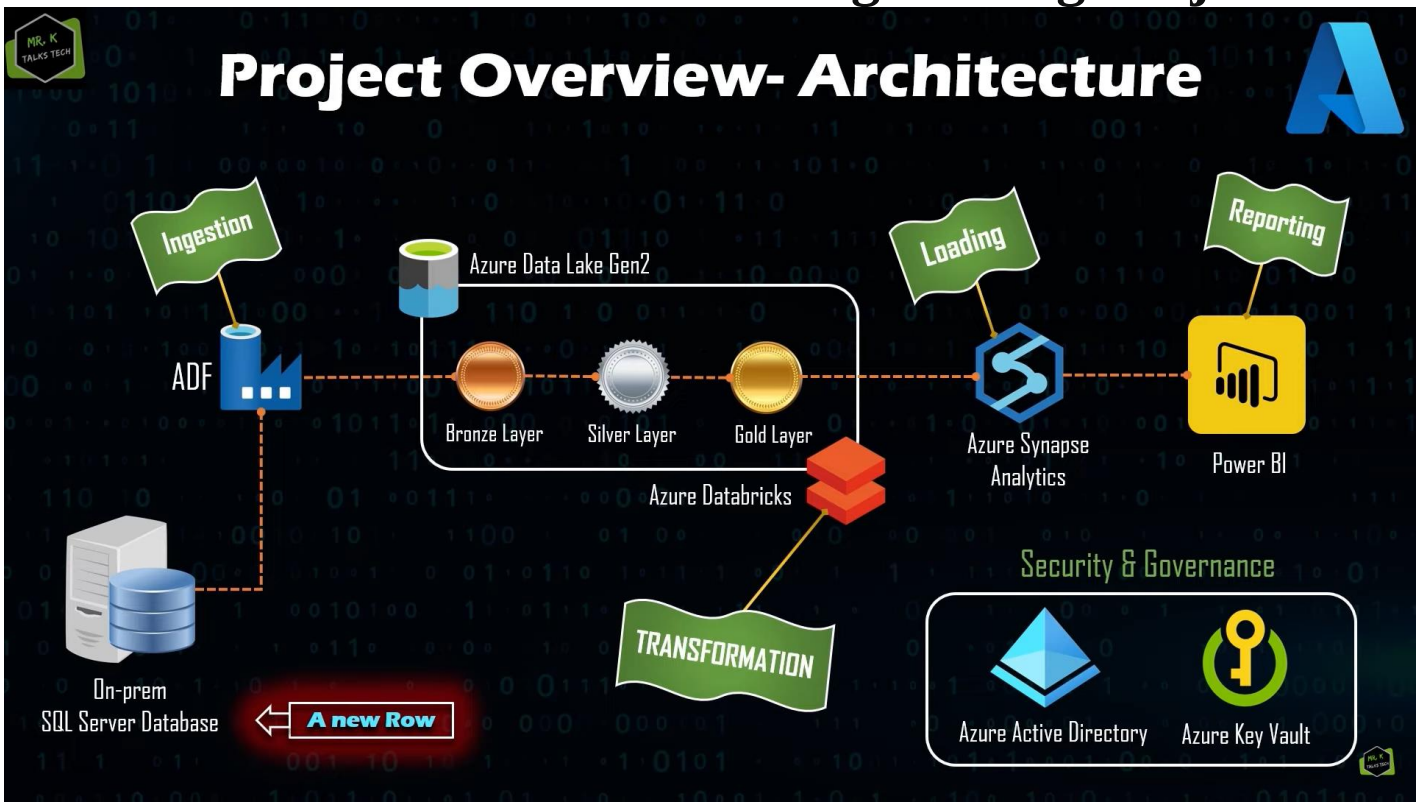
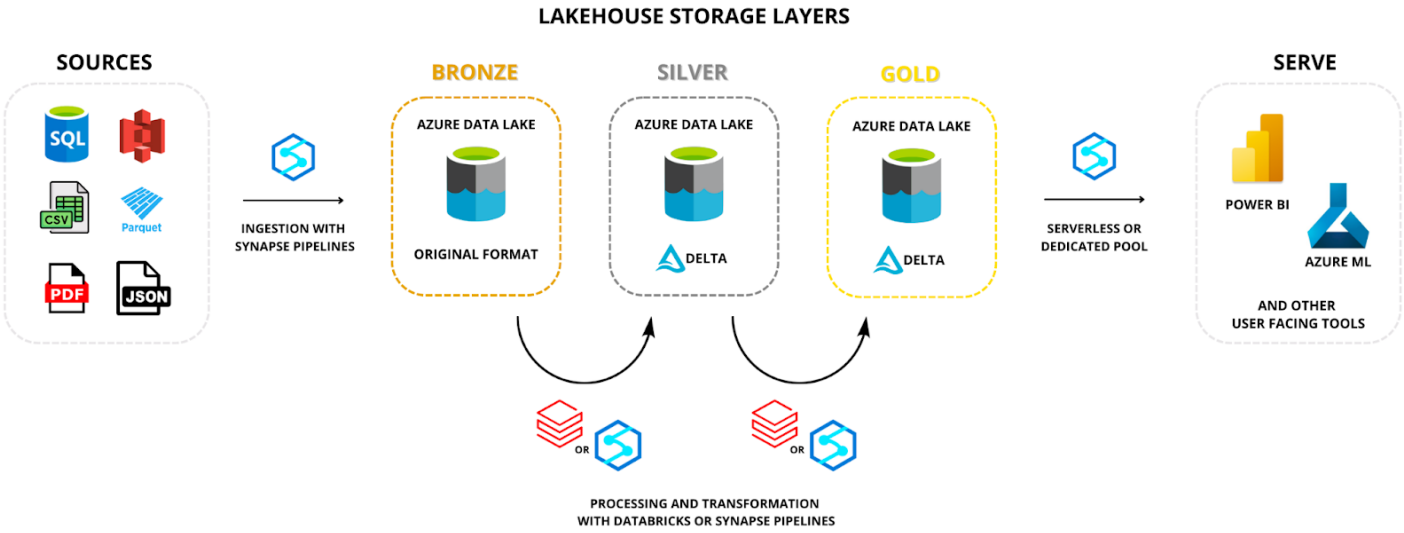
Azure Data Engineering Road Map Project

1. AZURE CLOUD
2. SQL SERVER MANAGEMENT STUDIO(SQL)
3. AZURE KEY VAULT (Security and Credentials)
4. ADLS GEN2(Object Storage)
5. ADF(Ingest/Orchestration)
6. PYTHON (API PySpark Framework)
7. AZURE SQL DB(SQL)
8. AZURE DATABRICKS(ETL)
9. AZURE SYNAPSE-ANALYTICS (Data Warehouse)
10. POWER Bl (Reporting Tool)
11. AZURE LOGIC APPS (Send Notifications)
12. AZURE ACTIVE DIRECTORY(Roles)
13. AZURE COSMO-DB(No-SQL)
14. AZURE STREAMING ANALYTICS(IOT)
15. POWER PLATFORMS (No Code Platform)
16. MICROSOFT FABRIC (SAAS Platform)

Part-1 End to End Azure Data Engineering Project





Part 2- Environment setup

Step 1:

Sql server--->file--->import

```
use database_name
go

create login rg with password = 'radharam203';
create user rg for login rg
```

goto--->security--->user--->rg--->right click--->properties--->membership--->db_datareader

Step 2:

goto resource group--->azure key vault--->secrets--->no secrets here.

Step 3:

Azure--->key-vault--->generate/import--->create a secret--->1. for user
2. for password

#These secrets are encrypted.

Part 3- Data Ingestion

Part-I

Step1: goto Azure Data Factory--->Integration Runtime.

Self-Hosted Integration Runtime (onprime data source).

go to manage tap--->integration runtime--->click new--->Azure, Self-Hosted--->self-hosted--->continue.

two setups

1. Manual
2. Express (go with)
   Download and install
   Search Microsoft integration runtime

Go to on-prem system--->search (Microsoft Integration Runtime)--->open--->connected to the cloud service.

Step 2:

Go to Author tab--->pipeline--->new pipeline--->copy data activity (source)

new dataset--->sql server--->set properties--->linked services--->new--->integration runtime(add Created IR)--->server name (local host) --->database name--->authentication type(sql authentication)--->user name(RG)--->password(azure key vault)

AKV linked service--->Name--->Azure key vault selection method(from azure subscription)--->azure key-vault-name(add key-vault-name)--->authentication mode(system assigned managed identity)--->test connection--->forbidden error(for reading permission).

#To solve this problem

Go to key vault--->access policies--->create--->goto secret permissions--->select all--->principal--->adf-mrk-demo(ADF Name)--->next--->review and create access policy--->publish.

Goto ADF--->secret name(refresh)--->password--->secret version (latest version)--->test connection--->Create.

Set properties--->table name (choose any table)--->click ok.

Preview data(source)

Step 3:

Sink (bronze layer) --->new--->new dataset--->ADLS Gen2--->parquet format--->set properties--->linked service--->name (ADLS1)--->connect via integration runtime (auto resolve Integration runtime)--->authentication type (account key)--->azure subscription--->storage account name(name)--->continue set properties--->file path--->browse--->add link (goto resource--->container--->browse).

Run the pipeline two ways 1. Debug 2. Trigger

Goto container--->bronze--->delete parquet file which created with pipeline

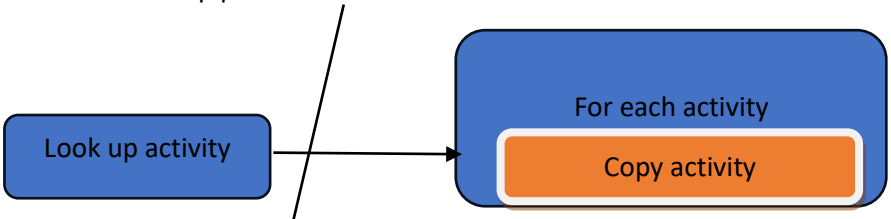
Part-II

Copy multiple tables in bulk by using Azure Data Factory

```
SELECT *
FROM
Database_name.INFORMATION_SCHEMA.TABLES
WHERE table_type= 'BASE TABLE' and TABLE_Schema = 'dbo'
```

#Instead of copy one single table, copy multiple tables at time.

Create new pipeline--->name



#All the tables at time once move on-Prem to ADLS gen2

Goto SQL server--->open SQL file--->script

```
select
s.name as SchemaName,
t.name as TableName
from sys.table t
inner join sys.schemas as s
on t.schema_id= s.schema_id
where s.name = 'SalesLT'
```

step 1:

#lookup activity

Add lookup Activity--->settings--->new dataset--->sql-server--->name--->linked service (On-Prem)--->table_name (leave as it is)

go to--->settings--->use query (paste sql script) --->preview data--->debug--->output

step 2:

#foreach activity

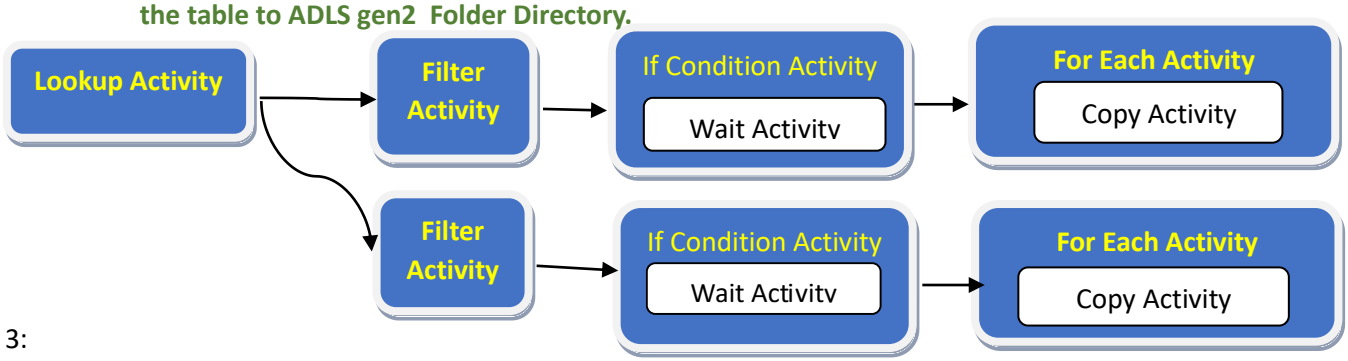
Add for each activity--->settings--->items (add dynamic content--->pipeline expression builder (activity outputs--->look for all tables)--->@activity ('look for all tables').output.value)--->inside for each activity (Activities--->click pencil icon--->drag copy data activity))

#copy activity

source--->new dataset--->sql server--->linked service(on-prime)--->continue ok--->goto--->settings--->use query--->add dynamic content--->Pipeline Expression Builder(@{concat('select * from', item().SchemaName, '.', item().TableName)})

sink --->new dataset --->ADLS gen2--->parquet--->set properties--->name--->linked service--->ADLS gen2.

#Note: Instead of copy all the tables only choose selected columns use "Filter Activity" and pass dynamic Parameter and use "IF Condition Activity" and inside "Wait Activity" pass dynamic parameters pull the table to ADLS gen2 Folder Directory.



step 3:

#Folder Structure

#to create Folder Structure in datalake important you should notice **Schema_Name** and **Table_Name**

Bronze/Schema/Tablename/Tablename.parquet

#Address table

Bronze/SalesLT/Address/Address.parquet

#goto again copy activity

Sink--->sink dataset--->open--->parameters--->new--->(name--->schema_name--->table_name)--->type(string).

Goto again copy activity--->sink { schemaname--->value (add dynamic content)

tablename--->value (add dynamic content)

@item.Schemaname

@item.TableName

Goto sink--->connection--->directory(add dynamic content)--->(pipeline expression builder--->@{concat(dataset().schemaname, '/',dataset().tablename)})

#Concat method is concat schema and table

Filename---> paste it @{concat(dataset ().tablename, '.parquet')}

#Everything is configured

Click publishall

Add trigger---> trigger now (once runs)--->click ok

Goto monitor--->pipeline runs

Goto data lake--->bronze container--->SalesLT--->all tables you can see

Part 4 - Data Transformation Part-I

#goto workspace--->click on Azure databricks--->Launch databricks workspace

1. New
2. Workspace
3. Repos (git repository integration)
4. Recent
5. Data(database)
6. Compute (spark cluster)
7. Workflows (create job inside notebook but here instead of we use ADF pipeline to trigger that create for the part of this project)

Create compute cluster (spark cluster) --->Name--->policy--->performance (Databricks runtime version (standard and ML))--->Node type--->Terminate after(15 minutes of inactivity).
Advanced options--->azure data lake storage credential passthrough(enable).

Open **ADLS Gen2**--->Access Control (IAM)--->Role assignments.
No connection between **Azure DataBricks** to **Azure Data Lake** in order to establish connection to mount the datalake difference ways to do it (service principle and credential passthrough method).

Create Cluster--->workspace--->shared--->right click on white space --->create notebook--->Name--->default language(python)--->click ok

#Azure Data Lake Storage Gen2

To mount an **ADLS Gen2** filesystem or a folder inside it, use the following commands to use credential passthrough method.

```
#mount bronze container

configs = {
        "fs.azure.account.auth.type": "CustomAccessToken",
        "fs.azure.account.custom.token.provider.class":spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}
#Optionally, you can add <directory-name> to the source URL of your mount point.
container-name1=bronze
storage-account-name1=datalakegen2

dbutils.fs.mount(
        source = "abfss://<container-name1>@<storage-account-name1>dfs.core.windows.net/",
        mount_point = "/mnt/<container-name1>",
        extra_configs = configs)

#Run cell(shift+enter)

dbutils.fs.ls("/mnt/bronze/SalesLT/")

#mount silver container
        #No data present inside silver mount
configs = {
        "fs.azure.account.auth.type": "CustomAccessToken",
        "fs.azure.account.custom.token.provider.class":
        spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}

dbutils.fs.mount(
        source = "abfss://silver@<storage-account-name1> dfs.core.windows.net/",
        mount_point = "/mnt/silver",
        extra_configs = configs)

#mount gold container

configs = {
        "fs.azure.account.auth.type": "CustomAccessToken",
        "fs.azure.account.custom.token.provider.class":spark.conf.get("spark.databricks.passthrough.adls.gen2.tokenProviderClassName")
}
# Optionally, you can add <directory-name> to the source URL of your mount point.
container-name2=gold
storage-account-name2=datalakegen2

dbutils.fs.mount(
        source = "abfss://<container-name2>@<storage-account-name2> dfs.core.windows.net/",
        mount_point = "/mnt/<container-name2>",
        extra_configs = configs)
```

Part-II

#Data already there in OLTP system

#Create two notebooks

1. **One is Bronze to silver**
2. **Another one is Silver to Gold**

#pyspark

```
dbutils.fs.ls('mnt/bronze/SalesLT')                                  #list of all tables
dbutils.fs.ls('mnt/silver')                                          #don't have any files
Input_path = '/mnt/bronze/SalesLT/Address/Address.parquet'           #one table
df=spark.read.format('parquet').load(input_path)                     #pyspark DataFrame
display(df)                                                          #display it works only databricks platform
```

#import

```
from Pyspark.sql.function import from_utc_timestamp, date_format
from Pyspark.sql.types import TimestampType
df1 = df.withColumn("ModificationDate", date_format(from_utc_timestamp(df["ModificationDate"]. Cast (Timestamp()), "UTC"), "yyyy-MM-dd")
display(df1)
```

one single notebook we can work any language using magic commands (Scala, R and SQL)

```
%sql
```

#markdown cell

```
%md
```

#Simple Example

```
Select 1 as column
```

#output is:

```
1
```

#doing transformation for all tables

```
Table_name = []
for i in dbutlis.fs.ls('mnt/bronze/SalesLT/'):
        Table_name.append(i.name.split('/')[0])

from Pyspark.sql.function import from_utc_timestamp, date_format
from Pyspark.sql.types import TimestampType
for i in table_name:
        path = '/mnt/bronze/SaleLT/' + i + '/' + i + '.parquet'
        df =sprk.read.format('parquet').load(path)
        column = df.columns

        for col in column:
                if "Date" in col or "date" in col:
                        df = df.withColumn(col, date_format(from_utc_timestamp(df[col].cast
                        (TimestampType()),"UTC"), "yyyy-MM-dd"))
        output_path = '/mnt/silver/SaleLT/' + i +'/'
        df.write.format('delta').mode("overwrite").save(output_path)

display(df)
```

#delta format is built on top parquet format it is developed by databricks, the delta formats features have all parquet format that already have.

#features of Delta format:

1. We can track the different version history.
2. Handle different schema changes

#The databricks recommends to use the Delta format to store the data into the data lake let's use delta format to write the transform data in both the silver and gold container.

goto container--->silver---> refresh

#Level 2 Transformation:
#Join and Aggregation in level Transformation basically final Fact and Dimension table in Datawarehouse.

```
dbutils.fs.ls('mnt/silver/SalesLT')
dbutils.fs.ls('mnt/gold')                                    #no. files in gold container
Input_path = '/mnt/silver/SalesLT/Address/'
df=spark.read.format('delta').load(input_path)
display(df)

from pyspark.sql import SparkSession
from pyspark.sql.functions import col, regexp_replace

column_names = df.columns                    #get the list of all columns
for old_col_name in column_names:
        # convert column name from ColumnName to Column_Name format
        new_col_name = "".join(["_" + char if char.isupper(0 and not old_col_name[i-1].isuper() else char for i, char
        in enumerate(old_col_name)]).lstrip("_")
#change the column name using withColumnRenamed and regexp_replace

df = df.withColumnRenamed(old_col_name, new_col_name)

display(df)

#Doing transformation for all tables (changing column names)
Table_name = []
for i in dbutils.fs.ls('mnt/silver/SalesLT/'):
        Table_name.append(i.name.split('/')[0])

for j in table_name:
        path = '/mnt/silver/SalesLT/' + j
        print(path)
df = spark.read.format('delta').load(path)
Column_names = df.columns                    #get the list of all columns
for old_col_name in column_names:
        # convert column name from ColumnName to Column_Name format
        new_col_name= " ".join(["_" + char if char.isupper(0 and not old_col_name[i-1].isupper() else char for i, char
        in enumerate(old_col_name)]).lstrip("_")
        #change the column name using withColumnRenamed and regexp_replace
        df = df.withColumnRenamed(old_col_name, new_col_name)

output_path = '/mnt/gold/SalesLT/' +name +'/'
df.write.format('delta').mode("overwrite").save(output_path)

display(df)
```

goto gold container--->refresh--->you can see all tables with data
#Storage amount notebook this notebook is just mounting the data lake right so we have to just run that once and we don't have to run that as a job but other two notebooks which is bronze to silver and silver to gold needs to run as a job.

Part-III

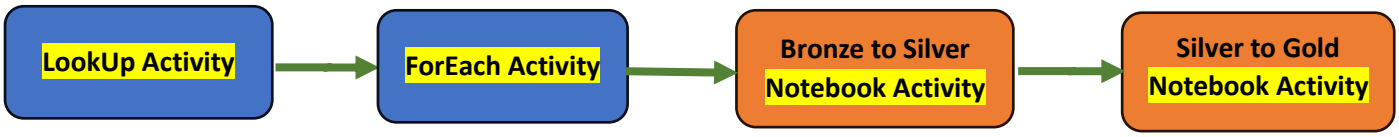
Goto workspace--->updates notebook
Goto manage--->linked services--->new--->compute--->Azure databricks--->name--->connect via integration runtime(autoresolveintegrationruntime)--->Azure Subscription Name--->databricks workspace--->cluster (existing interactive cluster)--->Authentication type(Access Token)--->Access Key Vault--->AKV linked service--->secret name--->choose from existing cluster--->test connection--->create.

Goto Databricks--->user settings--->generate new token--->copy.

Azure key vault--->secrets--->generate/import--->create a secret--->upload options--->name--->secret value(paste).

Publishall

 # Create new two Databricks Notebook Activities 1. Bronze to Silver 2. Silver to Gold
Goto author--->activity--->search notebook--->Databricks Activity ((bronze to silver)--->Azure Databricks (databricks linked service)--->select--->settings(notebook path)--->browse--->shared--->bronze to silver).



add another notebook (Silver to Gold)--->(Azure Databricks (databricks linked service)--->select--->settings(Notebook path)--->browse----->shared--->Silver to Gold).

publishall

#Test pipeline

add trigger--->trigger now--->click ok
refresh

#for incremental load "write.format('delta').mode("overwrite").save(output_path) bronze to silver which ignore all the data which is there already processed

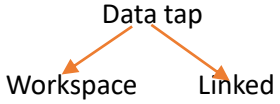
Part 5 - Data Loading (Azure Synapse Analytics)

#Create Synapse Workspace
#Azure Synapse built on top of ADF.
#Here Home, Data, Develop, Integrate, Monitor, Manage.
#Azure Synapse Analytics = ADF + Databricks

- SQL server less is only computing
- Dedicated pool is both computing and storage

Goto manage---> sql pools--->new

#in this go with sql serverless
Goto Data tap--->click + --->sql database--->serverless--->name



Data tap--->workspace--->create SQL database--->select SQL pool type(serverless)--->database(name)--->create SQL database
Goto Linked--->ADLS Gen2(Bronze, Silver, Gold)--->gold--->address(right click table--->new SQL script--->select top 100)--->File type(Parquet Format, Text Format, Delta Format).

#Select Delta Format

```
SELECT
        Top 100 *
FROM
Openrowset (
        Bulk 'https://mrkdatalakegen2.dfs.core.windows.net/gold/SalesLT/address/',
        Format = "DELTA"
) AS [result]
```

#Create view

```
USE  gold_db
GO
CREATE VIEW ADDRESS
AS
SELECT
        *
FROM
        Openrowset (
        Bulk 'https://mrkdatalakegen2.dfs.core.windows.net/gold/SalesLT/address/',
        FORMAT  = "DELTA"
) AS [result]
```

Goto workspace--->gold_db(sql)(refresh)--->views--->click--->new SQL script--->select TOP 100 rows

 # Create views all the available tables inside gold table instead of just typing create view each single table to we create a pipeline dynamically all the tables.

#Instead of create pipeline in Azure Data Factory we can create Azure Synapse Analytics.

goto Develop tap--->click +--->import (sql sp script which is already written).

#Create Stored Procedure

```
USE gold_db
GO
        CREATE or ALTER PROC createSQLserverlessview_gold @viewName nvarchar (100)
        AS
        BEGIN
        DECLARE @statement VARCHAR(MAX)
                SET @statement = N'CREATE or ALTER VIEW '+ @viewName +' AS
                SELECT  *
                FROM
                        Openrowset (
                        Bulk 'https://mrkdatalakegen2.dfs.core.windows.net/gold/SalesLT/address/' + @viewName, +
                                '/',
                        FORMAT = "DELTA"
                                ) AS [result]
        EXEC (@statement)
        END
        GO

        publishall
```

Goto manage tap--->linked service--->sql database--->connect via integrationruntime(autoresolveIR)

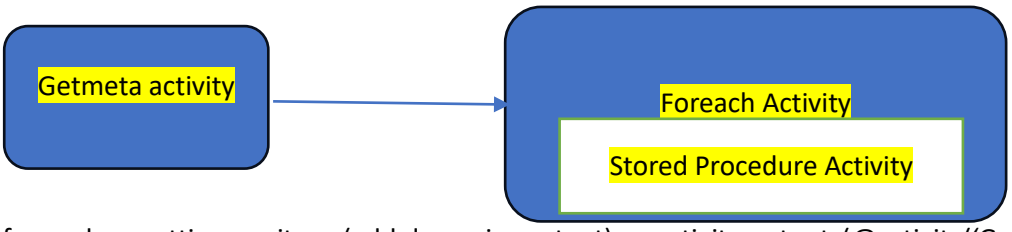
Account selection method (enter manually)--->fully qualified domain name--->database name(gold_db)--->Authentication type(system assigned managed identity)--->Test connection--->click create.

Publishall

(goto synapse workspace--->properties--->serverless sql endpoint--->copy)--->paste it.

#Create pipeline

goto integrate--->click +--->Create new pipeline--->Getmetadata Activity--->settings(new)--->ADLS Gen2--->binary--> setproperties--->linkedservice(synw-mrk-demo-01-WorkspaceDefaultStorage--->browse--->gold--->SaleLT--->click ok Getmetadata Activity Settings--->field list--->chid items--->debug--->refresh.



for each--->settings--->items(add dynamic content)--->activity outputs(@activity('Get Tablename').output.childitems)--->click ok--->goto edit option--->add stored procedure--->settings--->stored procedure name--->stored procedure parameters (new)--->name () --->type(string)--->value(add dynamic content)--->@item().name

goto develop--->sql scripts--->copy(viewName)
publishall

#Run the pipeline
run--->trigger now

goto monitor tap--->integration--->pipeline runs---> refresh
goto data tap--->refresh database--->views.

Part 6 - Data Reporting (Power BI)

Open power BI desktop--->choose get data option--->azure synapse analytics--->goto azure--->azure synapse resource--->goto properties--->serverless endpoint (copy link) --->paste it in power bi desktop--->server--->database(optional) (gold_db)--->Microsoft account--->sign in--->connect.

create relationship--->create reports, insights, patterns.

Part 7 - Security and Governance

#AAD (Azure Active Directory) and AKV (Azure Key Vault)

Goto azure resource group--->IAM--->role assignments--->no group.

#Azure Active Directory
Goto AAD--->groups--->new group--->group type(security)--->add owners--->add members--->create.

Goto azure resource group--->IAM--->add role assignments--->contributor--->next--->members--->assign access to (user, group or service principal) --->members (select members) --->select group.

Part 8

Run pipeline we have manual option (Debug, Add trigger).

Automate trigger--->add trigger--->name--->type(schedule)--->start date--->time zone (UTC)--->recurrence (every 1 day) --->advanced recurrence options--->click ok--->publish all.

Goto SSMS---> add two records--->execute.

Goto ADF--->click monitor--->pipeline runs--->refresh--->click pipeline name--->see the monitor.

Part 9
How to Send Email from Azure Logic Apps

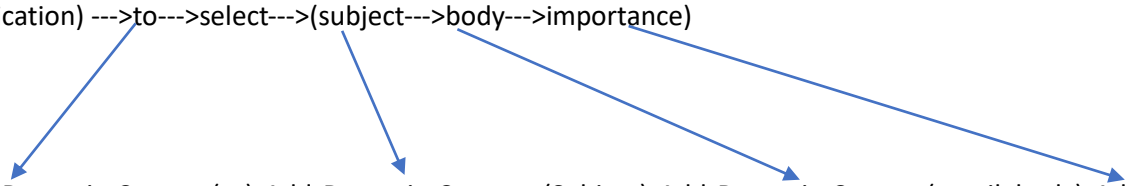
Goto azure--->resource group--->search for logic app--->create logic app--->give name--->Plan type (consumption plan) --->review and create.

Goto resource group--->logic app--->blank logic app--->request connector(triggers)--->http request--->use sample payload JSON.

```
{
        "type": "object",
        "properties ": {
                "to": {
                        "type": "string"
                },
                "subject": {
                        "type": "string"
                },
                "email_body": {
                        "type": "string"
                }
        }
}
```

Click done

Method--->post
New step--->Gmail--->send email(v2) --->connection name (Gmail)--->Authentication type (use default shared application) --->to--->select--->(subject--->body--->importance)



Add Dynamic Content(to) Add Dynamic Content (Subject) Add Dynamic Content(email_body) Add Dynamic Content (High)

Save

Postman (is third party tool) --->collections--->new request--->POST--->HTTP POST URL of API--->params (Body--->none(raw)--->text (JSON--->)

```
        {
                "to": {
                        "type": "string"
                },
                "subject": {
                        "type": "string"
                },
                "email_body": {
                        "type": "string"
)
```

Send

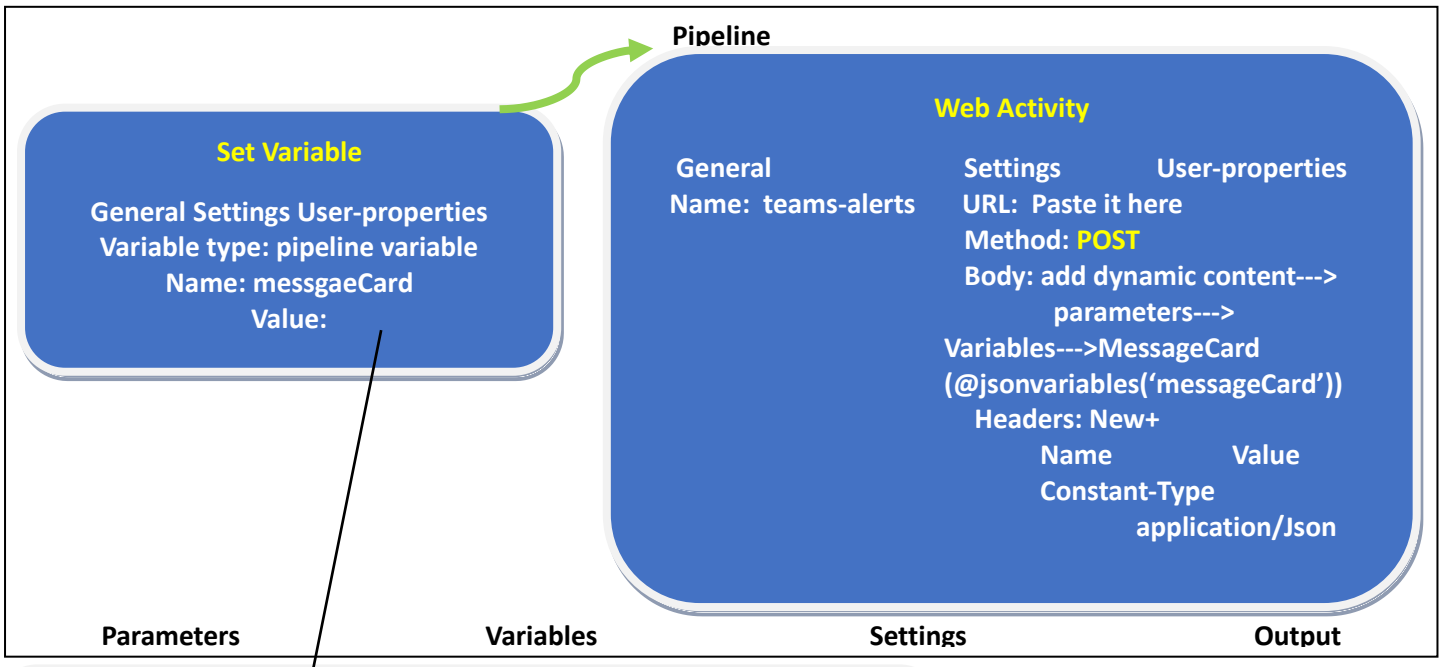
Part 10
How to Send bulk Email from Power Automate

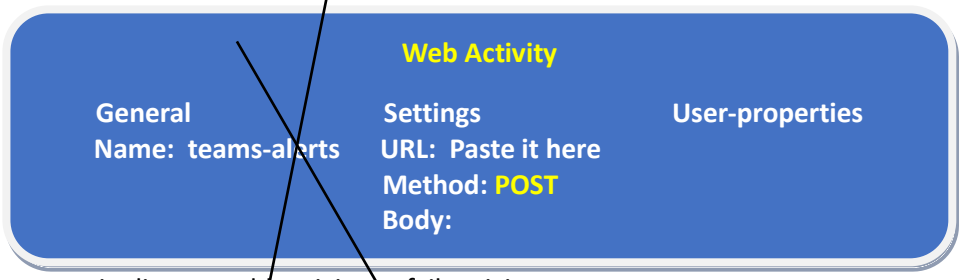
Login with domain mail id--->create--->instant cloud flow--->flowname(email)--->choose how to trigger this flow--->Excel Online (OneDrive)--->List rows Present in tables(add--->file (from OneDrive--->Table (Table))--->Gmail (send email(V2).

Part 11
Sending ADF Pipeline alerts to Microsoft Teams

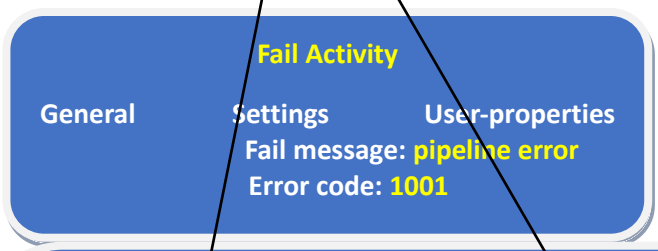
Goto Microsoft teams--->Click Teams--->General--->connection--->add Incoming Webhook--->create--->copy URL

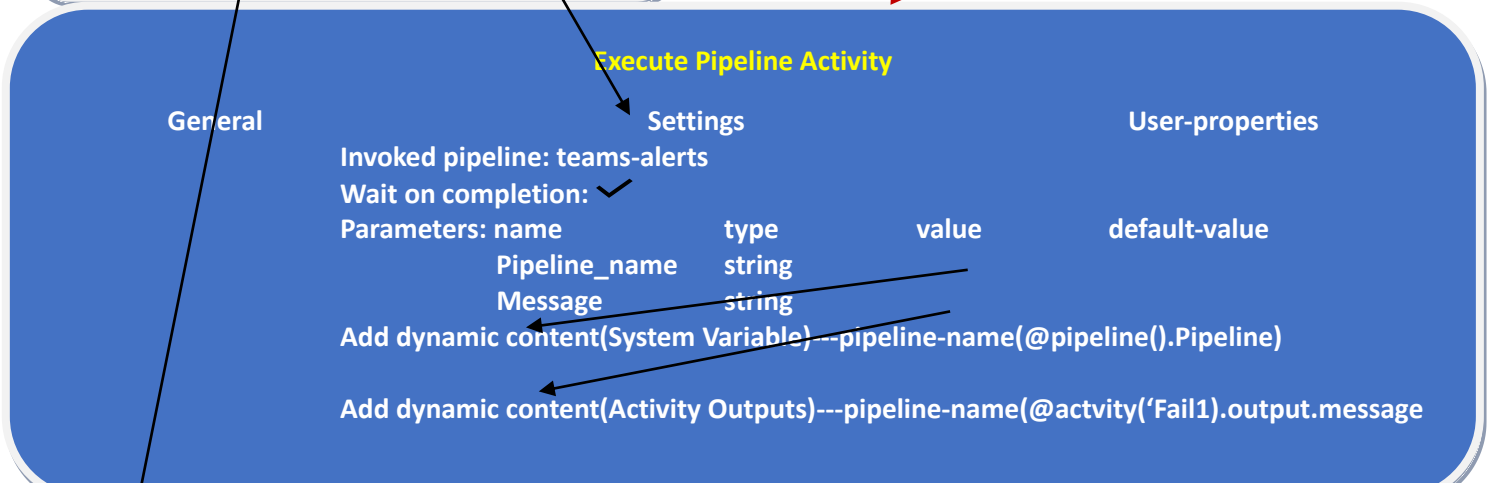
Goto Azure ADF--->create pipeline--->add activity--->web activity--->click run





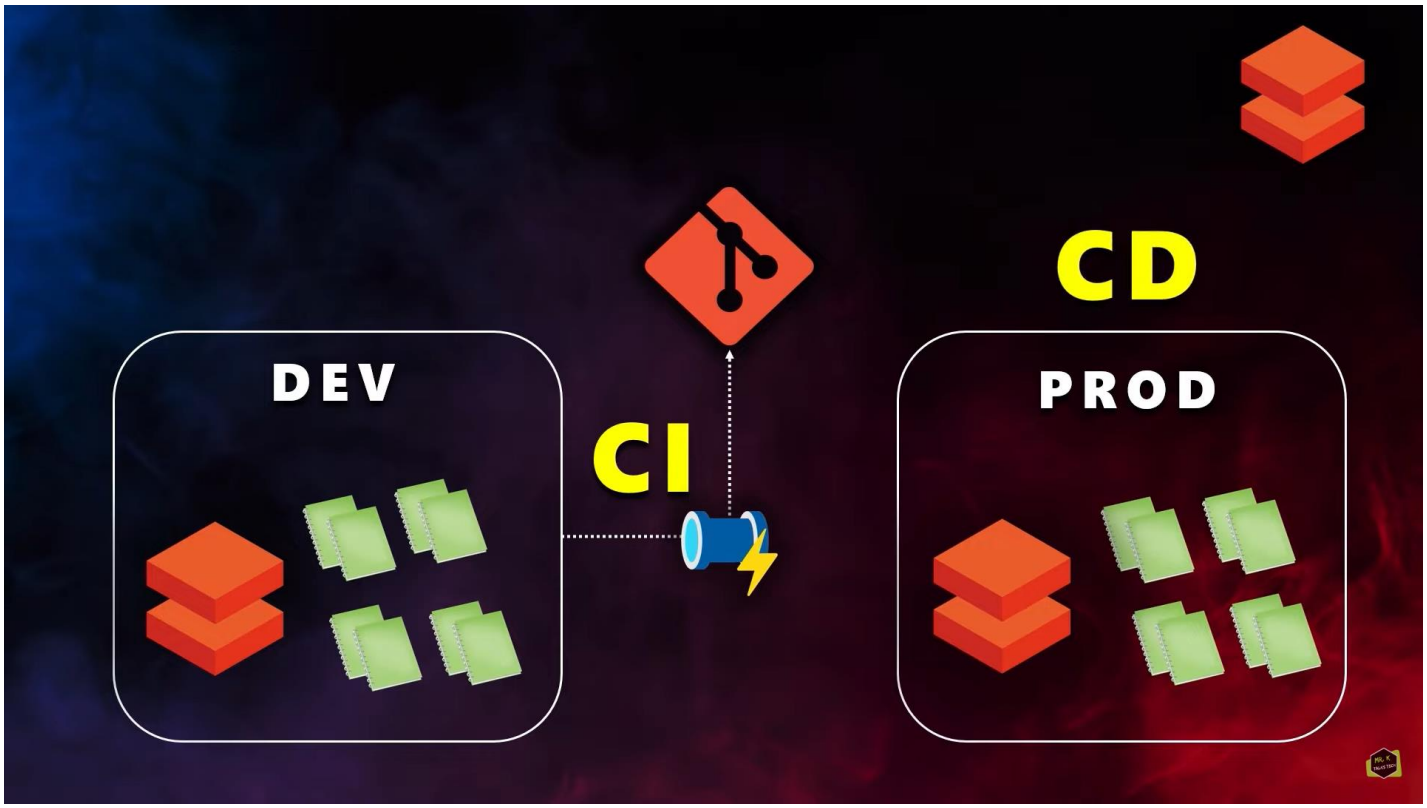
create pipeline---> add activity--->fail activity





```
{
        "@type”: "MessageCard”,
        "@context": "http://schema.org/extensions",
        "themeColor": “00706D7”
        " summary”: "pipeline status alert message",
        "sections”: [
                {
                        "activityTitle": "pipeline execution alert",
                        "facts”: [
                                {
                                        "name": "pipeline Name:”,
                                        "value": “@pipeline(). parameters. pipeline _ name}"
                                },
                                {
                                        "name" • "Message:”,
                                        "value":  @pipeline().parameters(). message}"
                                }
                        ],
                        "markdown": true
                }
        ]
}
```

Part 12: Complete CI/CD pipeline



What CI/CD?

-It is a set of practices used to automate and streamline the process of building, testing, and deploying code changes to different Environments.

There are Many Ways!!!

A company can have three environments

1. **Dev**
2. **QA(UAT)**
3. **Prod (Deploy)**

GitHub

Azure Devops

Data
Engineer 1

Azure
Databricks

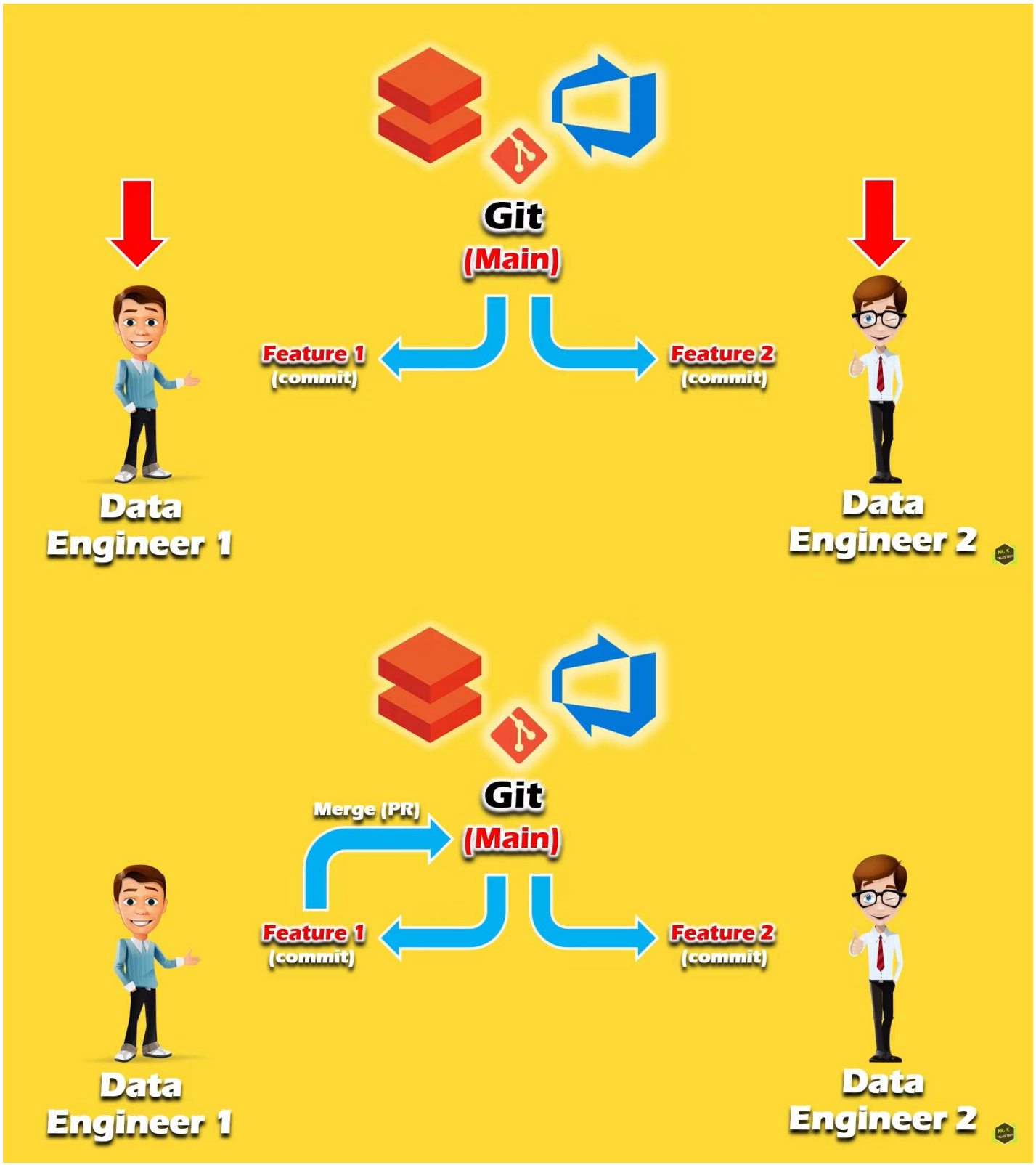
Data
Engineer 2

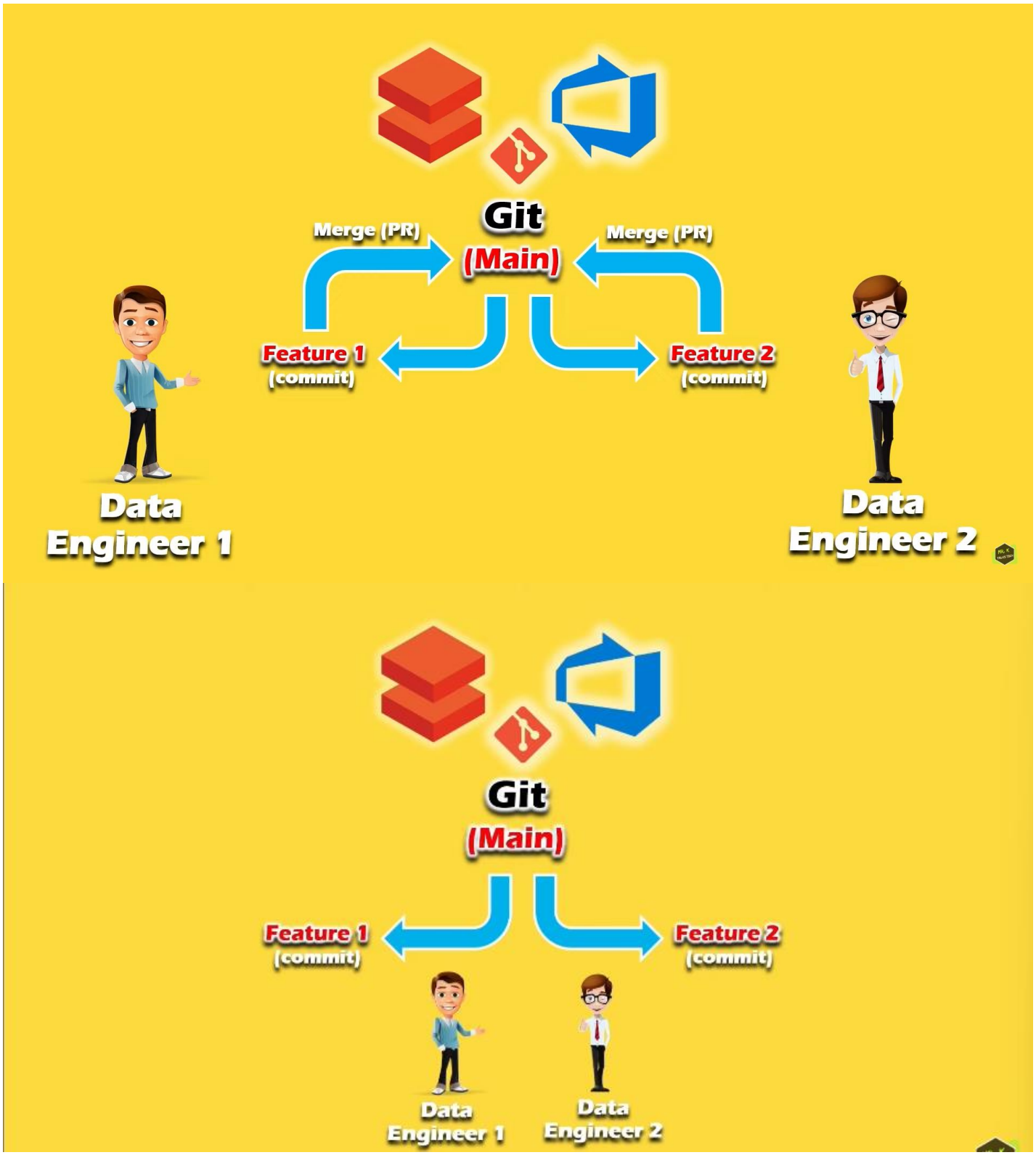
Data
Engineer 1

Azure
Databricks

Data
Engineer 2

Git


Git
(Main)
Feature 1
(commit)
Feature 2
(commit)
Data
Engineer 1
Data
Engineer 2
Git
(Main)
Merge (PR)
Feature 1
(commit)
Feature 2
(commit)
Data
Engineer 1
Data
Engineer 2



1. CI/CD
2. Setting up the Repos
3. Creating CI Pipeline
4. Creating CD Pipeline
5. Testing CI/CD Pipeline

1. Git Integration in Azure Databricks (Env Setup)

Two environments

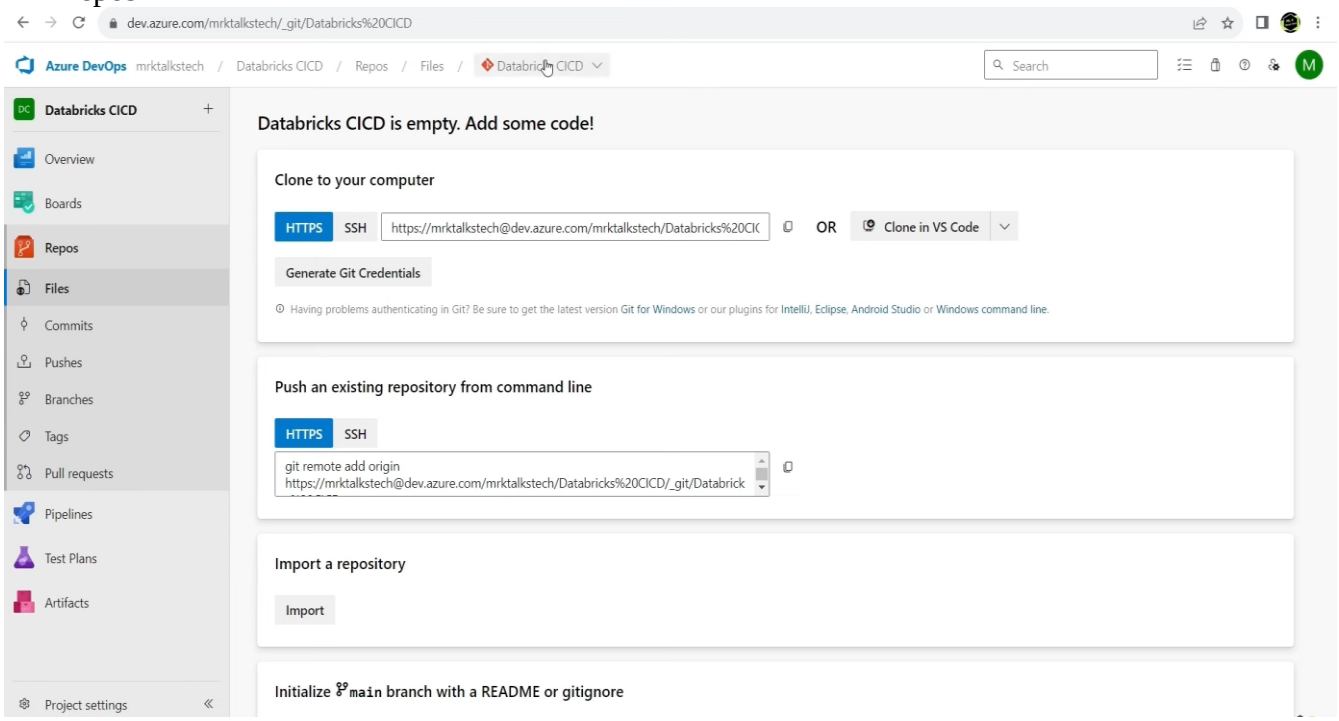
Dev **Prod**

No connection between dev and prod environments

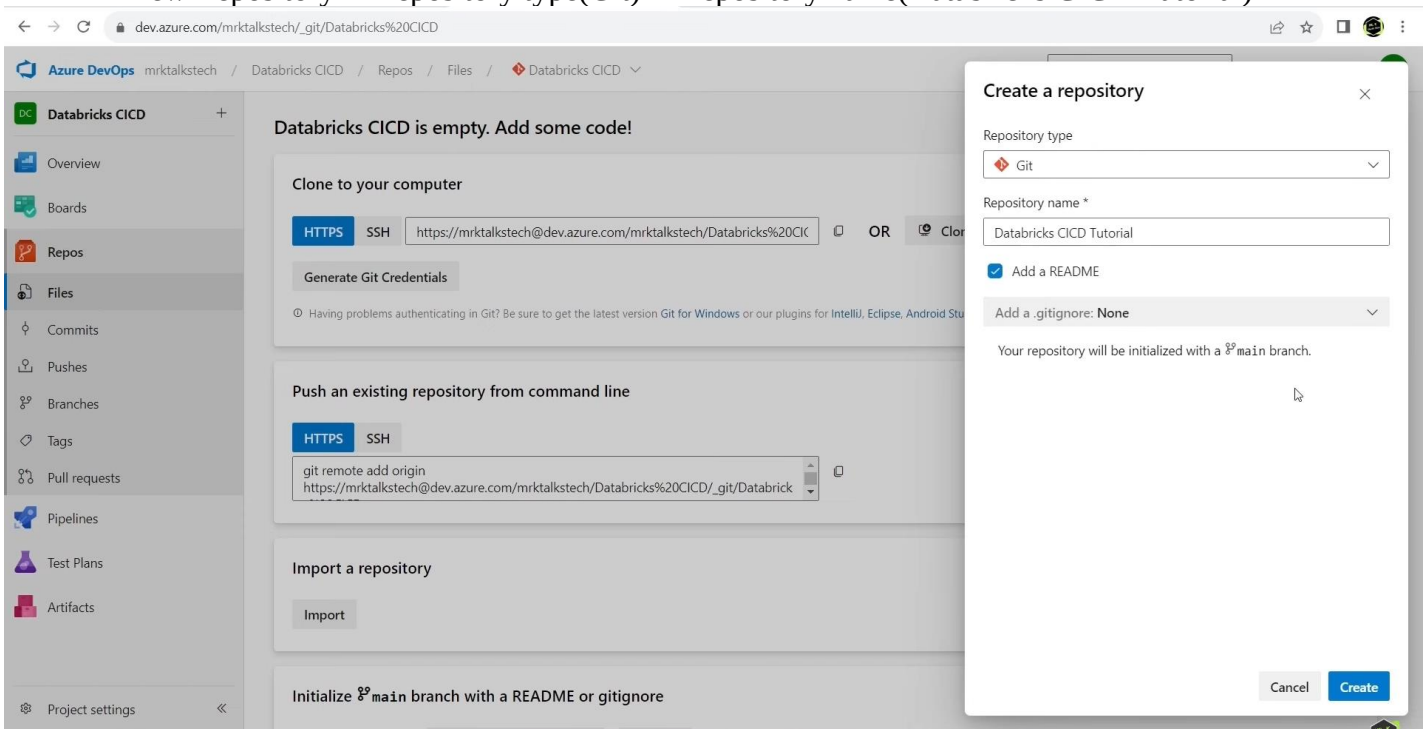
https://dev.azure.com/ganeshr0477/

Create new Project--->project name--->private--->click ok

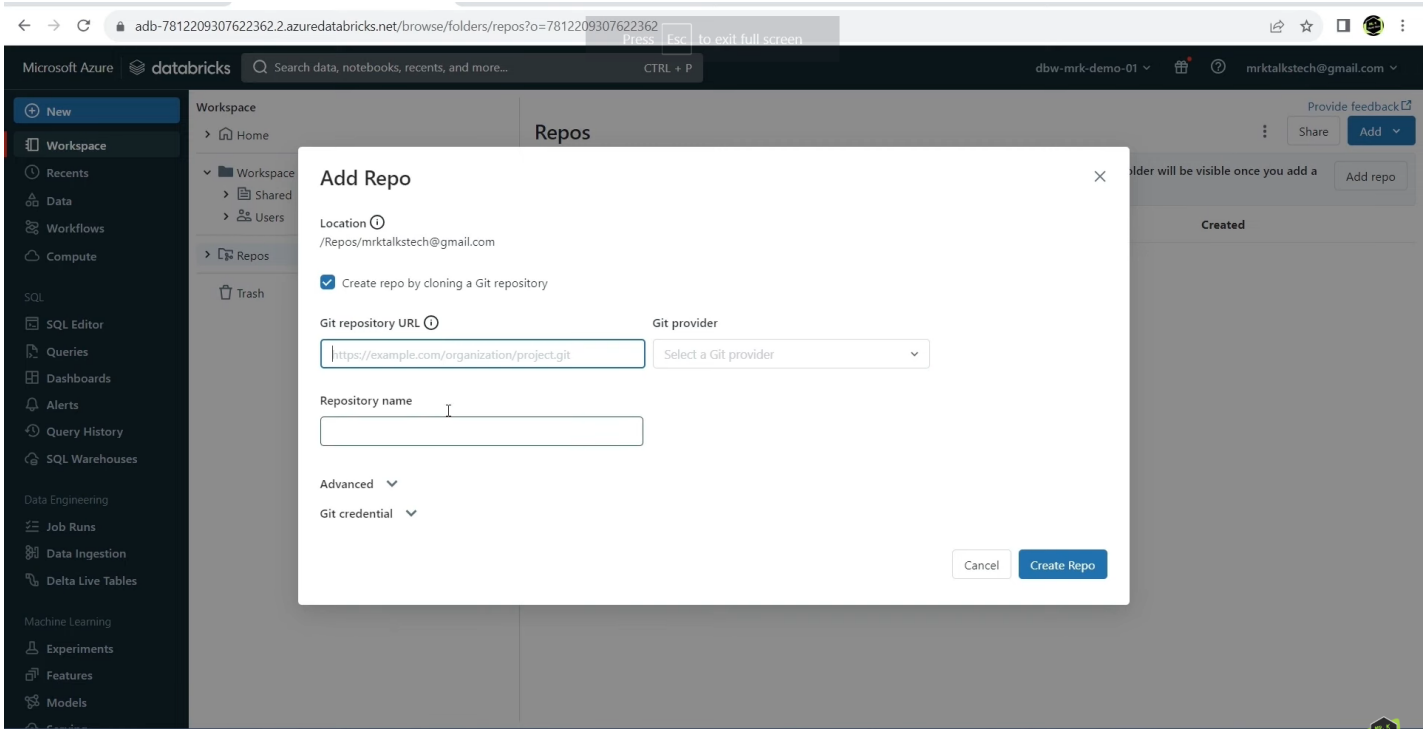
Repos

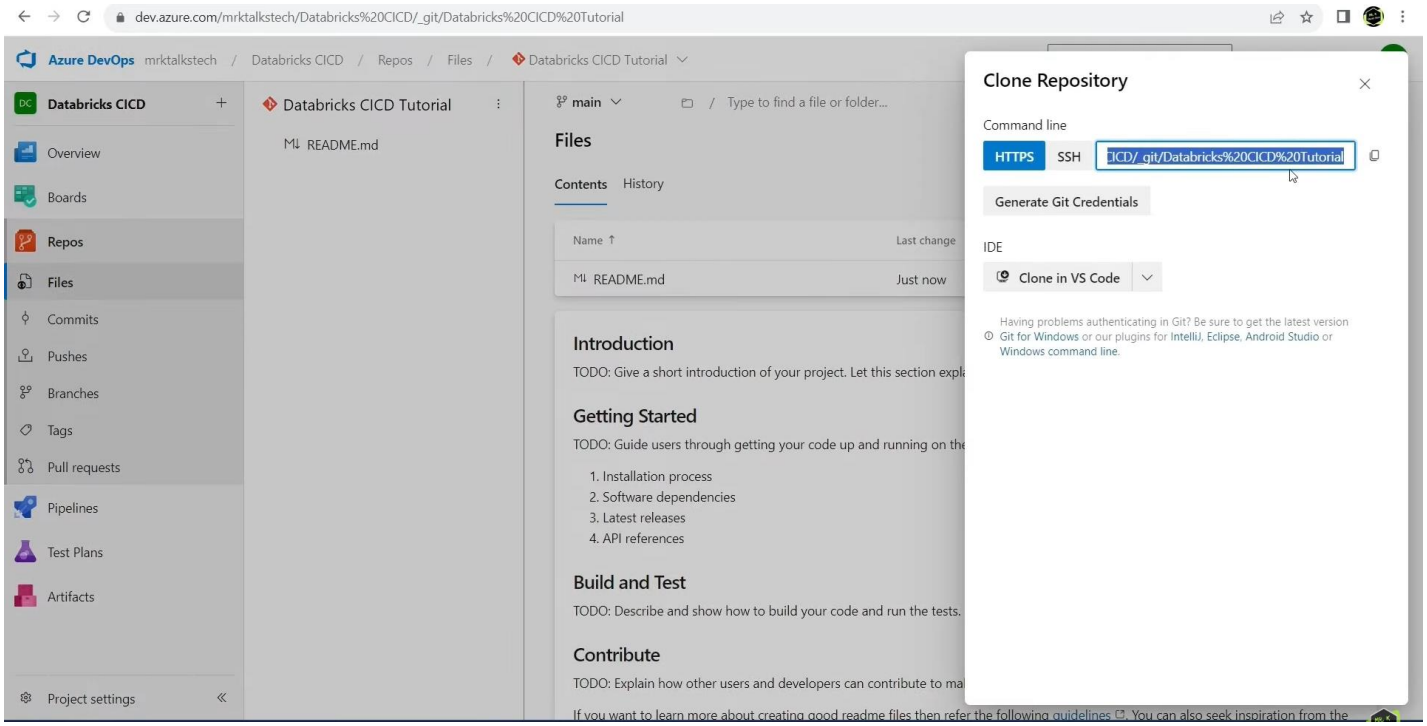


+ New Repository--->repository type(Git)--->repository name(Databricks CICD Tutorial)



Goto databricks dev workspace





Advanced:

Space checkout mode(uncheck)

Git credential:

Git Provider (Azure Devops service (Azure Active Directory))--->save

Create repo

We successfully created dev repos workspace

Rename

2. Main Branch Protection (Env Setup)

Databricks repos

Have only README file and one Main branch

Move file (Bronze to Silver) to repos--->…--->(move)--->Repos (repos name) --->move. --->confirm.

For committing the changes click(main branch)

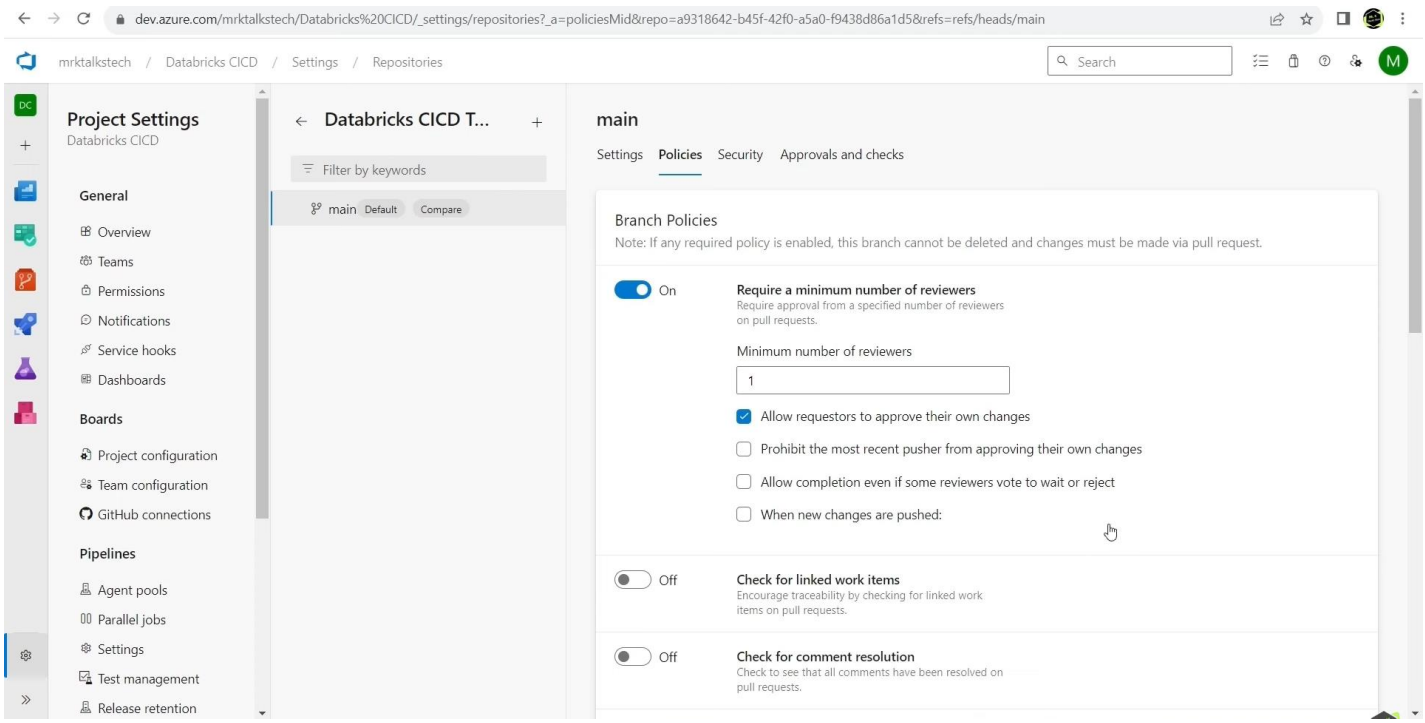
Added bronze to silver---> commit

For testing

Azure Devops--->files--->refresh.

To restrict changes anyone in file

Branches--->…--->Branch Policies--->you can see four options (enable first one)



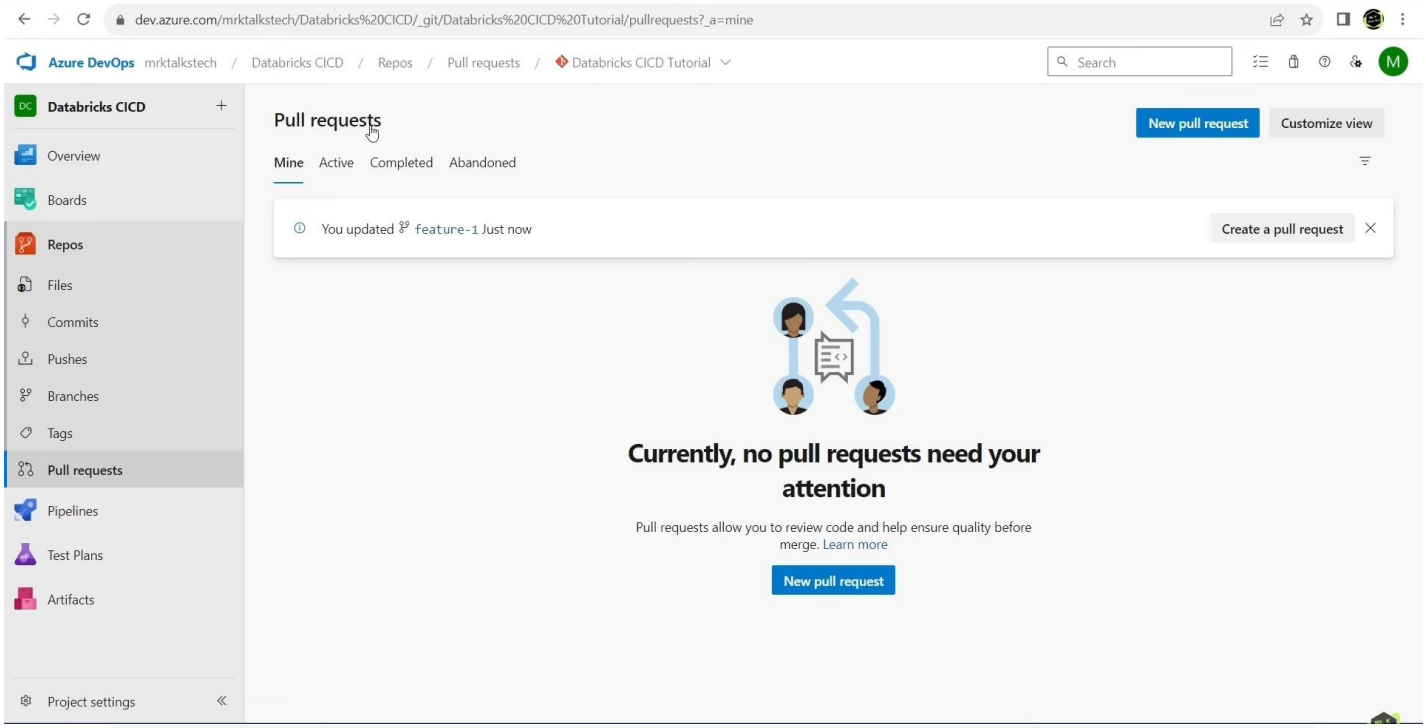
Same to that all file move databricks workspace to repos and click main option--->commit & push.

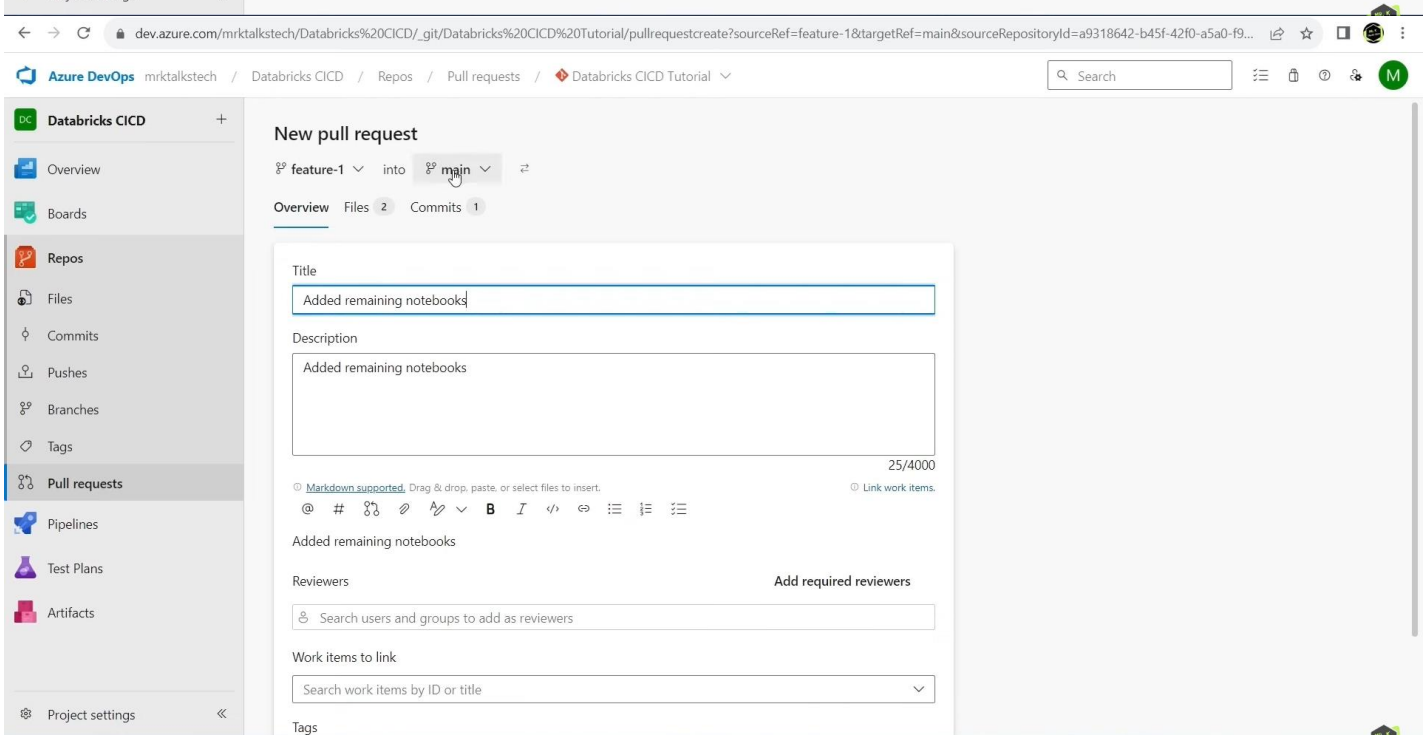
Getting error messages (now u can see successfully protected)

Create a Branch--->feature-1--->create (you can all files)

Click feature-1 branch--->added commit changes

Main branch having only two files
So now pull request
Goto azure Devops--->Repos--->create a pull request





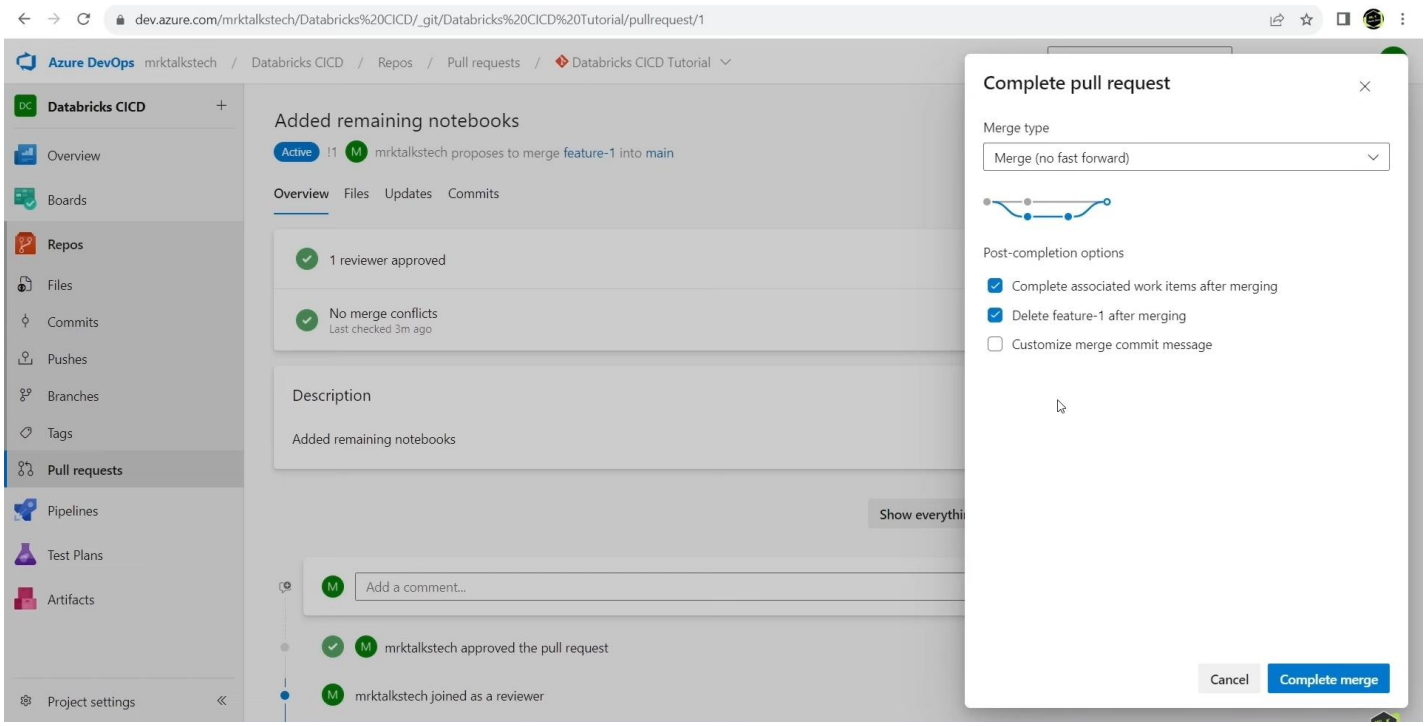
Source branch and target branch.

Overview Files Commits

Create

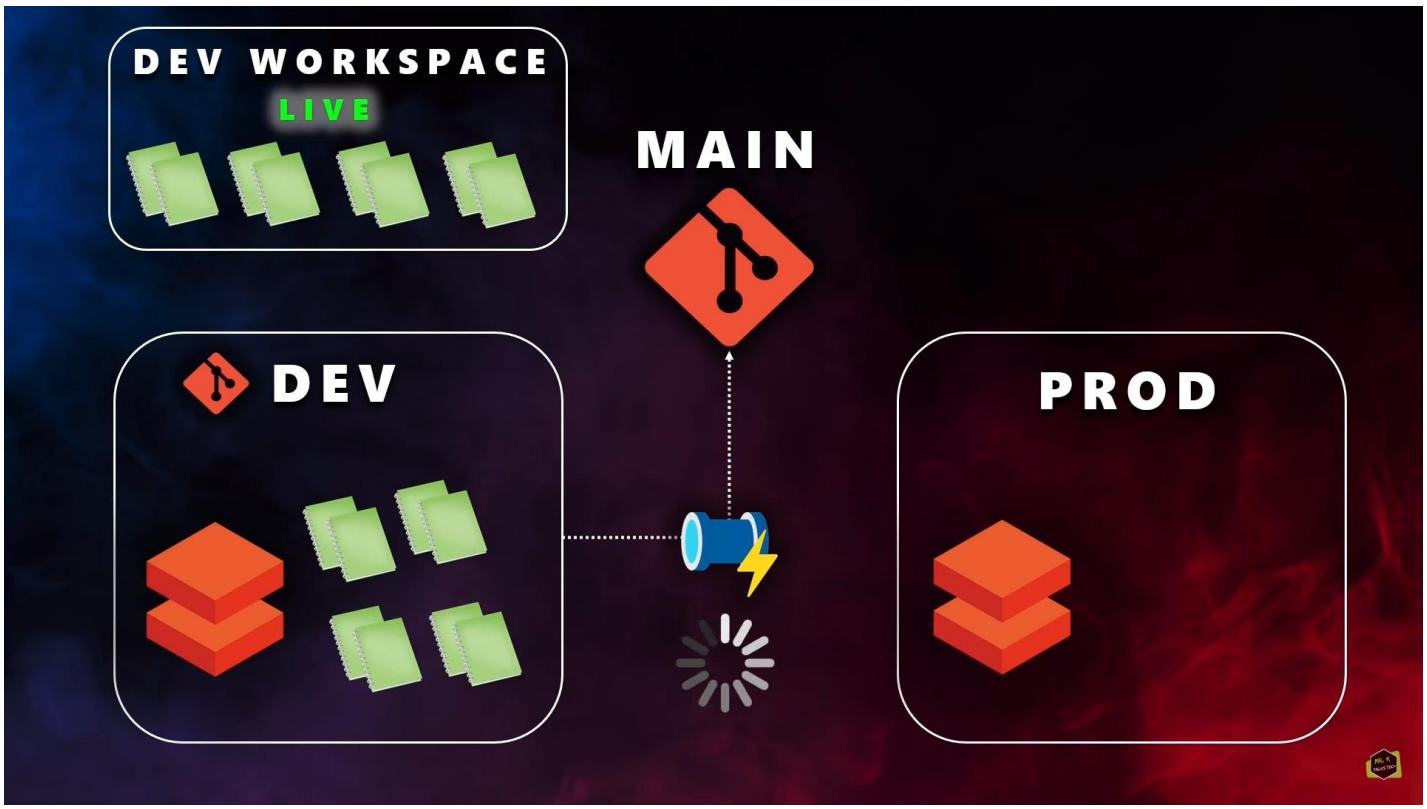
Overview Files Updates Commits

Approve (drop down--->approve) Commit (drop down--->Commit)



Goto repos--->Files

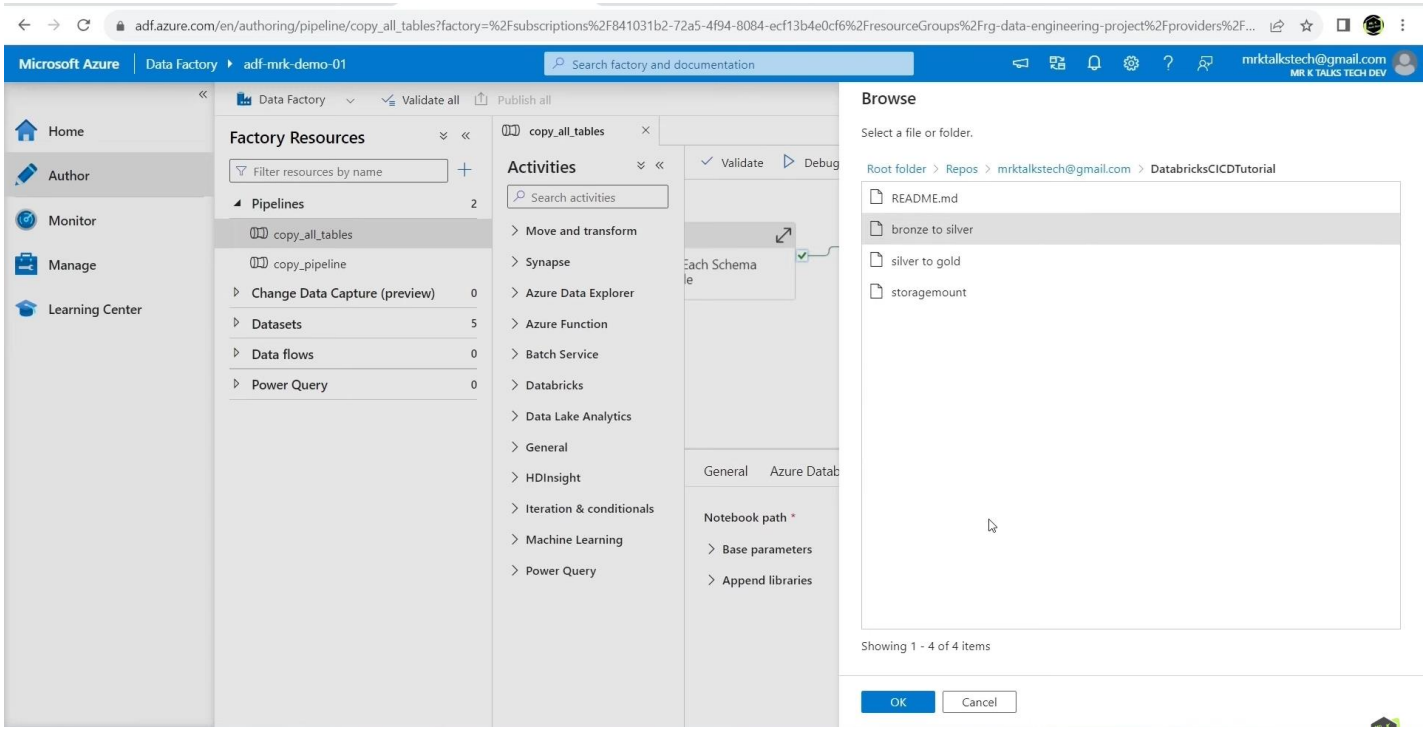
3. Continuous Integration (Cl) Pipeline process explained



No files in workspace.
Only it contains repos folder.

Goto Databricks workspace--->right click--->create--->folder(live)

Goto ADF--->pipeline--->if run (getting an error)
Notebook(Bronze to Silver)--->settings--->browse--->repos



Delete live folder

4. Organizing the folder structure needed for Cl / CD Pipeline

Goto Databricks workspace--->click on main--->create a branch(organize)

Right click on create (folder--->notebook)
Move files dragging and drop

Right click on create another (folder--->extra) #this will deploy to prod environment.

Click on organize branch
You see D--->Delete
A--->Added

Change the file location

Goto azure Devops------->refresh

Create pull request

Change organize(branch)---------->main(branch)
Create

Click approve

Click complete

Goto azure databricks organize--->drop down(main)

Goto azure Devops we have only main branch

For creating ci a pipeline set of code that we need to write and merge that onto the main branch of this repository that code we are using for building this CI/CD pipelines is called as YAML code.

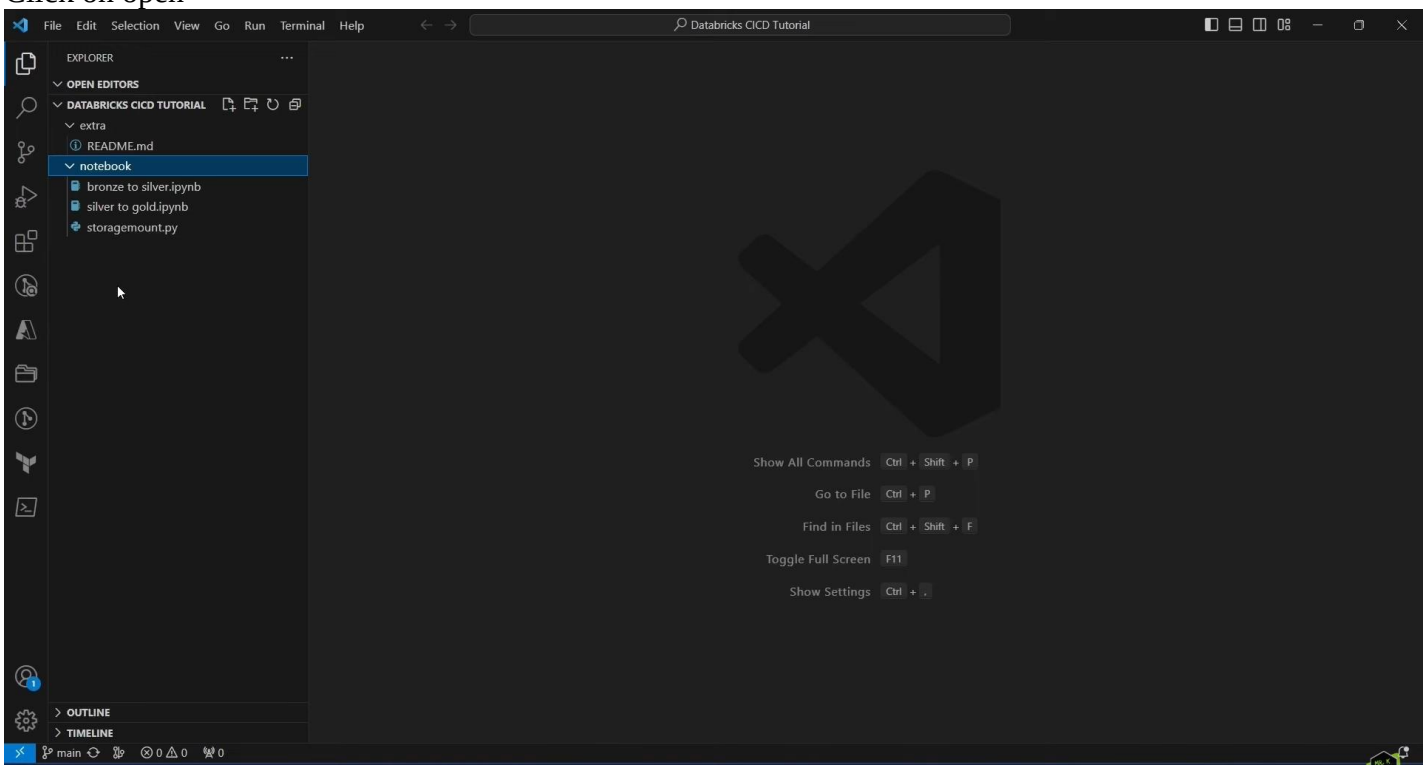
We use visual studio code instead of Databricks.

For doing that we have option called Clone.

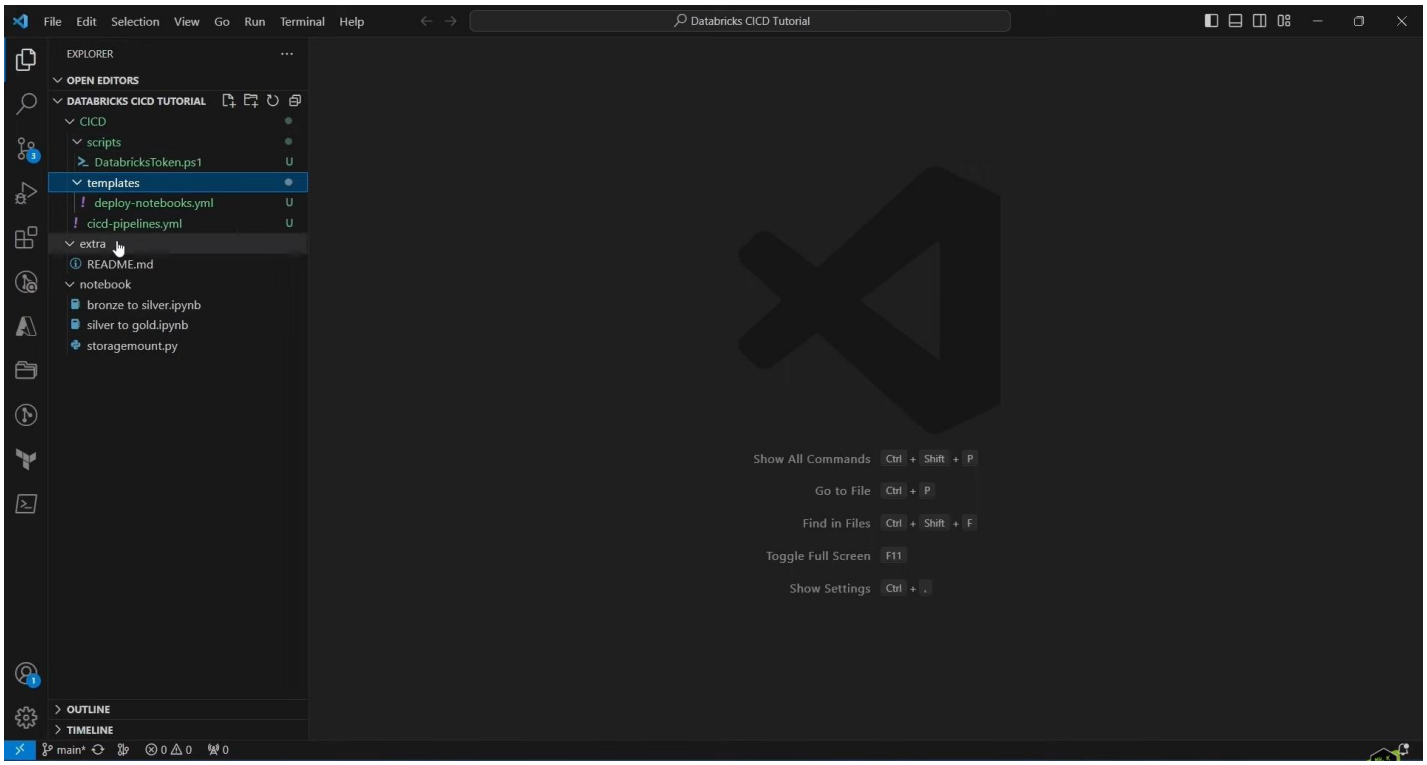
Click clone in VS Code.

Create a local folder

Click on open



Copy folder (CI/CD)
Expand



Click main branch
Create new branch(feature/ci-pipeline)

5. YAML code for Cl Pipeline and Variable Group, Pool-VM Image (Compute)

DatabrickToken.ps1

```
param
(
        [parameter(Mandatory = $true)] [String] $databricksWorkspaceResourceId,
        [parameter (Mandatory = $true)] [String] $databricksWorkspaceUr1,
        [parameter (Mandatory = $false)] [int] $tokenLifeTimeSeconds = 300
)
$azureDatabricksprincipalId -'2ff814a6-3304-4ab8-85cb-cdøe6f879cId '

$headers =@{}
$headers["Authorization"] - "Bearer $((az account get-access-token--resource
$azureDatabricksprincipa1Id I ConvertFrom-Json) . accessT(

$headers["X-Databricks-Azure-SP-Management-Token"] = "$((az account get-access-token --resource
https://management.core.windows.net/
$headers["X-Databricks-Azure-Workspace-Resource-Id"] = $databricksWorkspaceResourceId
$json =@{}
$json[“lifetime_seconds”]=$tokenLifeTimeSeconds

$req = Invoke-WebRequest -Uri "https://$databricksWorkspaceUr1/api/2.0/token/create"-Body ($json
I convertTo-Json)
$bearerToken = ($req.content I ConvertFrom-Json).token_value

return $bearerToken
```

deploy-notebooks.yml

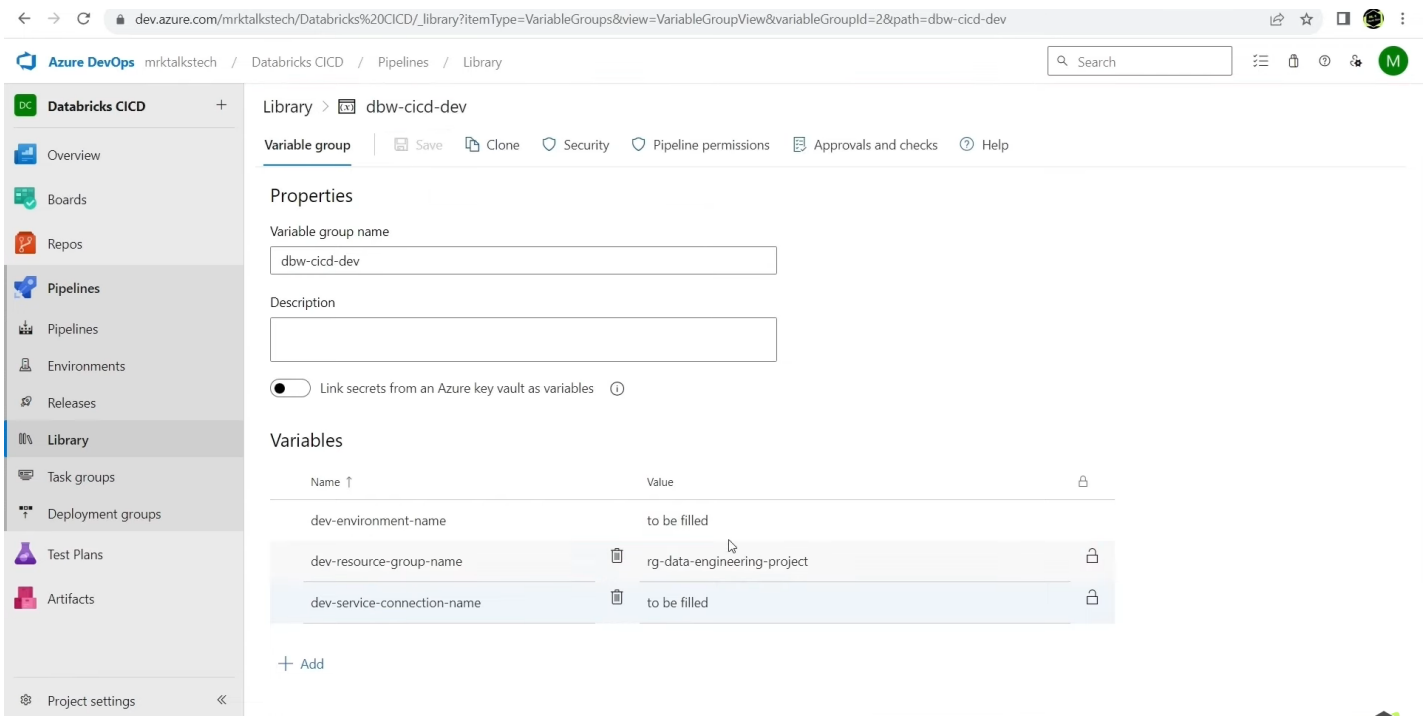
```
parameters :
    - name: stageId
      type: string
    - name: dependson
      type: object
      default: []
    - name: environmentName
      type: string
    - name: resourceGroupName
      type: string
    - name: serviceconnection
      type:string
    - name: notebooksPath
      type:string
stages :
    - stage: "${{ parameters.stageId } } "
      displayName: "Deploying to [${{upper(parameters.env)}}] Environmen
      dependson: ${{ parameters. dependson }}
      jobs:
          - deployment: Deploy
          displayName: "Deploying Databricks Notebooks"
          environment: ${{parameters. environmentName}} #template literal (${}) in javascript
          strategy:
              runonce:
                  deploy:
                      steps:
                      -   Checkout: self
                      -   Tasks: AzureCLI@2
                          Inputs:
                              azureSubscription: ${{parameters.serviceConnection}}
                              scriptType: “pscore”
                              scriptLocation:'InlineScript'
                                  InlineScript: |
                                      Az config set
                          extension.use_dynamic_install=yes_without_prompt
                              $databrickaworkspace =(az resource list –resource-group
                          ${{parameters.resourceGroupName}} –query
                          “[?type==’microsoft.Databricks/workspaces’]”|ConvertFrom-
                          Json)[0]

                          $databrickaworkspace =(az databricks workspaces show –ids
                          $ Databricks/workspace.id| ConvertFrom. Json)

                          $bearerToken =
                          $(Build.Repository.LocalPath)CICD/scripts/DatabricksToken
                          .ps1 DatabricksworkspaceResourceId
                          $databricksworkspaceInfo.id -databricksspaceUrl
                          $databricksspaceInfo.worksapceUrl

                          Install-Module -Name azure.databricks.cicd.tools -Force -
                          Scope CurrentUser
                          Import-Module -Name azure.databricks.cicd.tools
                          Import.DatabricksFolder -BearerToken $ bearerToken -
                          Region $DatabricksworkspaceInfo.location -LocalPath
```

$(Build.Repository.LocalPath)/${parameters.notebooksPath}
} -DatabricksPath '/live' -Clean

cicd-pipelines.yml

```
trigger:
    - main
variables:
        group: dbw cicd dev
    -   name:vmImageName
        value:"windows-latest"
    -   name:notebookspath #files notebook path dev to prod
        value:"notebook"
pool:
    vmImage: $(vmImagename #compute power
stages:
    - template: templates/deploy-notebooks.yml
      parameters:
            stageId: "Deploy to Dev Environment"
            env: "dev"
            environmentName: $(dev-environment-name)
            resourceGroupName: $(dev-resource-group-name)
            serviceconnection: $(dev-service-connection-name)
            notebookspath: $(notebookspath)
```

goto repos--->pipelines--->library--->variable group(click new)



6. Environments and Service connection

Goto Azuredevops--->repos--->pipelines--->environments--->create(dev-environment-databricks-cicd)--->none

Goto Azure resource group

Access control (IAM)--->Role Assignments

Goto project settings--->service connection--->new(ARM)--->Service principal(automated)--->next--->subscription(select)--->resource group.--->service connection name.(dev service connection)--->save.

Goto again Azure resource group

Access control (IAM)--->Role Assignments

7. Deploy Notebook Functionality and Generating Databricks Token

Goto azure portal copy that AzureDatabricksPrincipalid
in tenant

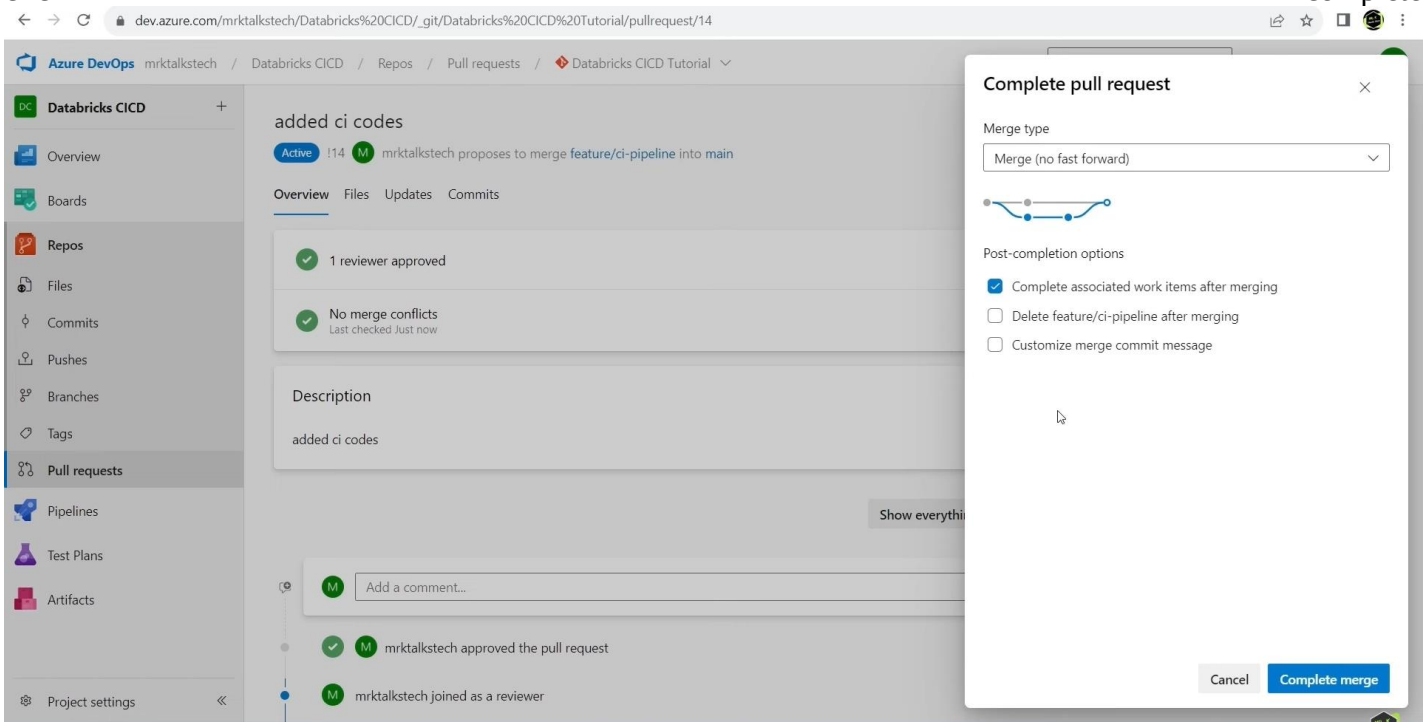
goto source control option
code added ci
commit

merging file
click pull request

once verify changes

give meaning full title
click create

click complete



click complete merge button

goto repos

switch from feature branch to main branch

8. Creating Pipeline in Azure DevOps

Click pipeline--->new--->azure repos git YAML--->select a repository(Databricks CICD tutorial

Existing azure pipeline YAML file--->Branch(Main)-→path(/CICD/cid.pipelines.yml)--->continue

Review--->Run(save)

You can see review file

Goto environments--->dev env--->Add resource(3 dots click--->security--->+(databricks CICD Tutorial)

Click Library--->dbw cicd dev (Pipeline permissions--->+ databricks CICD Tutorial)

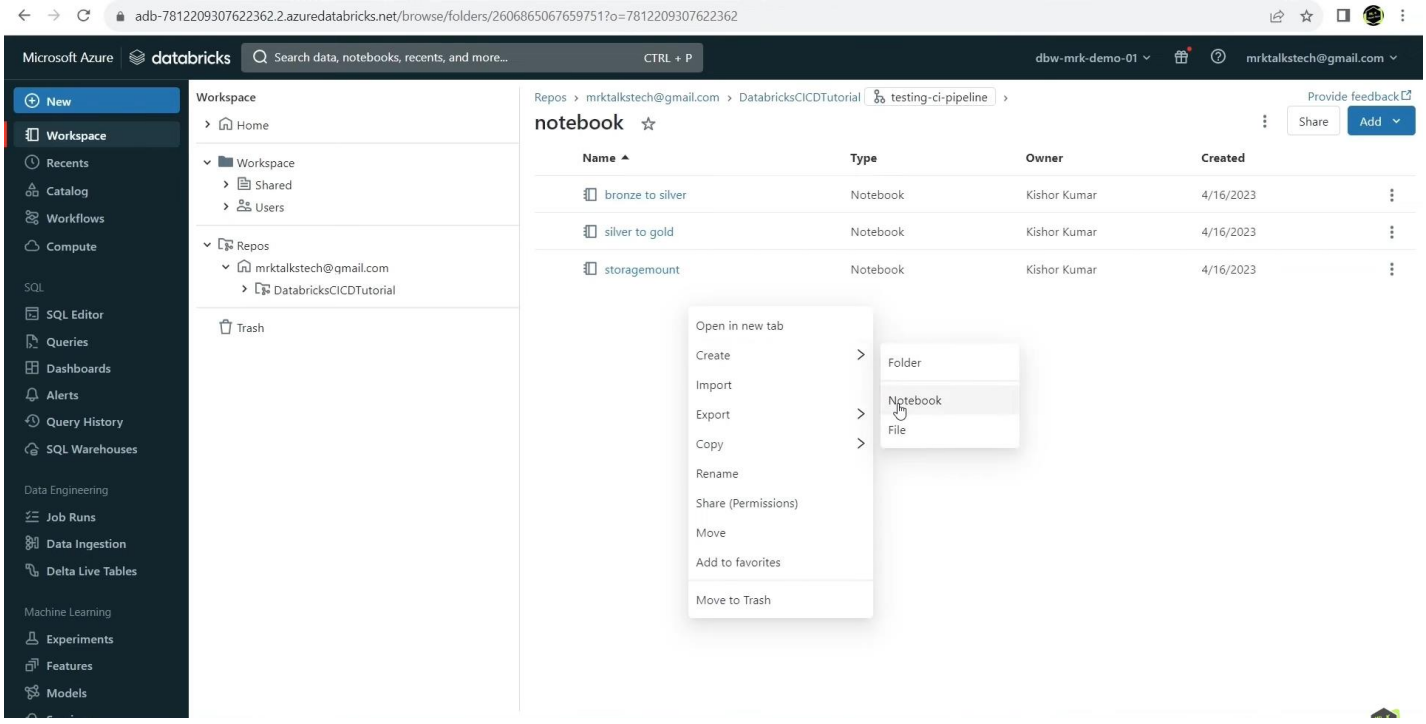
Project settings--->dev-service-connection--->service connection---> edit(3 dots option--->security)

Pipeline permission--->+ databricks CICD Tutorial

Now test the file

Goto to databricks repos--->databricks CICD Tutorial

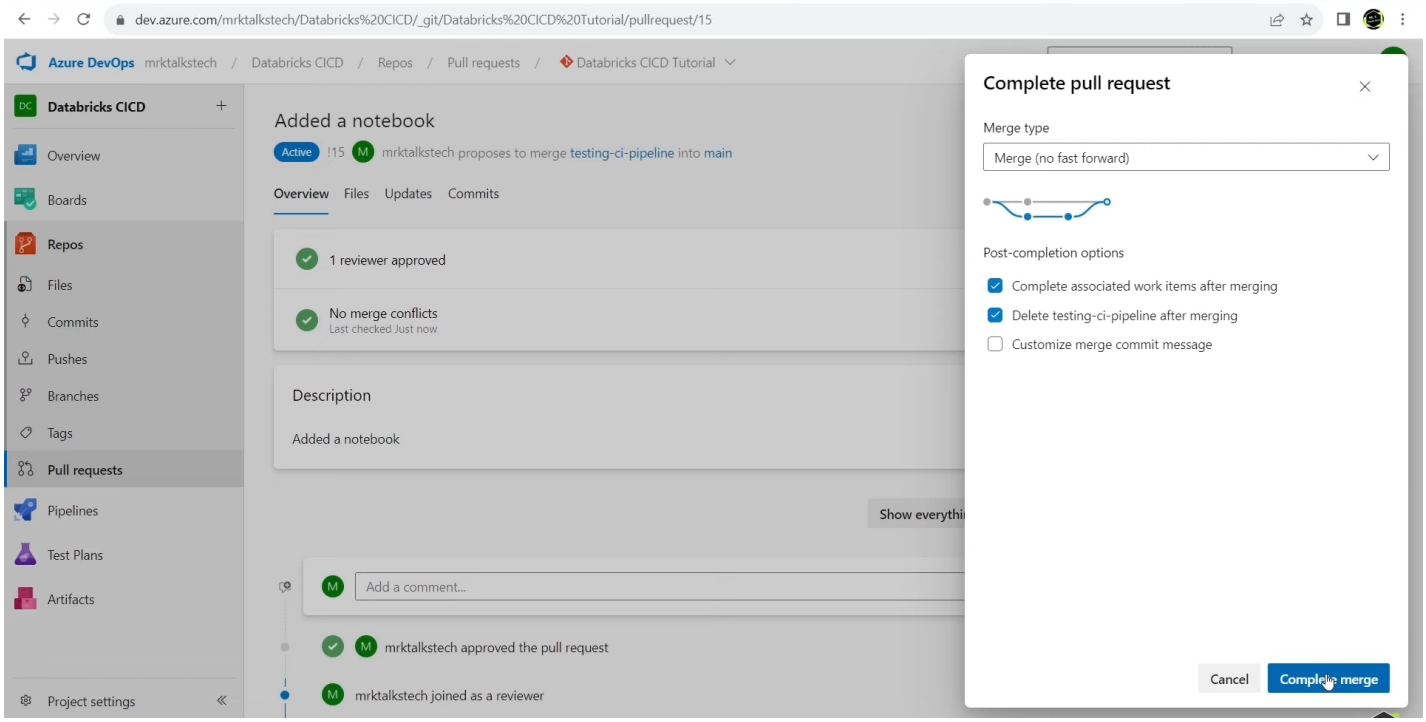
Click main branch--->create a branch(testing-ci-pipeline)--->right click--->create--->Notebook



Rename--->print(1+1)

Click on testing-ci-pipeline--->commit (added file)

Azure Devops--->azure repos--->create a pull request--->Files--->create--->complete



Goto pipelines
Click pipeline--->click on run

Goto environments--->

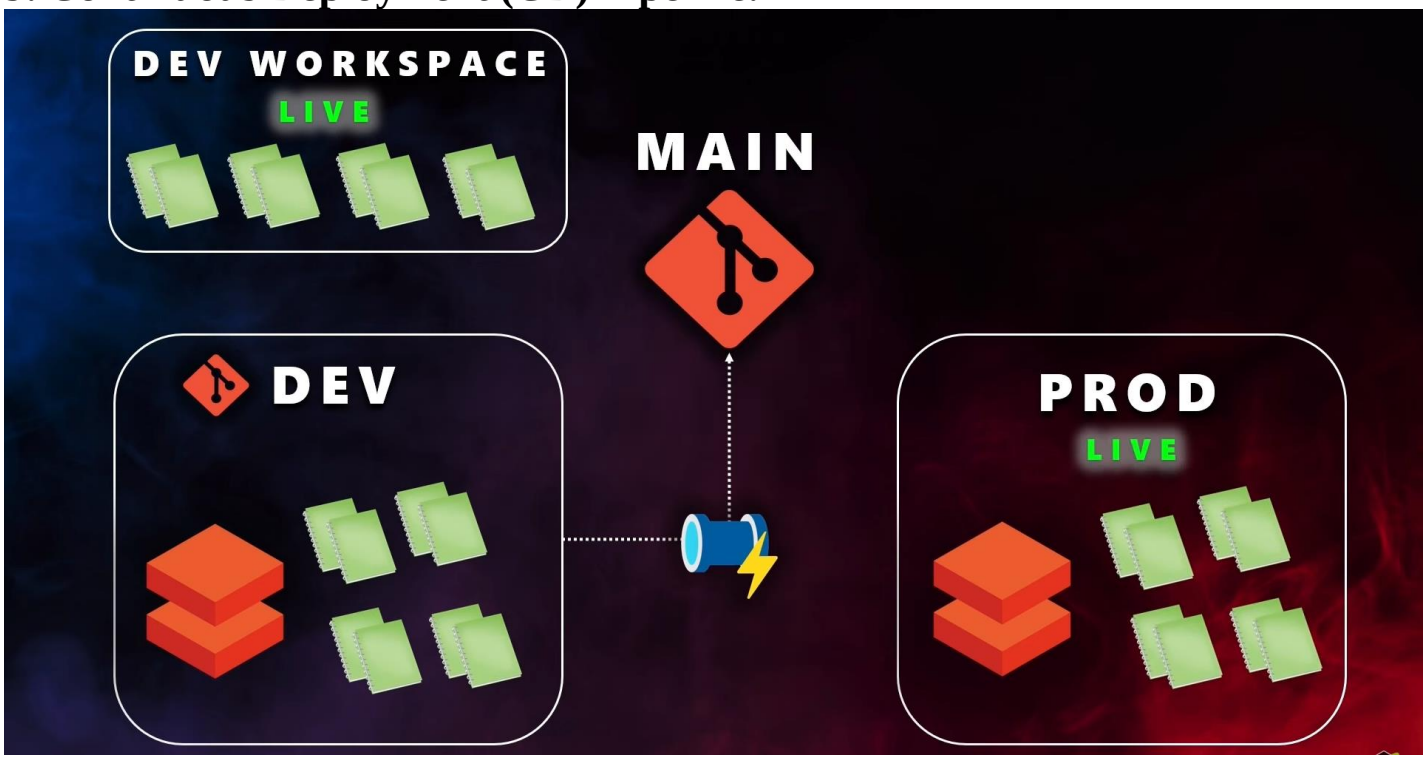
Goto databricks--->refresh--->workspace--->live(you see all notebooks)

Goto ADF--->pipeline--->Bronze to Silver(settings--->Browse--->live(Bronze to Silver))
Silver to Gold(settings--->Browse--->live(Silver to Gold))

Publish

It some of the real-time projects as part of the CI pipeline they will also be creating some Unit test functionalities which test your code before actually merging the changes to main branch.

9. Continuous Deployment (CD) Pipeline.



Goto VS Code
Create new branch for that bottom left corner click (feature)
Switch to feature branch to main branch
But will get few issues (previously we changes in main branch with pull request but here will not reflect in local VS Code)

In order to reflect we need change changes

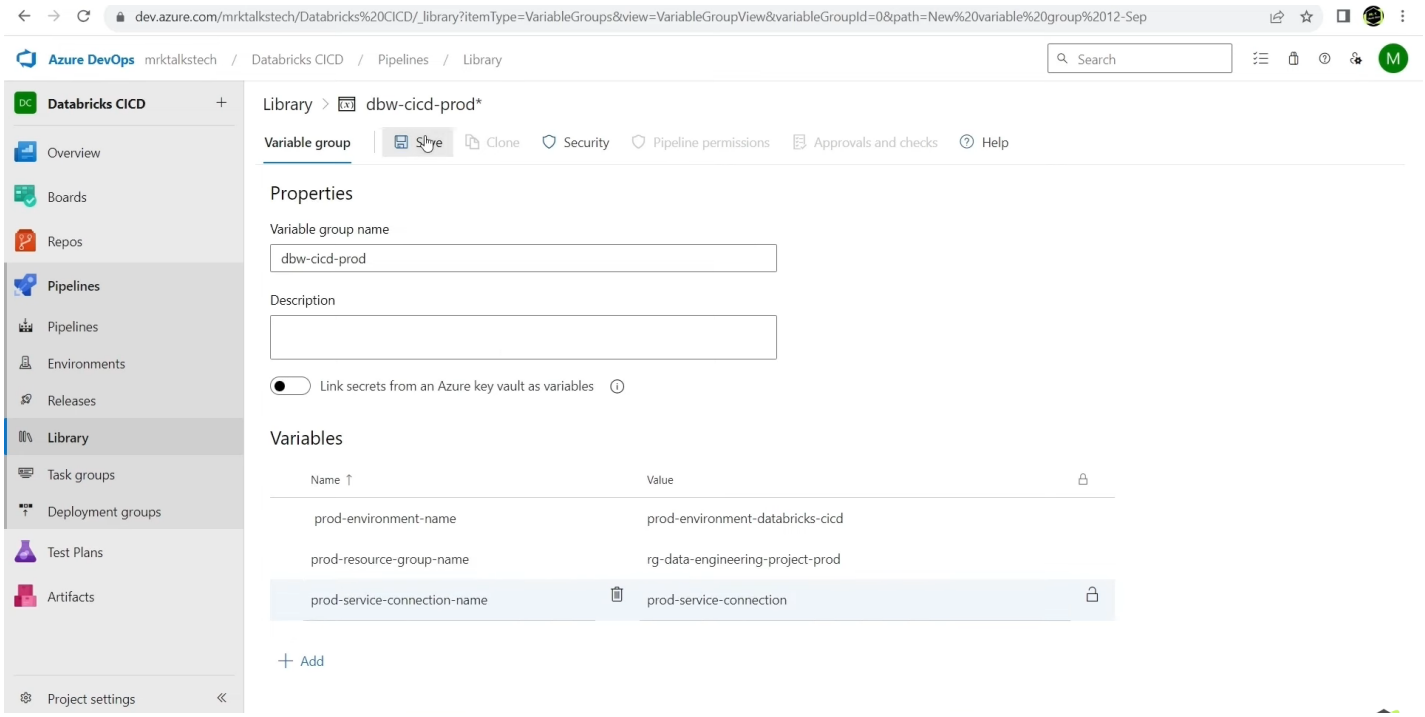
Goto source control---->click on (3 dots--->pull) #it will pull all latest code Azure Devops main branch to VS code

Click main branch bottom left corner

cicd-pipelines.yml

```
trigger:
    - main
variables:
    - group: dbw-cicd-dev
    - group: dbw-cicd-prod
    - name:vmImageName
      value:”windows-latest”
    - name:notebookspath #files notebook path dev to prod
      value:”notebook”
pool:
    vmImage: $(vmImagename #compute power
stages:
    - template: templates/deploy-notebooks.yml
      parameters:
          stageId: "Deploy_to_Dev_Environment"
          env: "dev"
          environmentName: $(dev-environment-name)
          resourceGroupName: $(dev-resource-group-name)
          serviceconnection: $(dev-service-connection-name)
          notebookspath: $(notebookspath)
    - template: templates/deploy-notebooks.yml
      parameters:
          stageId: "Deploy_to_Prod_Environment"
          env: "prod"
          environmentName: $(prod-environment-name)
          resourceGroupName: $(prod-resource-group-name)
          serviceconnection: $(prod-service-connection-name)
          notebookspath: $(notebookspath)
```

goto pipelines--->dbw-cicd-dev--->Library--->+variable group(dbw-cicd-prod)--->add



Goto environments--->new environment--->none

Next one is service connection
Project settings--->service connection--->new service connection(ARM)--->service principal(automate)--->next.--->sign in into authentication--->resource group(name)--->Prod service connection--->save.

Goto prod resource group

Before testing assign the permissions

Pipeline--->library--->pipeline permission---> +databricks CICD Tutorial

Environment--->prod-environment-databricks-cicd--->add resource (3 dots)--->security--->pipeline permissions--->+ databricks CICD Tutorial

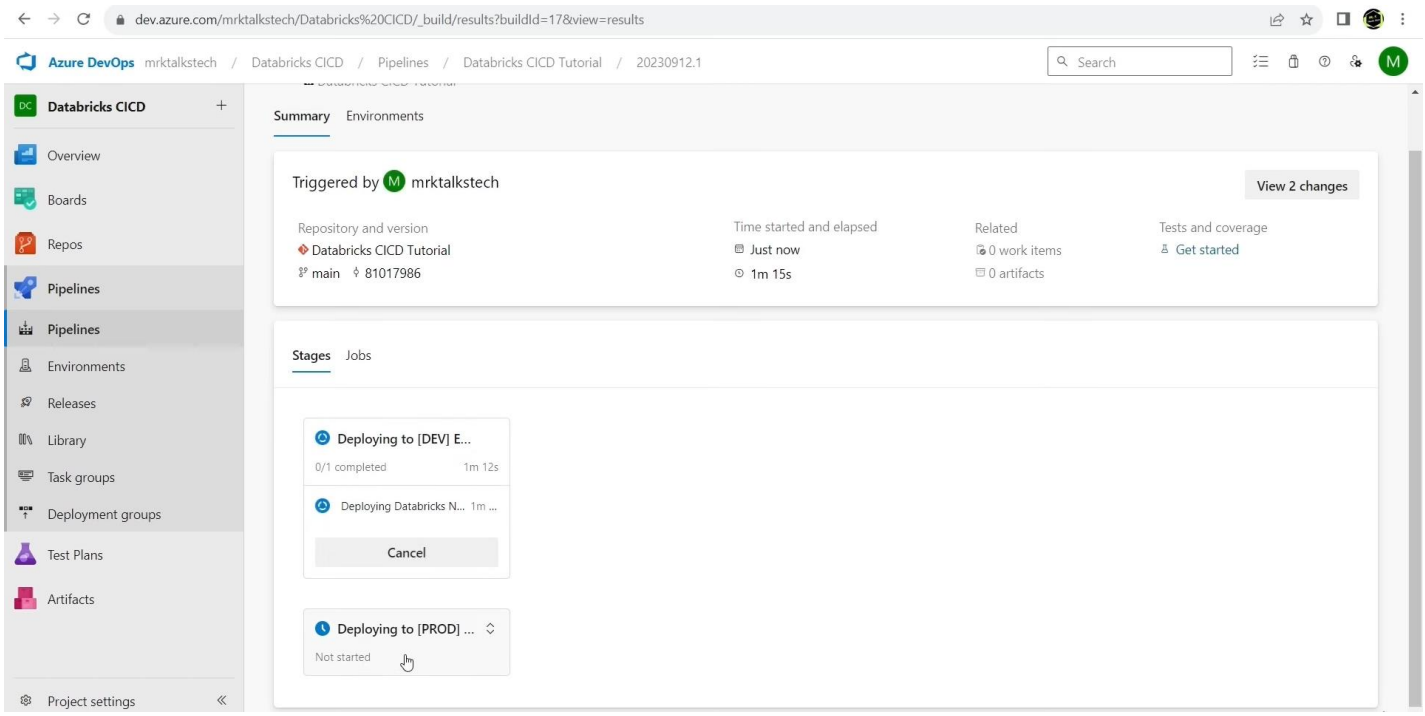
The final access is service connection
Prod-service-connection--->edit(3 dot)--->security---> permissions--->+ databricks CICD Tutorial

Now CICD pipeline have access to all prod version of variable group, environment and service connection
Now we began testing the CICD deployment functionality firstly lets merge all new changes for feature branch for that will
go to source control--->add cd pipeline changes(commit)

goto repos--->create a pull request--->create--->approve--->complete-→complete merge

goto pipeline

lets click on pipeline run



Here prod stage is not dependent on dev stage
To make that dependency we need to change small change in YAML.

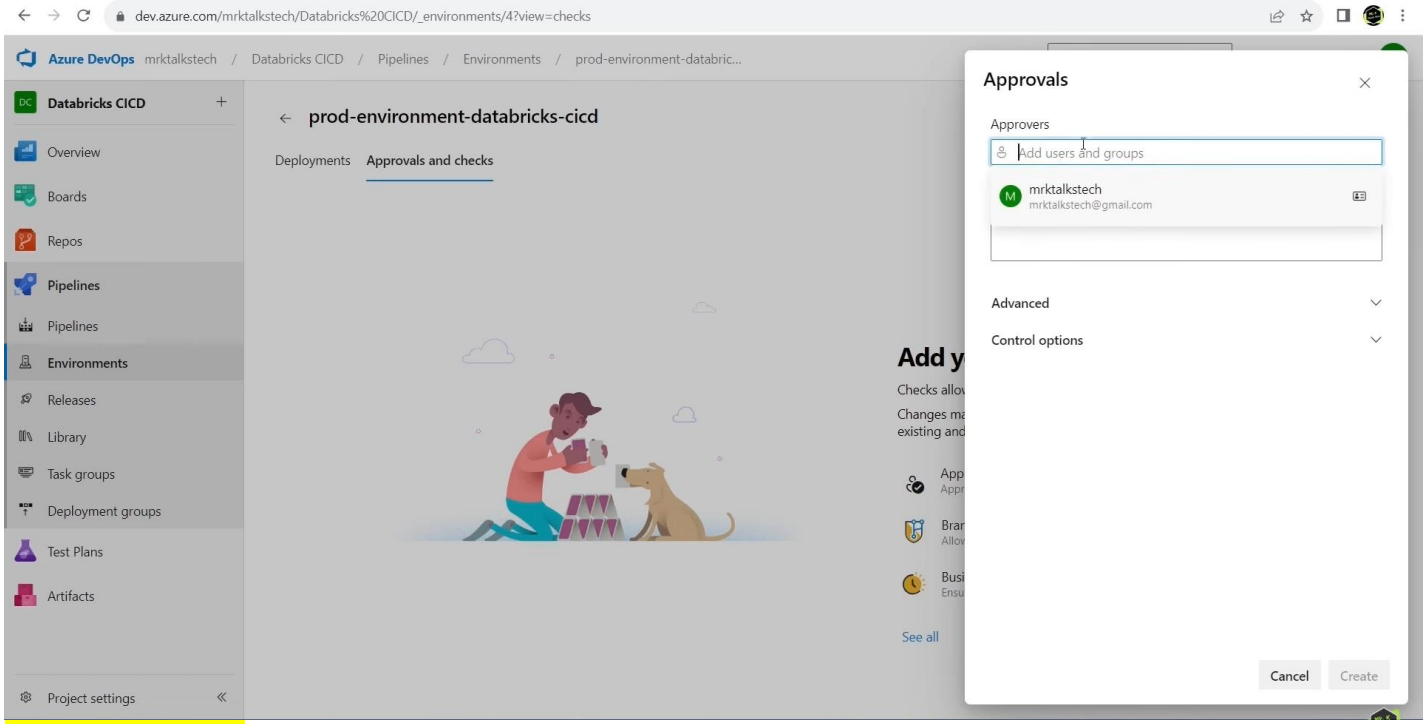
```
trigger:
        - main
variables:
        - group: dbw-cicd-dev
        - group: dbw-cicd-prod
        - name:vmImageName
          value:"windows-latest"
        - name:notebookspath #files notebook path dev to prod
          value:"notebook"
pool:
        vmImage: $(vmImagename #compute power
stages:
        - template: templates/deploy-notebooks.yml
          parameters:
                dependson: ["Deploy_to_Dev_Environment"]
                stageId: "Deploy_to_Dev_Environment"
                env: "dev"
                environmentName: $(dev-environment-name)
                resourceGroupName: $(dev-resource-group-name)
                serviceconnection: $(dev-service-connection-name)
                notebookspath: $(notebookspath)
        - template: templates/deploy-notebooks.yml
          parameters:
                stageId: "Deploy_to_Prod_Environment"
                env: "prod"
                environmentName: $(prod-environment-name)
                resourceGroupName: $(prod-resource-group-name)
                serviceconnection: $(prod-service-connection-name)
                notebookspath: $(notebookspath)
```

goto repos--->create pull request--->files--->create--->approve--->complete merge.

10. Prod Environment Protection and End to End CICD Pipeline Testing

Deploy to prod shouldn’t happen immediately.

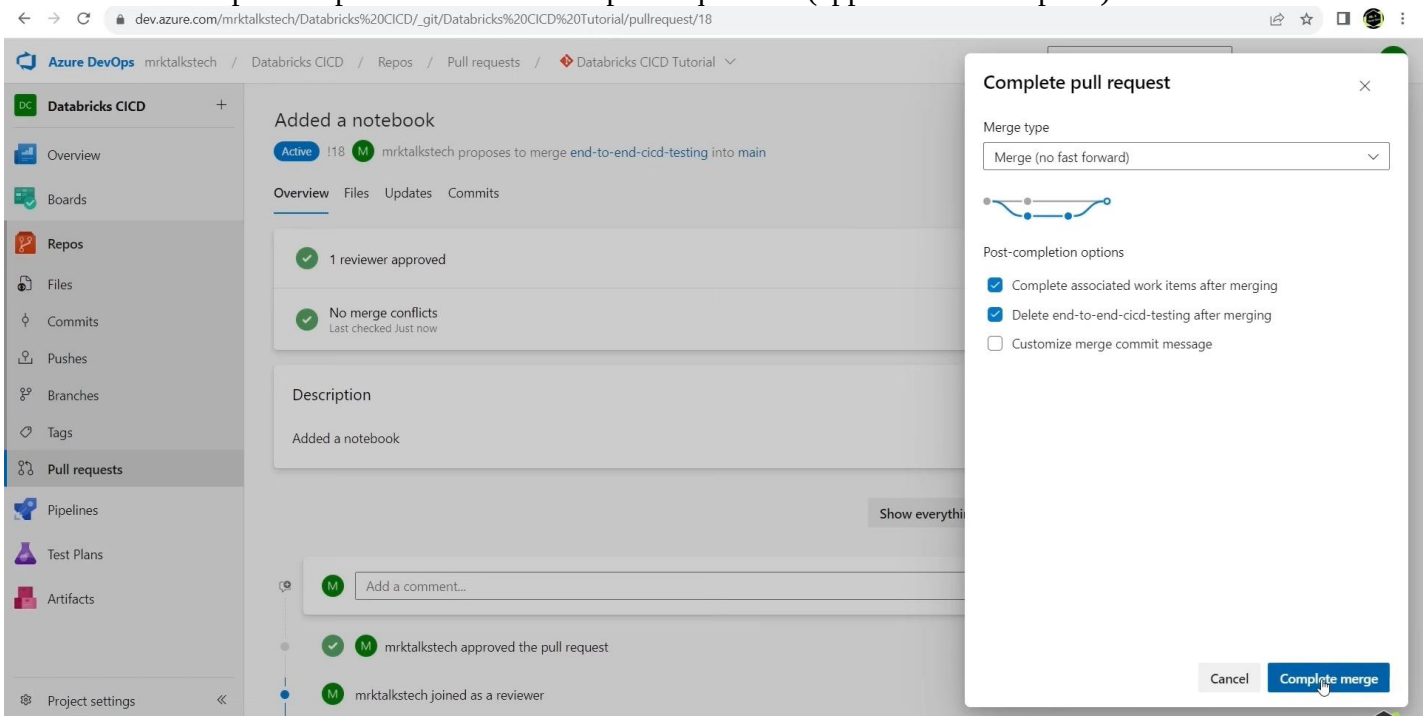
Goto environment--->prod--->approval and checks--->+add checks(approvals--->TL--->create)



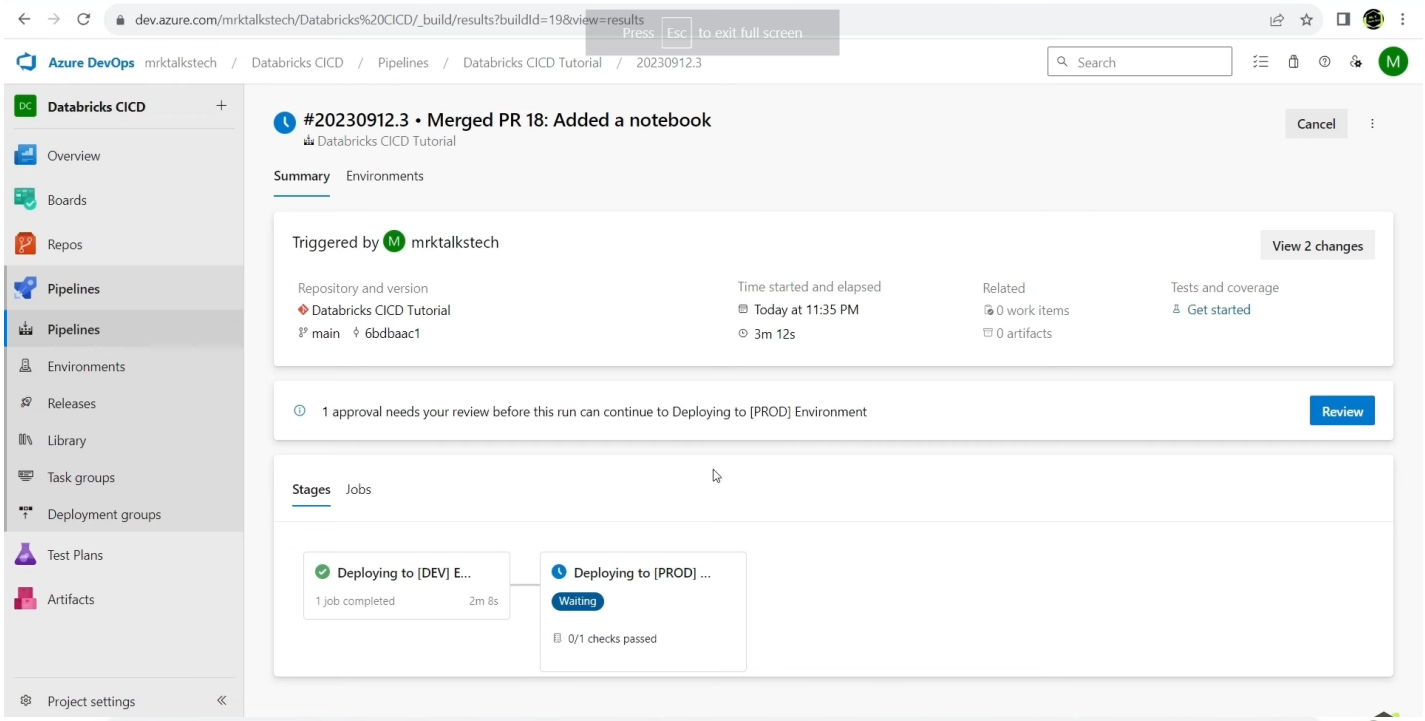
We have protected

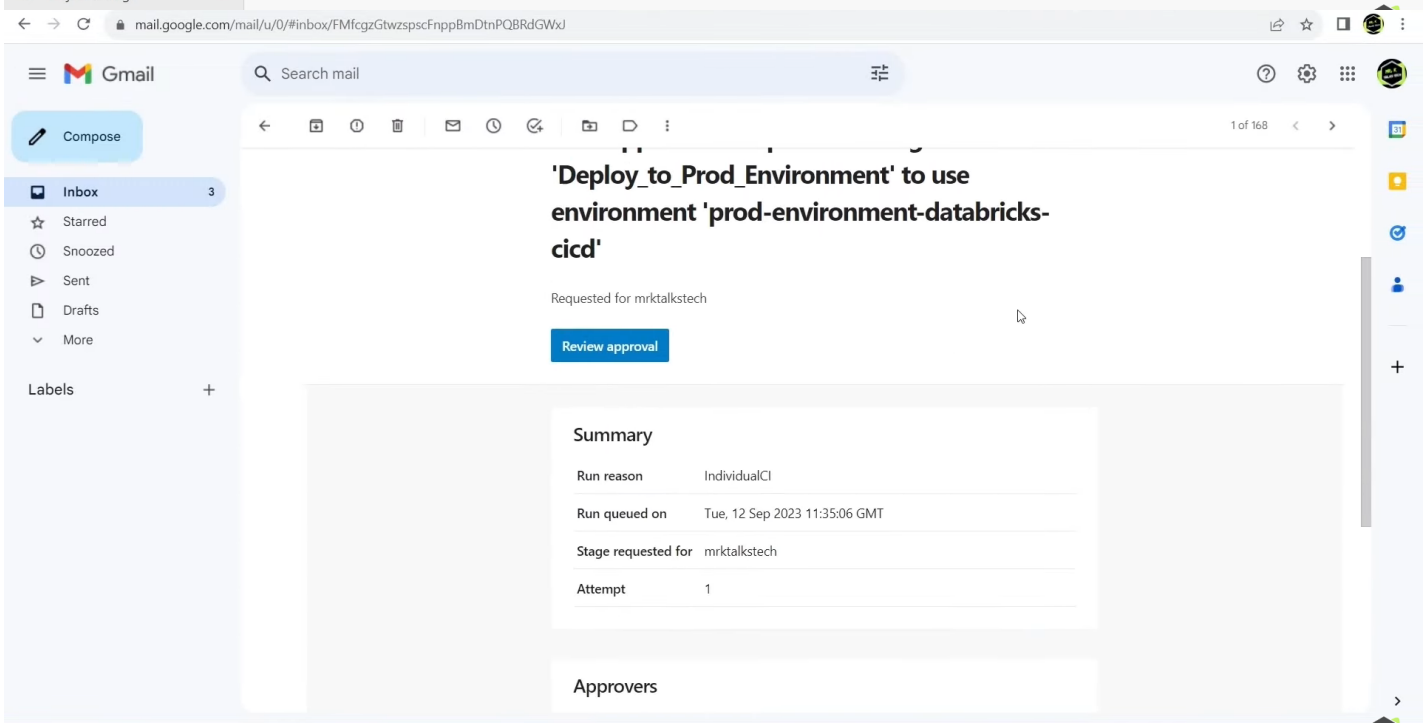
Goto databricks--->repos--->notebook--->testing-ci-pipeline (switch main branch)--->create a new branch(end-t-to-end)--->create new notebook--->commit &push(feature-1 Branch)

Goto Azure Devops--->repos--->click a create pull request--->(approve and complete)



Now merging is done--->new notebook is added(merging) in main branch.
Goto pipeline


dev.azure.com/mrktalkstech/Databricks%20CICD/_build/results?buildId=19&view=results
Press Esc to exit full screen
Azure DevOps mrktalkstech / Databricks CICD / Pipelines / Databricks CICD Tutorial / 20230912.3
Search
Databricks CICD
Overview
Boards
Repos
Pipelines
Pipelines
Environments
Releases
Library
Task groups
Deployment groups
Test Plans
Artifacts
Project settings
#20230912.3 • Merged PR 18: Added a notebook
Databricks CICD Tutorial
Cancel
Summary
Environments
Triggered by mrktalkstech
View 2 changes
Repository and version
Databricks CICD Tutorial
main 6bdbaac1
Time started and elapsed
Today at 11:35 PM
3m 12s
Related
0 work items
0 artifacts
Tests and coverage
Get started
1 approval needs your review before this run can continue to Deploying to [PROD] Environment
Review
Stages
Jobs
Deploying to [DEV] E...
1 job completed
2m 8s
Deploying to [PROD] ...
Waiting
0/1 checks passed


mail.google.com/mail/u/0/#inbox/FMfcgzGtwzspscFnppBmDtnPQBRdGWxJ
Gmail
Search mail
Compose
Inbox 3
Starred
Snoozed
Sent
Drafts
More
Labels
1 of 168
'Deploy_to_Prod_Environment' to use environment 'prod-environment-databricks-cicd'
Requested for mrktalkstech
Review approval
Summary
Run reason IndividualCI
Run queued on Tue, 12 Sep 2023 11:35:06 GMT
Stage requested for mrktalkstech
Attempt 1
Approvers