipynb	3m ago	5a37cdab adding notebooks azuser2415_mml.local@techademy.com
 README.md	5h ago	b33cc54c Added README.md azuser2415_mml.local
 Silver_notebook.ipynb	3m ago	5a37cdab adding notebooks azuser2415_mml.local@techademy.com

12. Your pipeline should have been automatically triggered as well

Pipelines

Recent All Runs

Recently run pipelines

Pipeline	Last run
 OnlineRetail	#20241218.2 • Merge changes from main branch  Individual CI for  production

13. The pipeline changes can be observed in both the production branch

production / Type to find a file or folder...

Files

Contents History

Name ↑	Last change	Commits
azure-pipelines.yml	16m ago	fd47b5d
Gold_notebook.ipynb	26m ago	a95c27a
Logistic Regression.ipynb	26m ago	a95c27a
README.md	40m ago	e5ede48
Silver_notebook.ipynb	26m ago	a95c27a

as well as the release folder in our repo clone

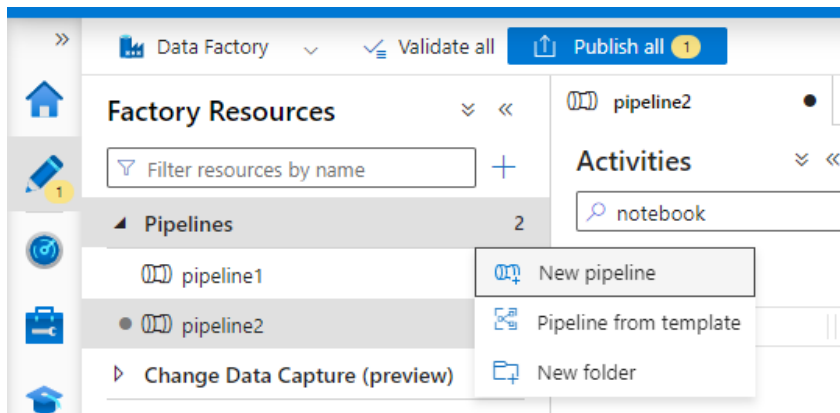
Workspace >

release ☆

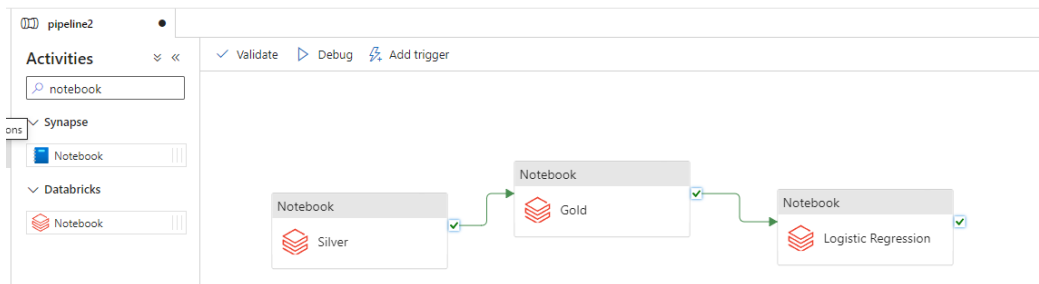
Name ↕	Type	Owner
Gold_notebook	Notebook	azuser2415_mml.local
Logistic Regression	Notebook	azuser2415_mml.local
Silver_notebook	Notebook	azuser2415_mml.local

11. Creating a pipeline for automating notebooks

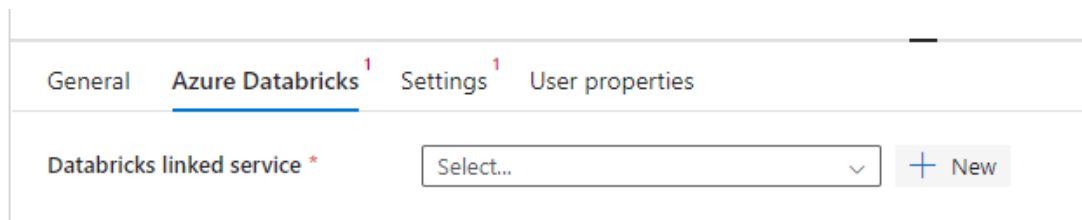
1. Go to your Azure DataFactory > Author > Pipeline > New



2. From activities drag 3 notebook activities, one for each notebook and link them on condition "on success"





3. In Silver Notebook's Azure Databricks tab, create a new linked service



4. **Select your azure subscription and workspace**
choose Existing interactive cluster and grab an access token like we did before
After entering the access token select the cluster from drop down menu

New linked service

 Azure Databricks [Learn more](#) 


New linked service

Name *

AzureDatabricks1

Description

Connect via integration runtime * ⓘ

 AutoResolveIntegrationRuntime

▼

Account selection method *


☒ From Azure subscription

☐ Enter manually

Azure subscription * ⓘ

MML Learners (2a3c6418-97b9-4d96-a24b-2c2d7633d375) ▼

Databricks workspace * ⓘ

Databricks123 ▼ 

Select cluster

☐ New job cluster

☒ Existing interactive cluster

☐ Existing instance pool

Databrick Workspace URL * ⓘ

https://adb-2945023889648544.4.azure.databricks.net

Authentication type *

Access Token ▼

Access token

Azure Key Vault

Access token *

.....

Choose from existing clusters * ⓘ

azuser2415_mml.local's Cluster ▼

Annotations

+ New

▼

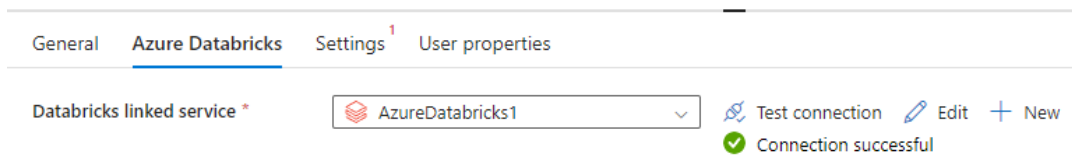
Create

Cancel

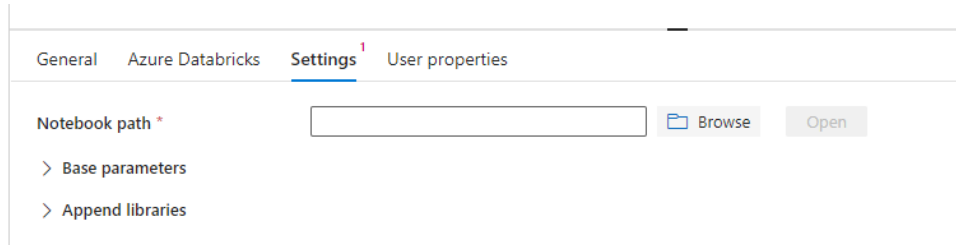
 Connection successful

 Test connection

5. Then click on create



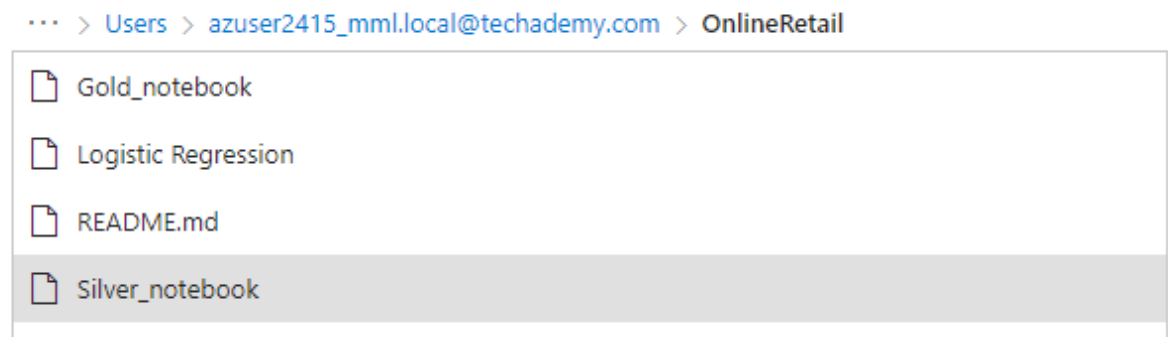
6. Next in the Settings tab browse for notebook path



7. Browse and select your silver notebook

Browse

Select a file or folder.



8. Follow the same steps for gold notebook and logistic regression notebook as well

9. Next Validate and Publish the pipeline



Your pipeline has been validated.

No errors were found.

Publish all

You are about to publish all pending changes to the live environment. [Learn more](#)

Pending changes (1)

NAME	CHANGE	EXISTING
▼ Pipelines		
🔗 pipeline2	(New)	-

10. Trigger the pipeline

✓ Validate ▶ Debug ⚡ Add trigger

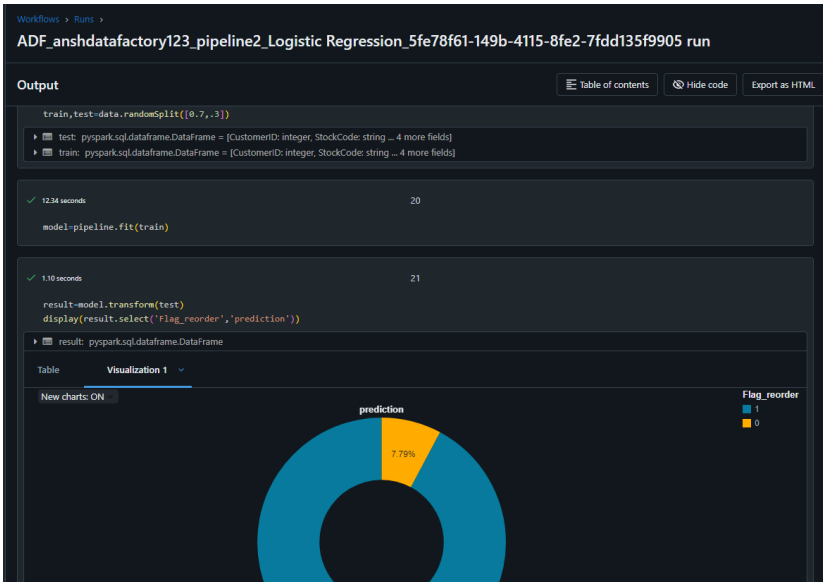
Trigger now

Trigger on-demand run of the last published pipeline

New/Edit

11. You can view the execution by going to databricks > Job Runs side tab

General		17 DEC 12 AM		17 DEC 12 PM		18 DEC 12 AM	
Alerts		Failed	Skipped	Succeeded			
Query History		Start time	Job	Run as	Launched	Duration	Status
SQL Warehouses		Dec 18, 2024, 06:30 PM	ADF_anshdatafactory123_pipel...	azuser2415_mmllocal	By runs submit API	2m 17s	Running
Data Engineering		Dec 18, 2024, 06:26 PM	ADF_anshdatafactory123_pipel...	azuser2415_mmllocal	By runs submit API	3m 27s	Succeeded
Job Runs							
Data Ingestion							
Delta Live Tables							



12. We have successfully completed execution of all 3 notebooks

Start time	Job	Run as	Launched	Duration	Status
Dec 18, 2024, 06:32 PM	ADF_anshdatafactory123_pipel...	azuser2415_mml.local	By runs submit API	47s	✔ Succeeded
Dec 18, 2024, 06:30 PM	ADF_anshdatafactory123_pipel...	azuser2415_mml.local	By runs submit API	2m 23s	✔ Succeeded
Dec 18, 2024, 06:26 PM	ADF_anshdatafactory123_pipel...	azuser2415_mml.local	By runs submit API	3m 27s	✔ Succeeded

Showing 1 - 2 items

<input type="checkbox"/> Pipeline name ↑↓	Run start ↑↓	Run end ↑↓	Duration	Triggered by	Status ↑↓
<input type="checkbox"/> > pipeline2	12/18/2024, 6:26:27 PM	12/18/2024, 6:33:59 PM	7m 33s	Manual trigger	✔ Succeeded
<input type="checkbox"/> pipeline1	12/18/2024, 12:25:04 PM	12/18/2024, 12:25:23 PM	19s	Manual trigger	✔ Succeeded

12. Results

1. Bronze Layer (Raw Data Ingestion):

- Successfully ingested raw transactional data from Online Retail.xlsx into Azure Data Lake using Data Factory.
- Verified raw data consistency and ensured schema alignment for further transformations.

2. Silver Layer (Data Cleaning & Transformation):

- Cleaned the dataset by handling missing values, removing duplicates, and filtering invalid transactions (e.g., negative quantities).
- Sessionized customer interactions to better understand user behavior and interaction duration.

3. Gold Layer (Aggregation & Metrics):

- Created aggregated customer activity metrics, such as total purchases, average spending, and frequency of transactions.
- Generated business-ready insights, including customer segmentation and purchasing trends.

4. Predictive Model (Churn Prediction):

- Implemented a logistic regression model in Databricks to predict the likelihood of customer churn.
- Achieved high accuracy and interpretability, identifying key features influencing customer retention.

5. CI/CD Integration:

- Successfully integrated the project with Azure Git for version control.
- Automated deployment processes with CI/CD workflows, ensuring seamless updates to the production environment.

13. Conclusion

- 1.** The project provides a scalable and efficient ETL workflow for customer behavior analysis using the Medallion architecture (Bronze, Silver, Gold layers).
- 2.** Key business insights, such as customer purchase trends and churn likelihood, empower data-driven decision-making for improving customer retention.
- 3.** The use of Azure Data Factory and Databricks ensures seamless orchestration and powerful transformations, enabling efficient data processing and predictive analytics.
- 4.** By incorporating CI/CD pipelines, the workflow achieves operational reliability and streamlined deployment across environments.
- 5.** The methodology can be extended to similar retail datasets, offering a robust framework for customer analytics in other industries.