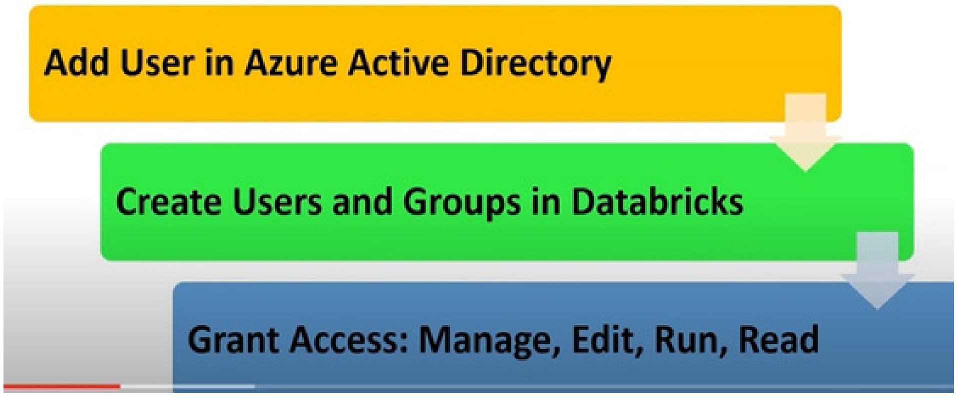
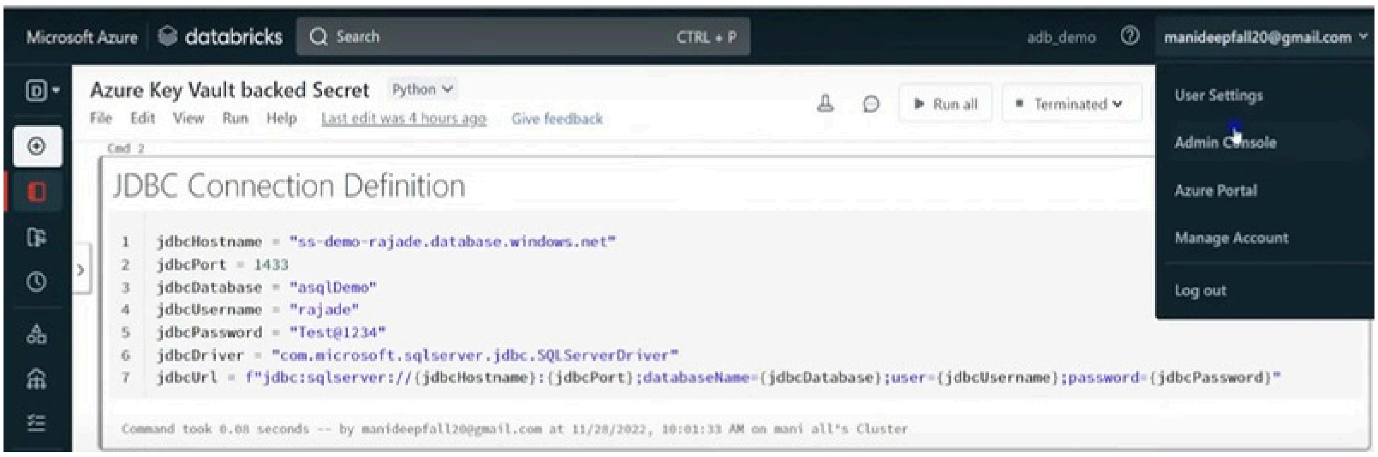
PYSPARK

Workspace object Access Control


Add User in Azure Active Directory
Create Users and Groups in Databricks
Grant Access: Manage, Edit, Run, Read

Steps :

- We need to create users and groups.So log in to azure data bricks -> go to the admin console . There we can add a user

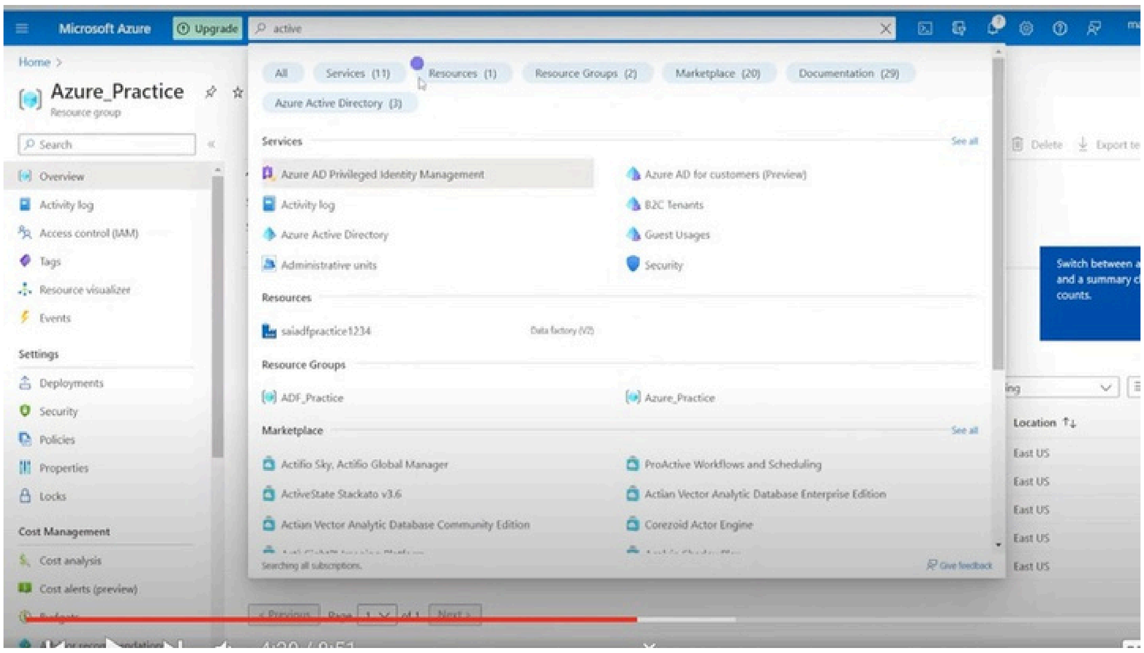


JDBC Connection Definition

```
jdbcHostname = "ss-demo-rajade.database.windows.net"
jdbcPort = 1433
jdbcDatabase = "asqlDemo"
jdbcUsername = "rajade"
jdbcPassword = "Test@1234"
jdbcDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbcUrl = f"jdbc:sqlserver://{jdbcHostname}:{jdbcPort};databaseName={jdbcDatabase};user={jdbcUsername};password={jdbcPassword}"
```

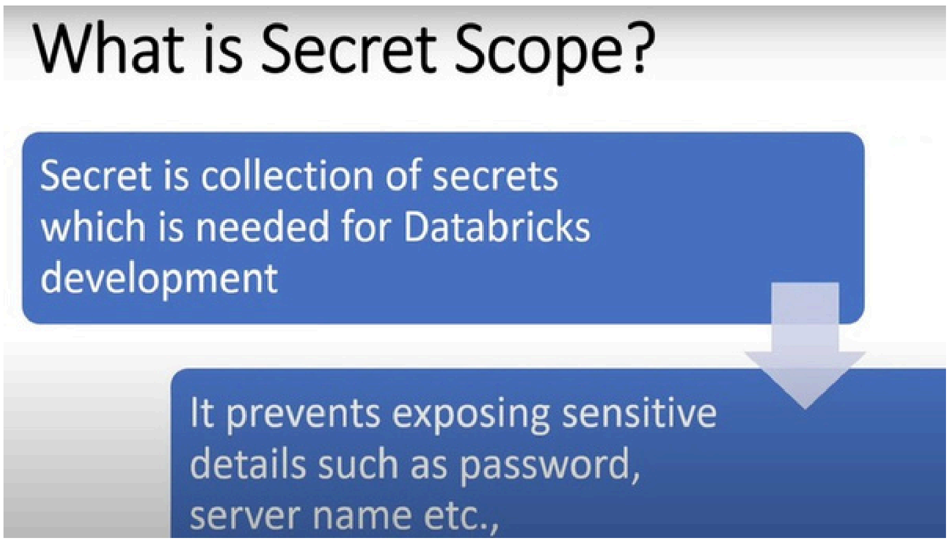
```
Command took 0.08 seconds -- by manideepfall20@gmail.com at 11/28/2022, 10:01:33 AM on mani all's Cluster
```

- No go to azure active directory and add that same user that we added to databricks



- We need to create a group .So log in to azure data bricks -> go to the admin console . There we create a group . After creating a group we can add that user we have created already.
- Now inside databricks , goto workspace -> Users. Inside that we can create folders and put any types of files inside those folders .If we want to restrict access for those folders or files , we can click that dropdown menu that appears right infront of that folder or file , goto permisssions and add or remove users accordingly.

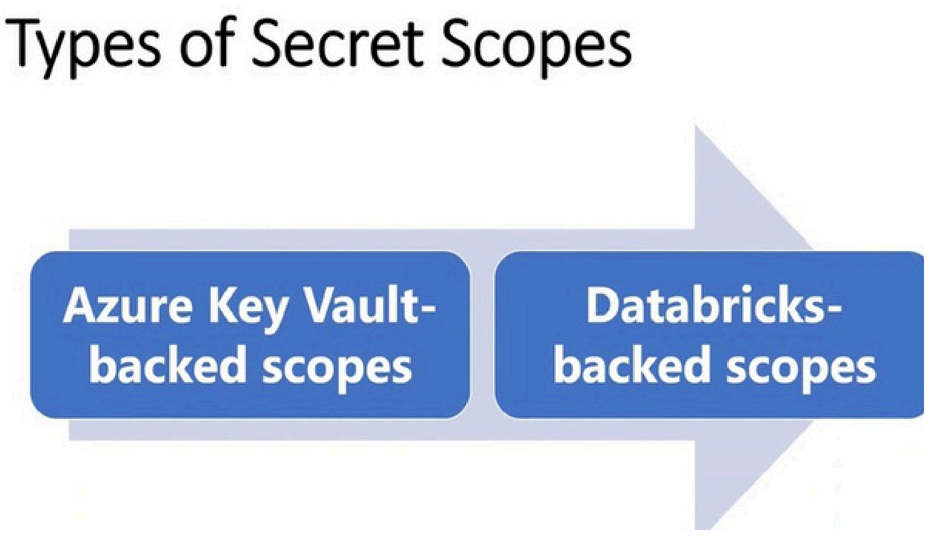
Secret Scopes : Azure Key Vault



What is Secret Scope?

Secret is collection of secrets which is needed for Databricks development

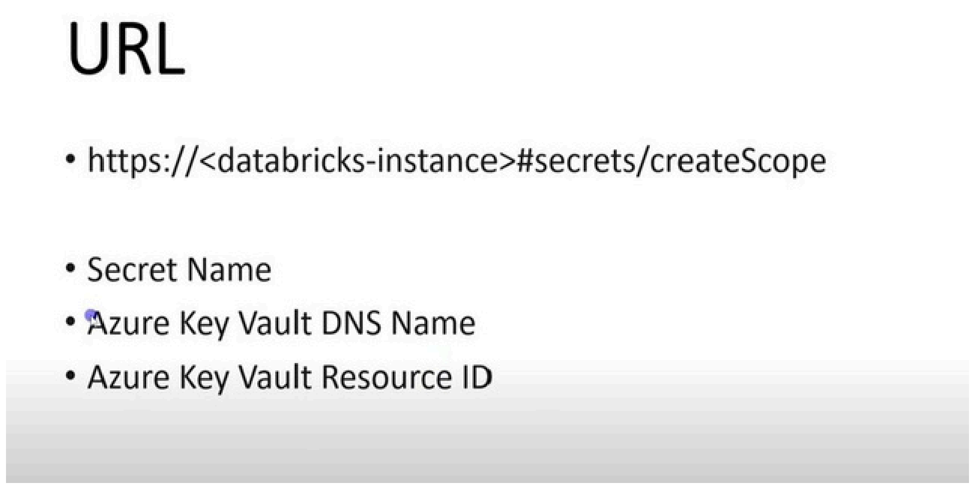
It prevents exposing sensitive details such as password, server name etc.,



Types of Secret Scopes

Azure Key Vault-backed scopes

Databricks-backed scopes



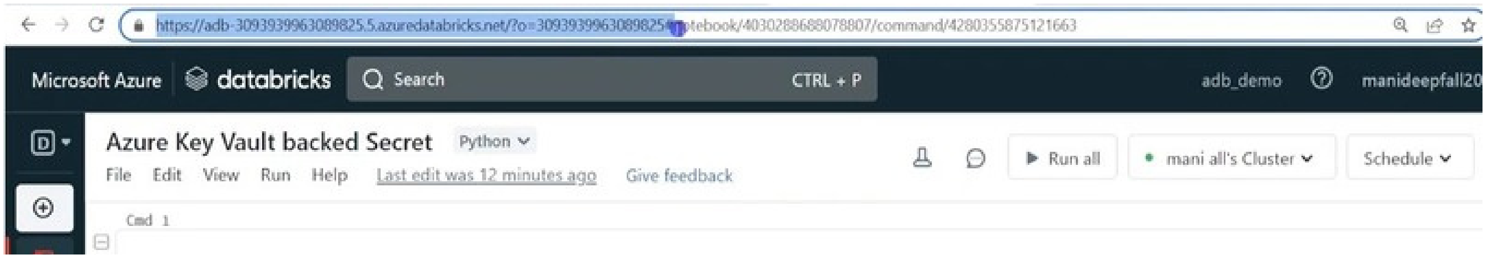
URL

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

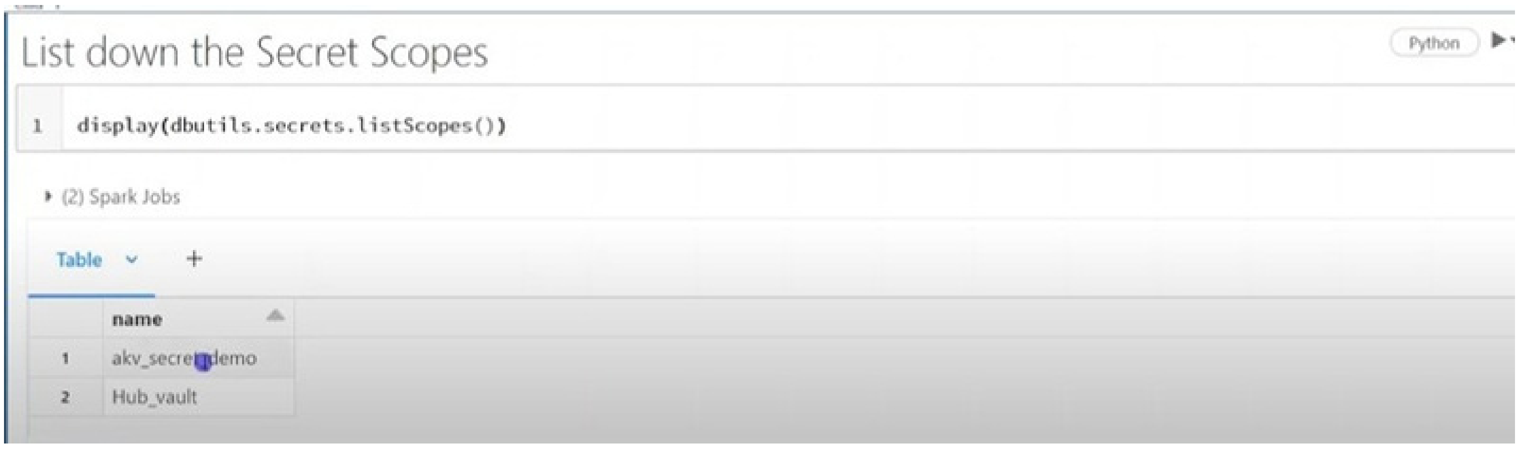
- Secret Name
- Azure Key Vault DNS Name
- Azure Key Vault Resource ID

Procedure :

- First go to azure console and go to Key vault and create a one .Inside that , first we have to click on generate and then we can store all the secrets as key and vlue pairs where the value is our secret . Then go inside that and go to 'properties' which is under settings in left pane . Inside that we can see the 'Vault URI' and 'Resource ID' , copy them .
- Now go to azure console -> Access policies -> Create -> now put a tick on select all under 'Key permissions and 'Secret permission' and click on 'next' -> in that search box type Azure databricks and then select that .
- Now go to that URL. In that URL , <databricks-instance> is ;


https://adb-3093939963089825.5.azuredatabricks.net/?o=3093939963089825#notebook/4030288688078807/command/4280355875121663

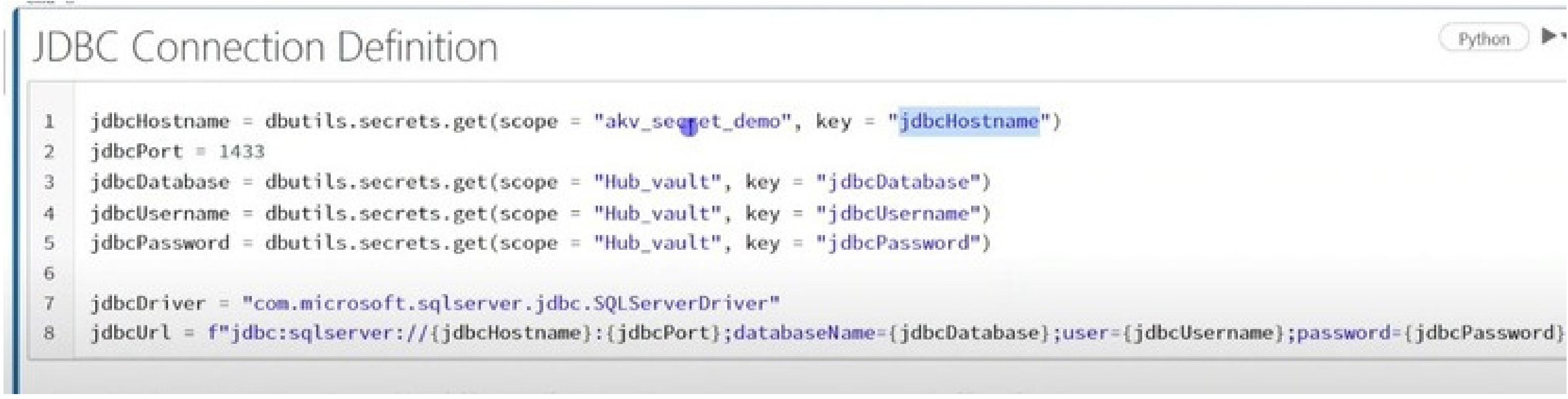
- Now give a name for "Scope Name" and paste the copied Vault URI and Resource ID
- Now go back to databricks ;


List down the Secret Scopes

```
display(dbutils.secrets.listScopes())
```

	name
1	akv_secret_demo
2	Hub_vault

(Here akv_secret_demo is the name of our Scope that we have created)



JDBC Connection Definition

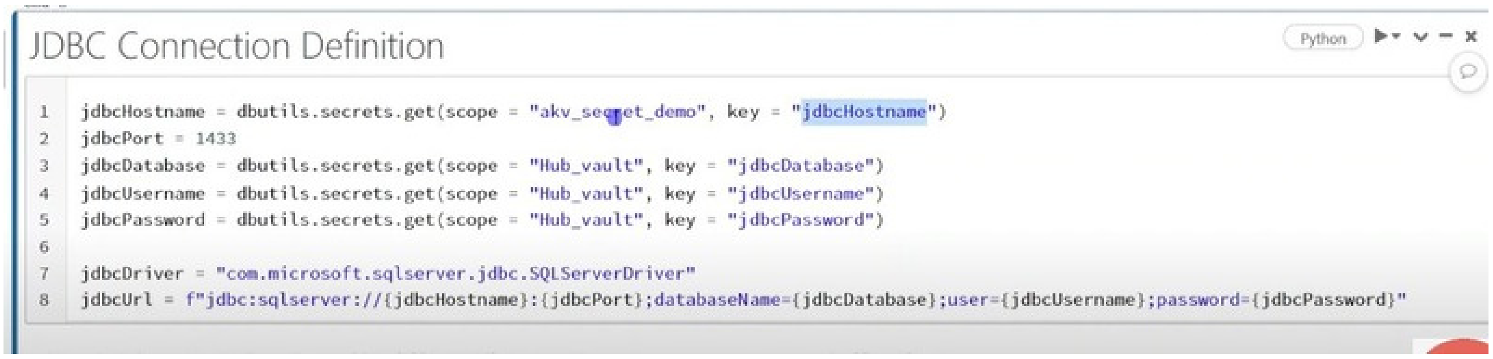
```python
jdbcHostname = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcHostname")
jdbcPort = 1433
jdbcDatabase = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcDatabase")
jdbcUsername = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcUsername")
jdbcPassword = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcPassword")

jdbcDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbcUrl = f"jdbc:sqlserver://{jdbcHostname}:{jdbcPort};databaseName={jdbcDatabase};user={jdbcUsername};password={jdbcPassword}
```

- As shown in the above figure we can access our secrets .There ‘key’ is the name of the key that we have given when we stored the secrets inside azure key vault

Databricks Workflows: Job Scheduling

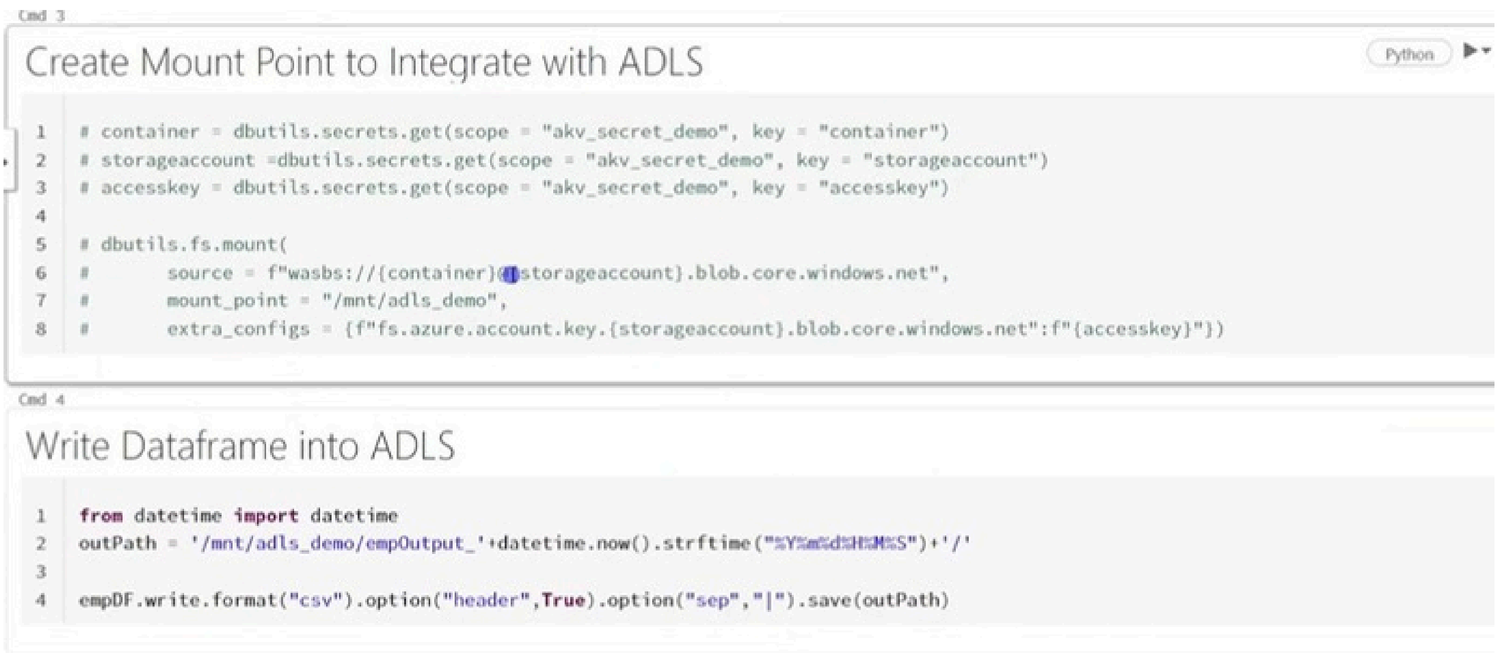
- Workflow is nothing but create jobs and scheduling them in regular intervals.Once we created pipelines we need to schedule them.For that we use jobs.In databricks this concept is called as workflows.Within workflows we can create jobs.Jobs is nothing but a collection of jobs .Task is nothing but running a one specific bsiness process.Task might execute a one single notebook or a python file .
- It is possible to create multiple tasks and create a dependency between them.
- Lets consider a demo where we are pulling data from a azure sql db and writing it into a azure data lake storage.



JDBC Connection Definition

```python
jdbcHostname = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcHostname")
jdbcPort = 1433
jdbcDatabase = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcDatabase")
jdbcUsername = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcUsername")
jdbcPassword = dbutils.secrets.get(scope = "Hub_vault", key = "jdbcPassword")

jdbcDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbcUrl = f"jdbc:sqlserver://{jdbcHostname}:{jdbcPort};databaseName={jdbcDatabase};user={jdbcUsername};password={jdbcPassword}"
```


Cmd 3

Create Mount Point to Integrate with ADLS

Python

```
# container = dbutils.secrets.get(scope = "akv_secret_demo", key = "container")
# storageaccount =dbutils.secrets.get(scope = "akv_secret_demo", key = "storageaccount")
# accesskey = dbutils.secrets.get(scope = "akv_secret_demo", key = "accesskey")

# dbutils.fs.mount(
#       source = f"wasbs://{container}@{storageaccount}.blob.core.windows.net",
#       mount_point = "/mnt/adls_demo",
#       extra_configs = {f"fs.azure.account.key.{storageaccount}.blob.core.windows.net":f"{accesskey}"})
```

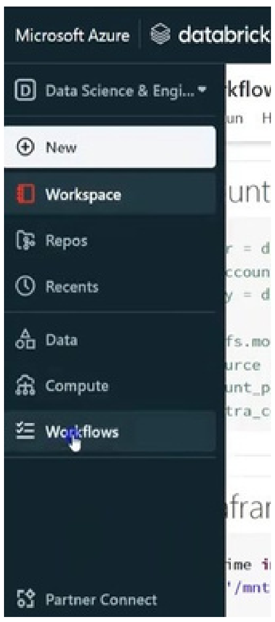
Cmd 4

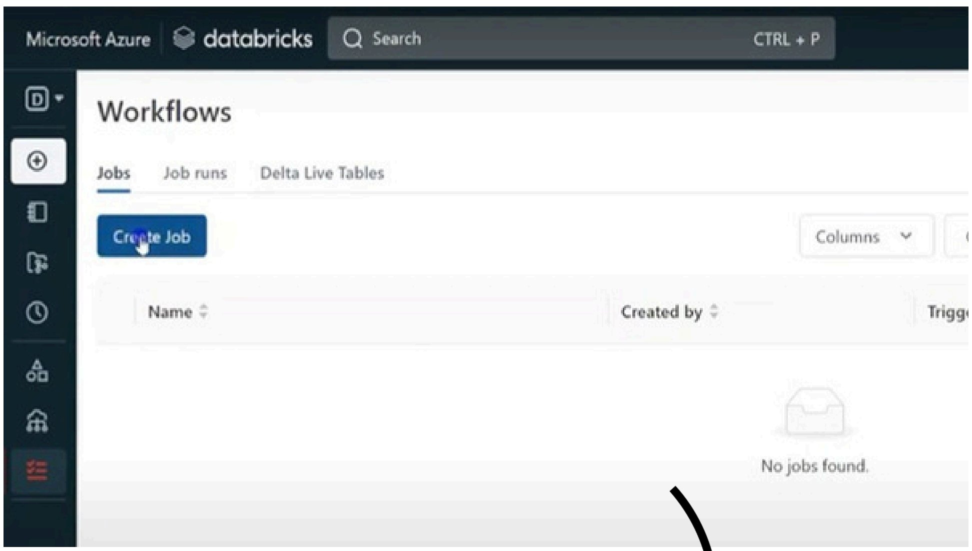
Write Dataframe into ADLS

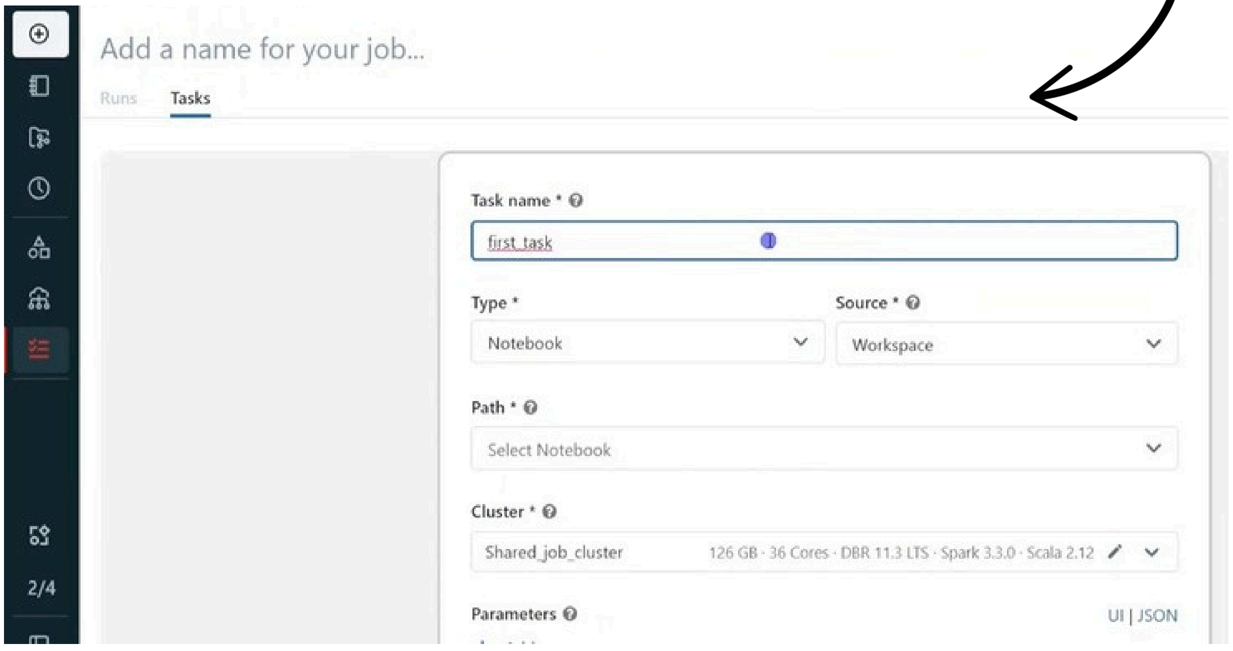
```
from datetime import datetime
outPath = '/mnt/adls_demo/empOutput_'+datetime.now().strftime("%Y%m%d%H%M%S")+'/'

empDF.write.format("csv").option("header",True).option("sep","|").save(outPath)
```

Procedure:






Add a name for your job...

Runs Tasks

Task name *
first_task

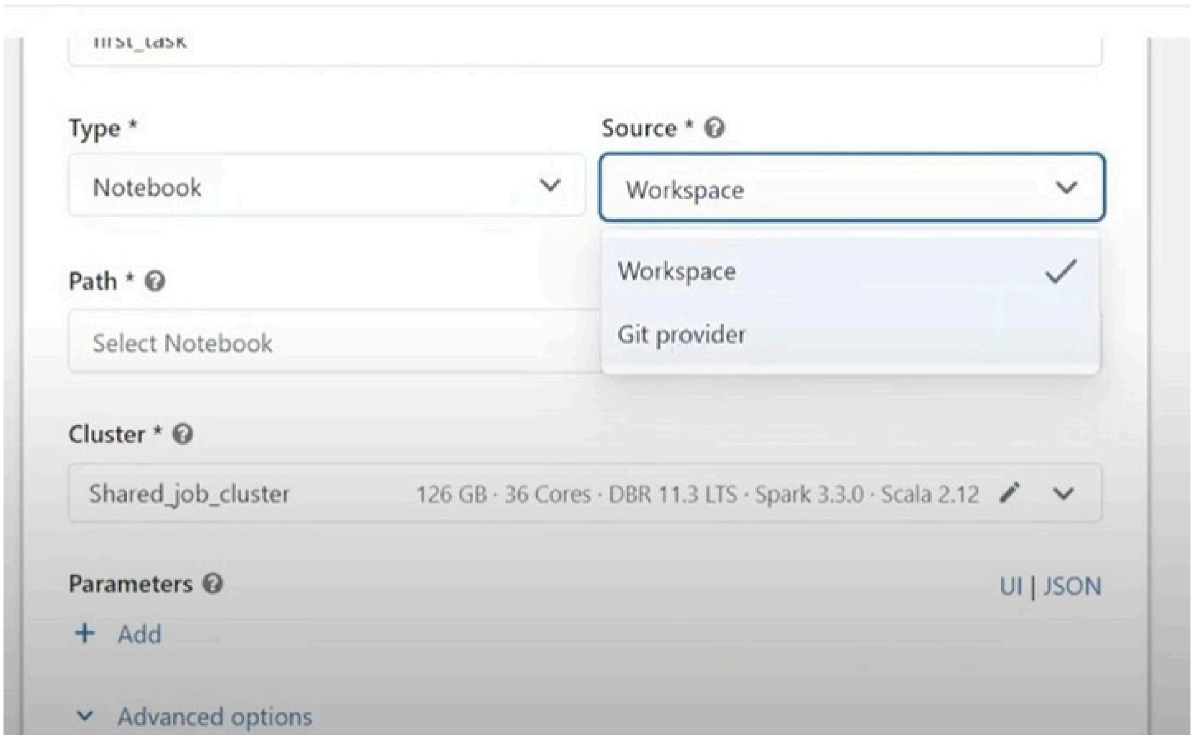
Type *
Notebook

Source *
Workspace

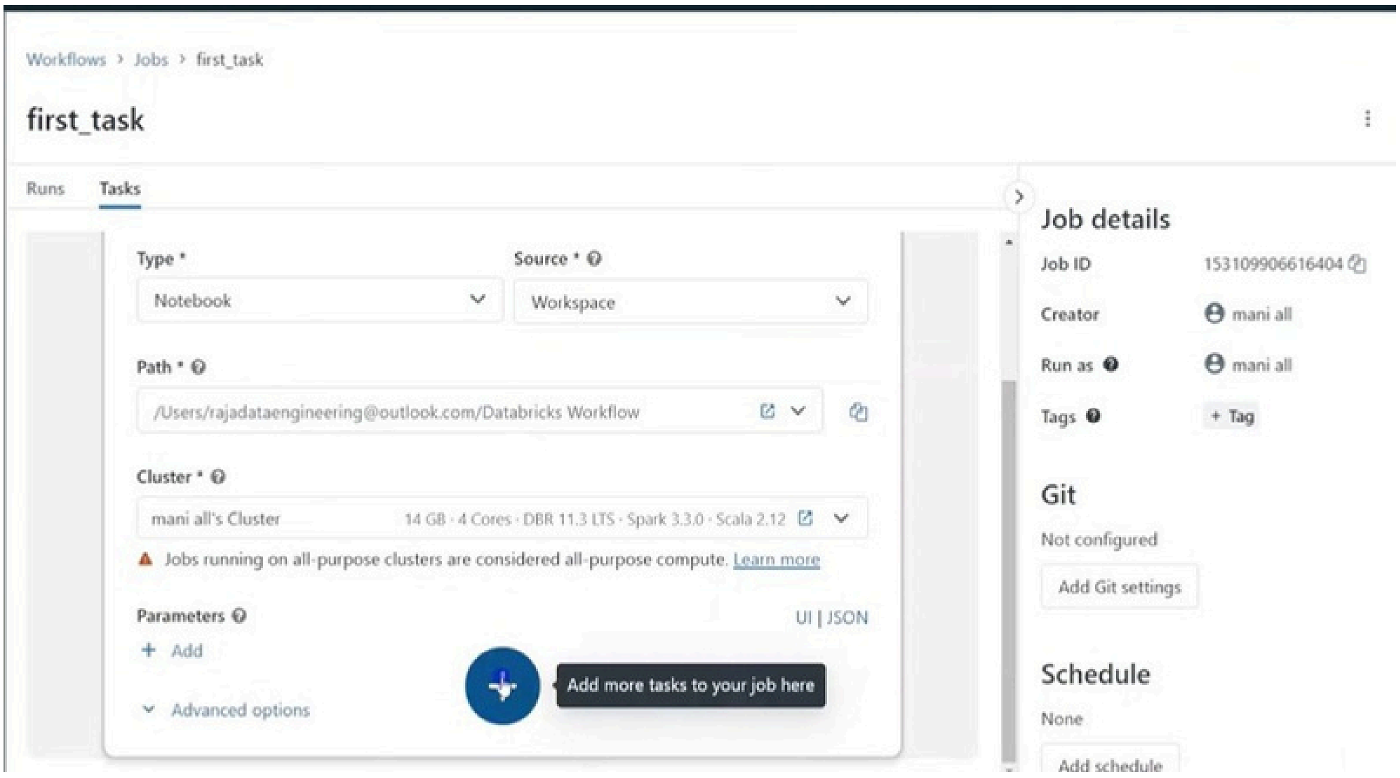
Path *
Select Notebook

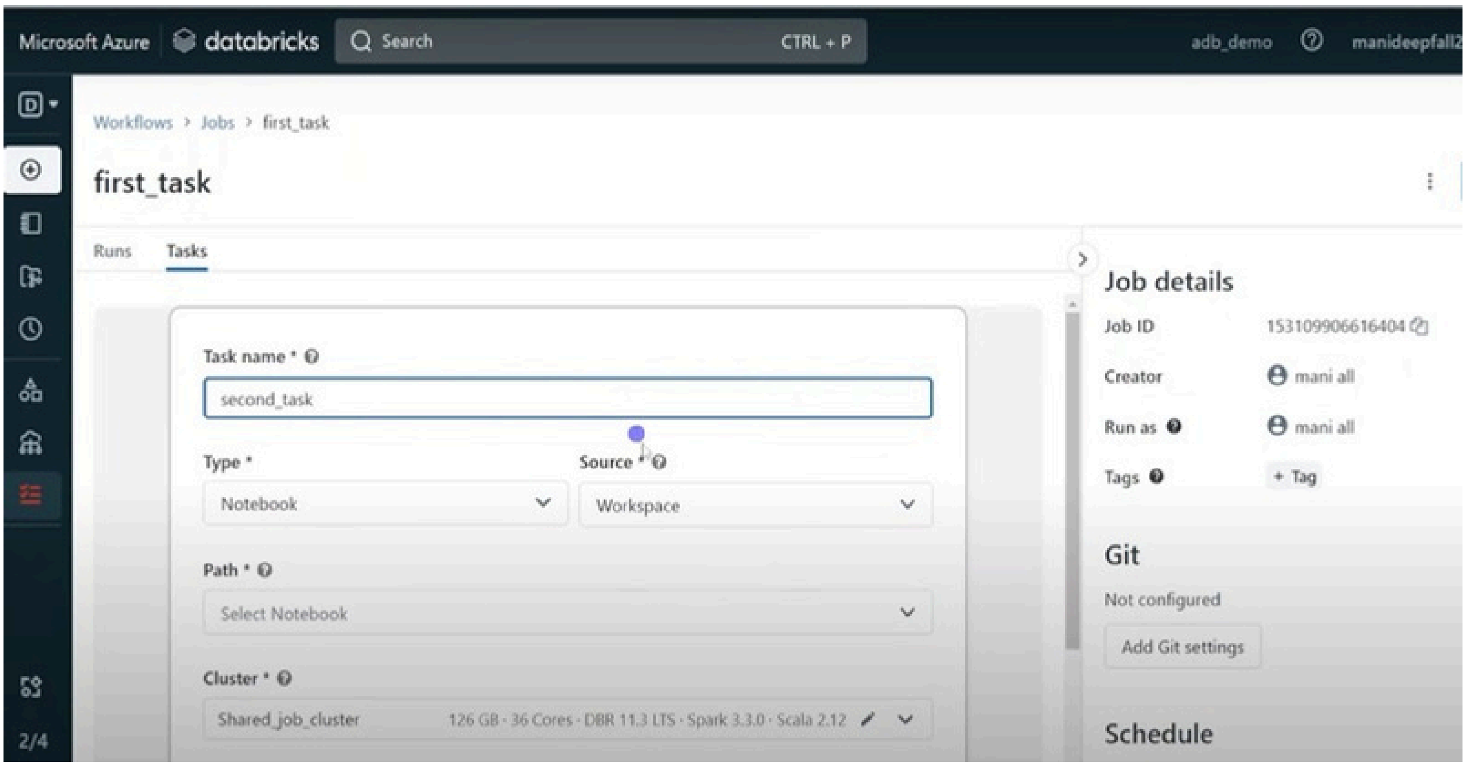
Cluster *
Shared_job_cluster 126 GB · 36 Cores · DBR 11.3 LTS · Spark 3.3.0 · Scala 2.12

Parameters UI | JSON



- Now select the path of the notebook (click on the drop down icon infront of the path -> go to Users and select the user -> then select the notenook .)
- Now click on cluster and select job cluster . Job cluster is a type of a cluster where it is getting created by the time task is initiated and will be terminated when is task is done execution. After that configure the job .
- After that as we can see in the above figure , we can go to advanced options , if we want to install some packages .
- Now click on create
- Here we are having only a single task. But it is possible to add multiple tasks. If we want to create another task , we have to click on the + symbol that appears once we created the first task.


Workflows > Jobs > first_task
first_task
Runs
Tasks
Type *
Notebook
Source *
Workspace
Path *
/Users/rajadataengineering@outlook.com/Databricks Workflow
Cluster *
mani all's Cluster
14 GB · 4 Cores · DBR 11.3 LTS · Spark 3.3.0 · Scala 2.12
Jobs running on all-purpose clusters are considered all-purpose compute. Learn more
Parameters
UI | JSON
+ Add
Advanced options
Add more tasks to your job here
Job details
Job ID
153109906616404
Creator
mani all
Run as
mani all
Tags
+ Tag
Git
Not configured
Add Git settings
Schedule
None
Add schedule



Workflows > Jobs > first_task

first_task

Runs | Tasks

Task name *: second_task

Type *: Notebook

Source *: Workspace

Path *: Select Notebook

Cluster *: Shared_job_cluster 126 GB · 36 Cores · DBR 11.3 LTS · Spark 3.3.0 · Scala 2.12

Job details

Job ID: 153109906616404

Creator: mani all

Run as: mani all

Tags: + Tag

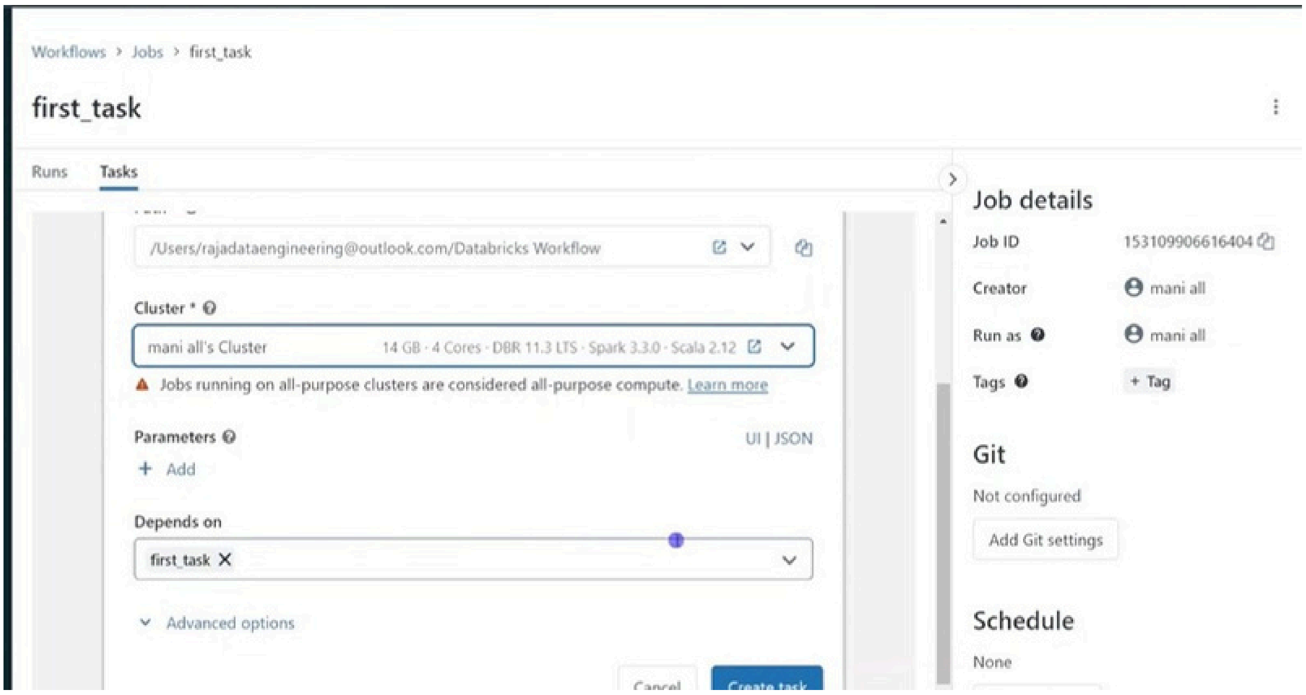
Git

Not configured

Add Git settings

Schedule

- As we configure the first task , configure this as well. But here we have a new section called Depends on ,



Workflows > Jobs > first_task

first_task

Runs | Tasks

/Users/rajadataengineering@outlook.com/Databricks Workflow

Cluster *: mani all's Cluster 14 GB · 4 Cores · DBR 11.3 LTS · Spark 3.3.0 · Scala 2.12

Jobs running on all-purpose clusters are considered all-purpose compute. Learn more

Parameters: + Add (UI | JSON)

Depends on: first_task

Advanced options

Cancel | Create task

Job details

Job ID: 153109906616404

Creator: mani all

Run as: mani all

Tags: + Tag

Git

Not configured

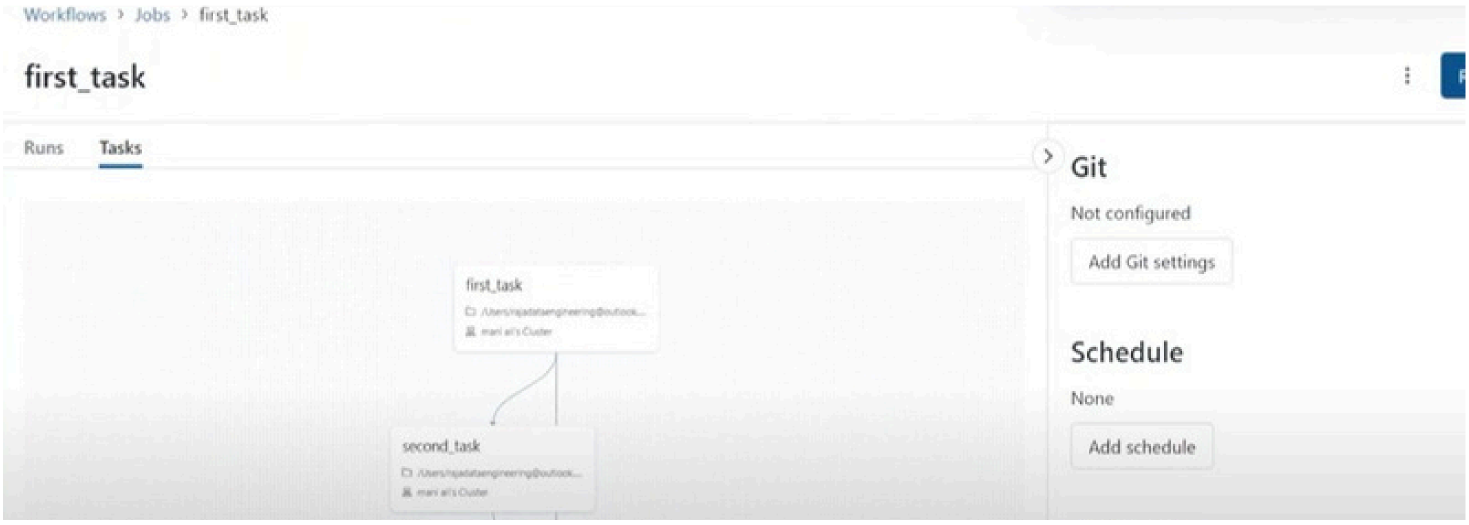
Add Git settings

Schedule

None

- If we say , the second task depends on the first task, then if the first task is fail to execute , then this second task also wont be executed.

- Once we hace created the tasks , now click on 'Add schedule '


Workflows › Jobs › first_task

first_task

Runs | Tasks

Git

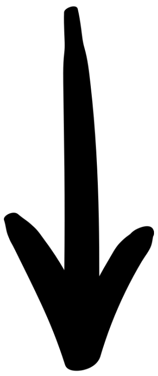
Not configured

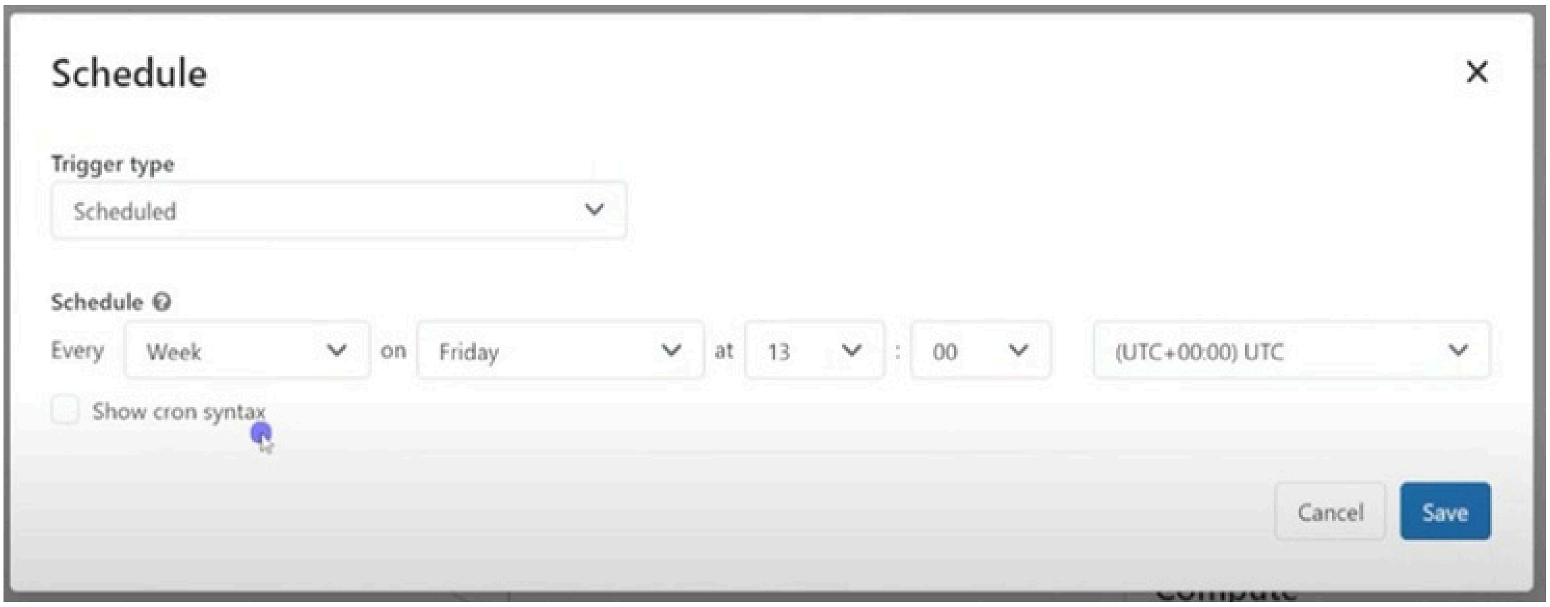
Add Git settings

Schedule

None

Add schedule




Schedule

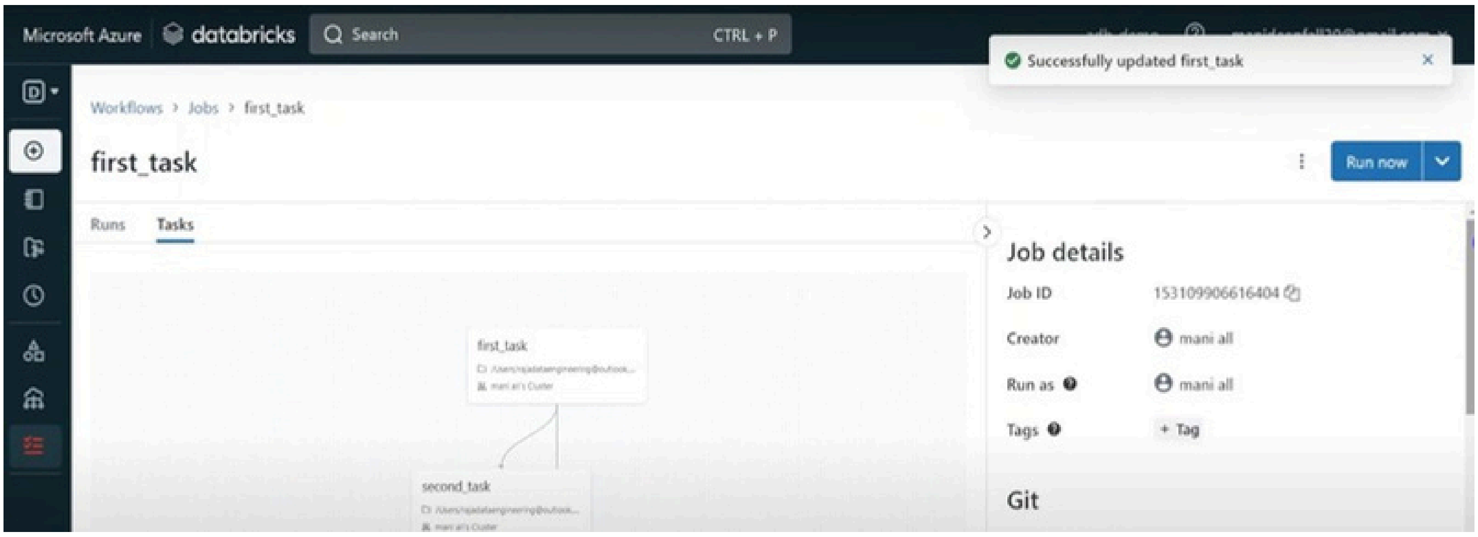
Trigger type: Scheduled

Schedule: Every Week on Friday at 13 : 00 (UTC+00:00) UTC

Show cron syntax

Cancel | Save

- To run it manually , click on ‘Run Now’

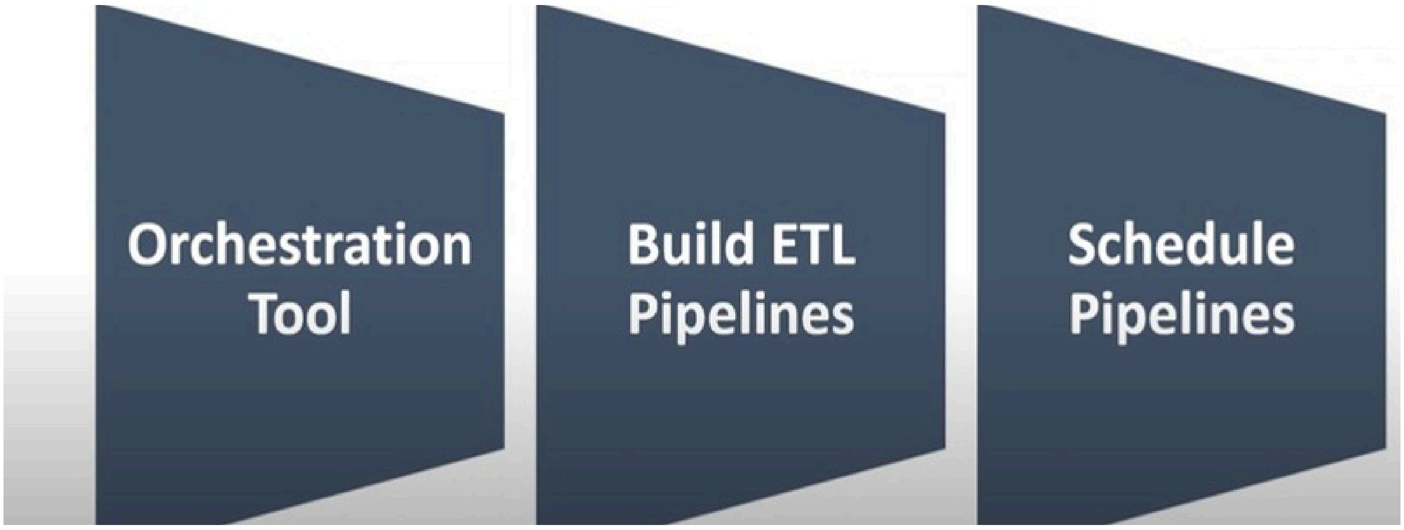


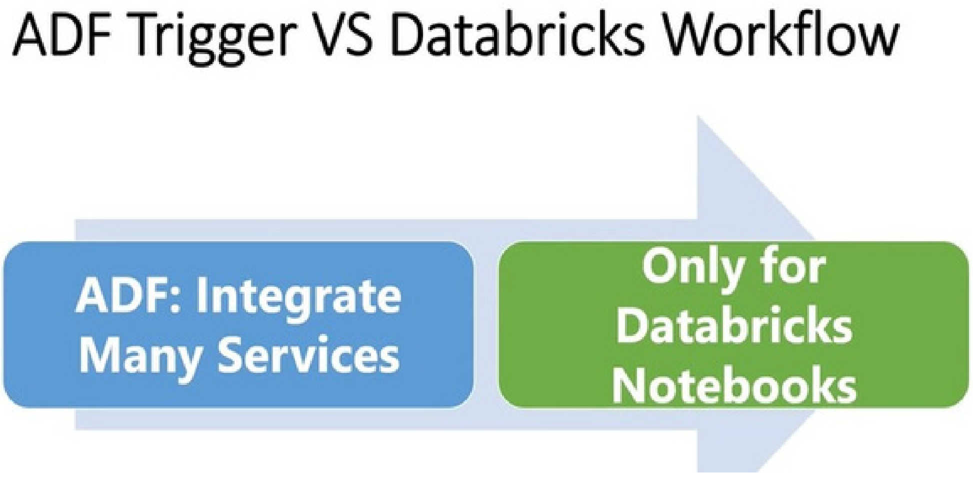
Job details

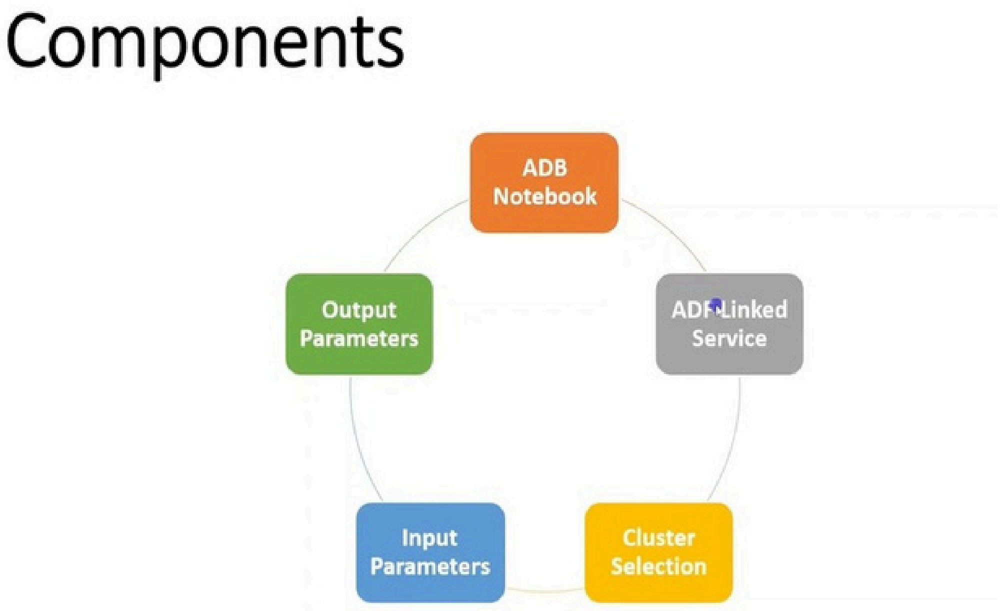
Job ID	153109906616404
Creator	mani all
Run as	mani all
Tags	+ Tag

Azure Data Factory + Azure Databricks

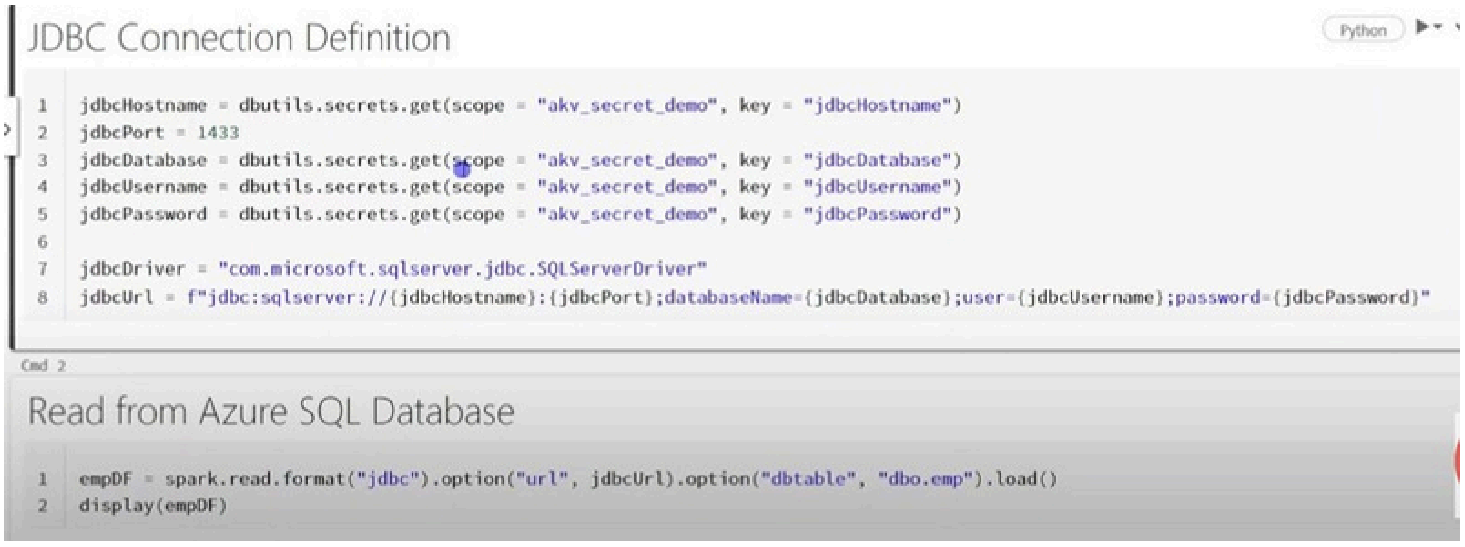
What is ADF




ADF Trigger VS Databricks Workflow
ADF: Integrate Many Services
Only for Databricks Notebooks



- Now for the demo lets consider our previous example where we read the data from a azure sql db and write them in to azure data lake



JDBC Connection Definition

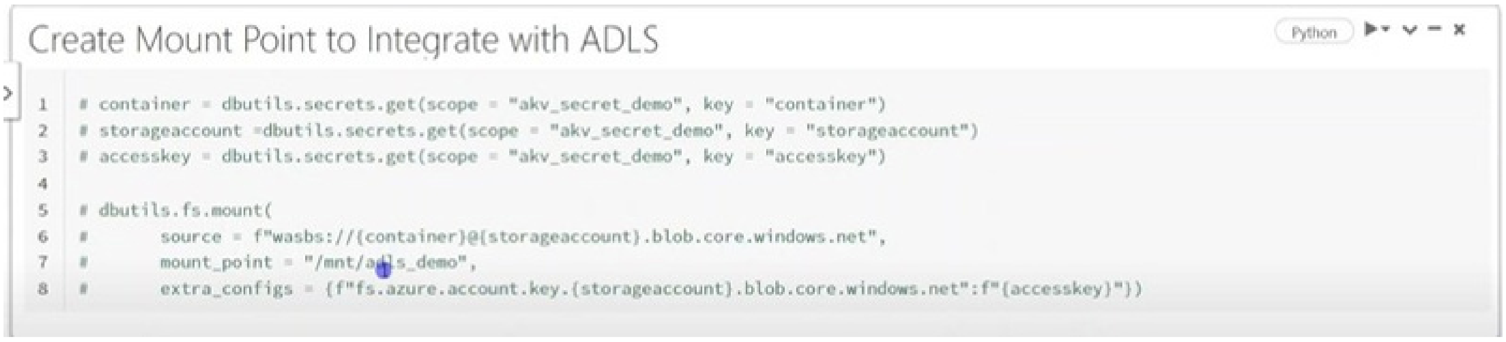
```
jdbcHostname = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcHostname")
jdbcPort = 1433
jdbcDatabase = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcDatabase")
jdbcUsername = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcUsername")
jdbcPassword = dbutils.secrets.get(scope = "akv_secret_demo", key = "jdbcPassword")

jdbcDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver"
jdbcUrl = f"jdbc:sqlserver://{jdbcHostname}:{jdbcPort};databaseName={jdbcDatabase};user={jdbcUsername};password={jdbcPassword}"
```

Cmd 2

Read from Azure SQL Database

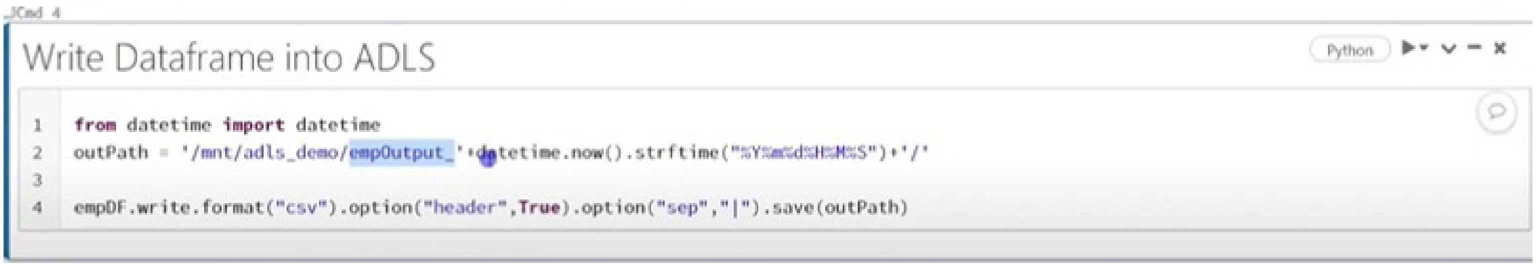
```
empDF = spark.read.format("jdbc").option("url", jdbcUrl).option("dbtable", "dbo.emp").load()
display(empDF)
```



Create Mount Point to Integrate with ADLS

```
# container = dbutils.secrets.get(scope = "akv_secret_demo", key = "container")
# storageaccount =dbutils.secrets.get(scope = "akv_secret_demo", key = "storageaccount")
# accesskey = dbutils.secrets.get(scope = "akv_secret_demo", key = "accesskey")

# dbutils.fs.mount(
#       source = f"wasbs://{container}@{storageaccount}.blob.core.windows.net",
#       mount_point = "/mnt/adls_demo",
#       extra_configs = {f"fs.azure.account.key.{storageaccount}.blob.core.windows.net":f"{accesskey}"})
```



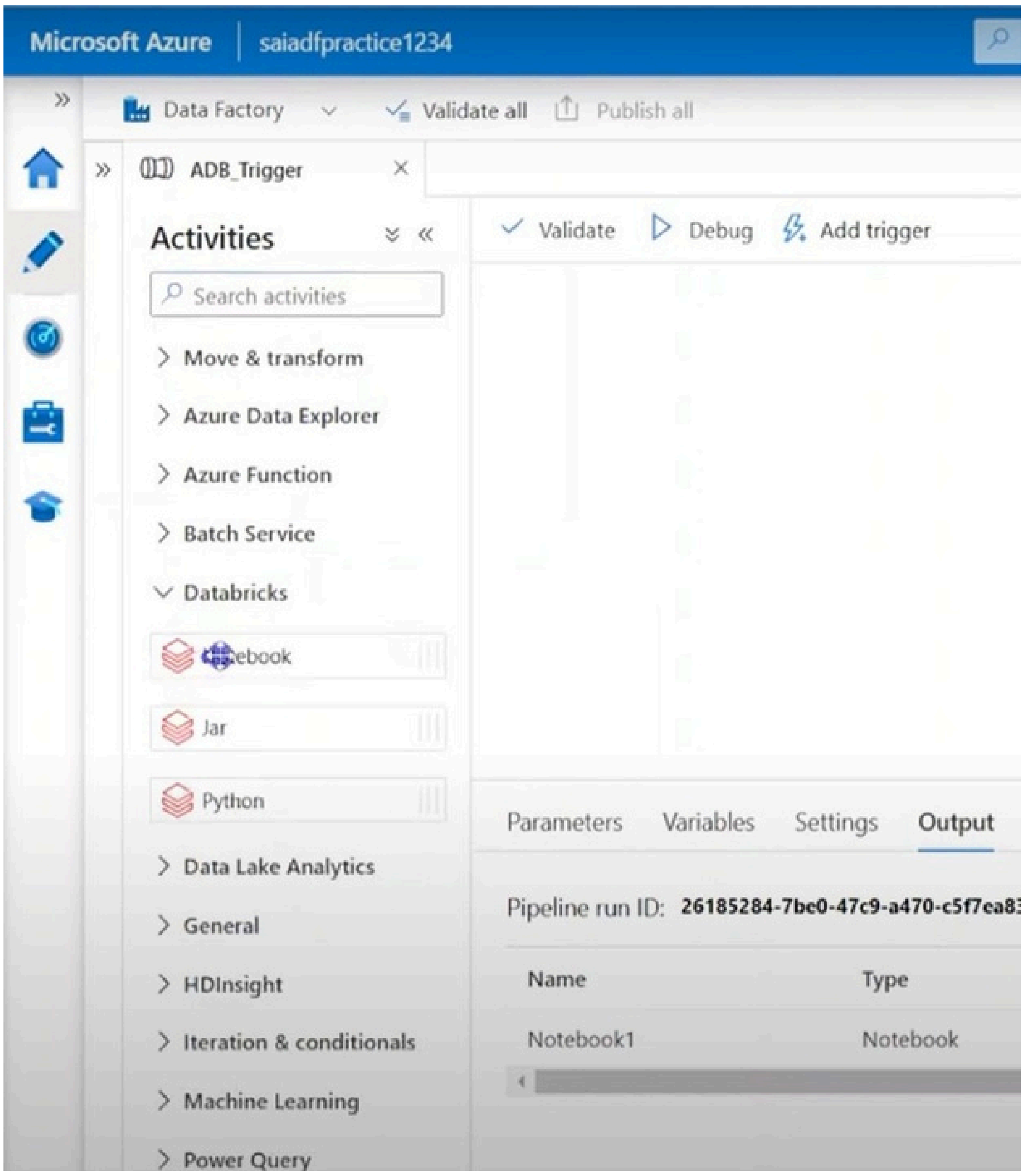
Cmd 4

Write Dataframe into ADLS

```
from datetime import datetime
outPath = '/mnt/adls_demo/empOutput_'+datetime.now().strftime("%Y%m%d%H%M%S")+'/'

empDF.write.format("csv").option("header",True).option("sep","|").save(outPath)
```

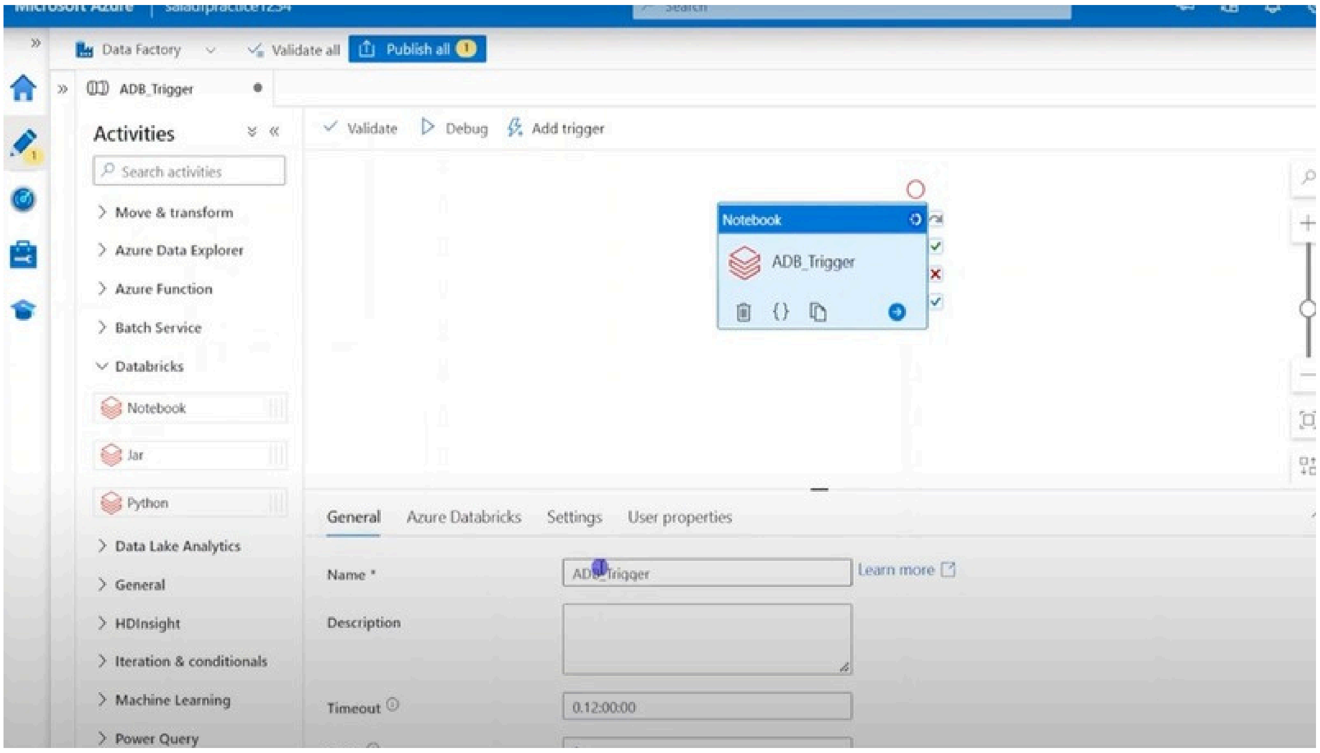
- Now go to the azure console -> Data Factory and select that Notebook option , drag it and drop it in the center



Pipeline run ID: 26185284-7be0-47c9-a470-c5f7ea8

Name	Type
Notebook1	Notebook

- Now under general , just give it a name

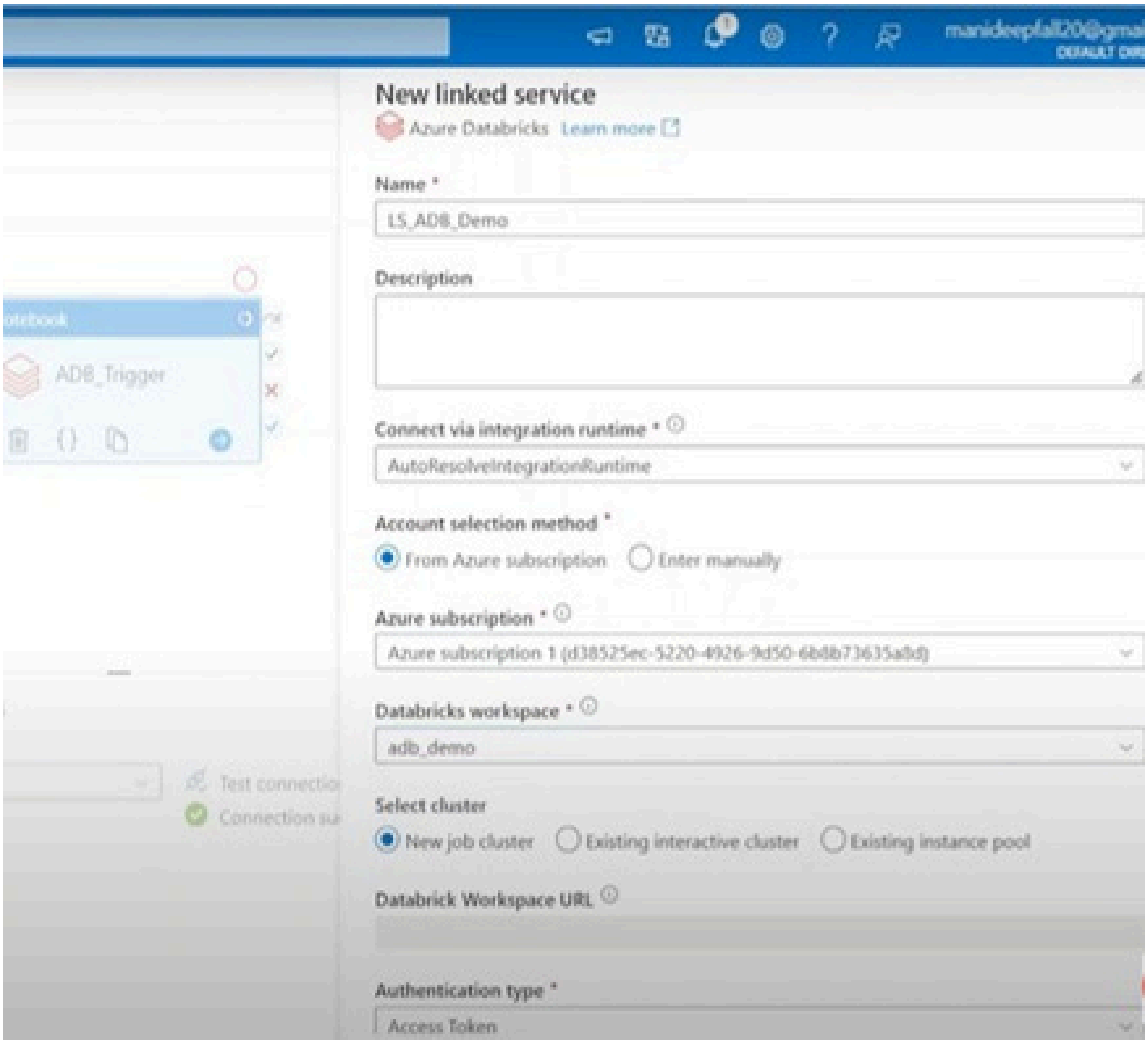


Name * ADB_Trigger

Description

Timeout 0.12:00:00

- Now go to Azure Databrixks , which is right next to general .In there click on '+new'



New linked service

Azure Databricks Learn more

Name *

LS_ADB_Demo

Description

Connect via integration runtime *

AutoResolveIntegrationRuntime

Account selection method *

[✓ From Azure subscription] Enter manually

Azure subscription *

Azure subscription 1 (d38525ec-5220-4926-9d50-6b8b73635a8d)

Databricks workspace *

adb_demo

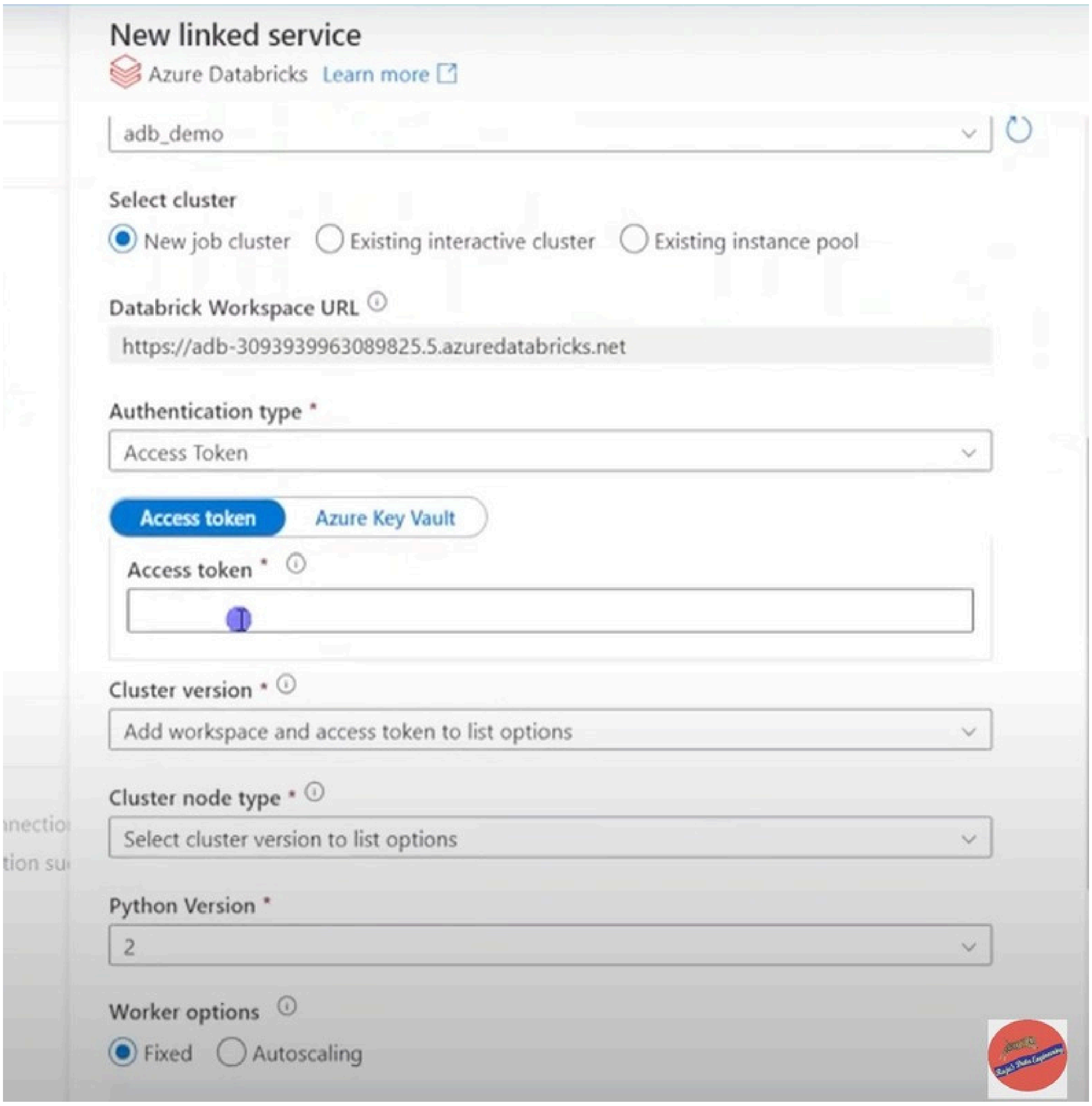
Select cluster

[✓ New job cluster] Existing interactive cluster Existing instance pool

Databrick Workspace URL

Authentication type *

Access Token


New linked service

Azure Databricks Learn more

adb_demo

Select cluster

(● New job cluster) ○ Existing interactive cluster ○ Existing instance pool

Databrick Workspace URL

https://adb-3093939963089825.5.azuredatabricks.net

Authentication type *

Access Token

[✓ Access token] Azure Key Vault

Access token *

Cluster version *

Add workspace and access token to list options

Cluster node type *

Select cluster version to list options

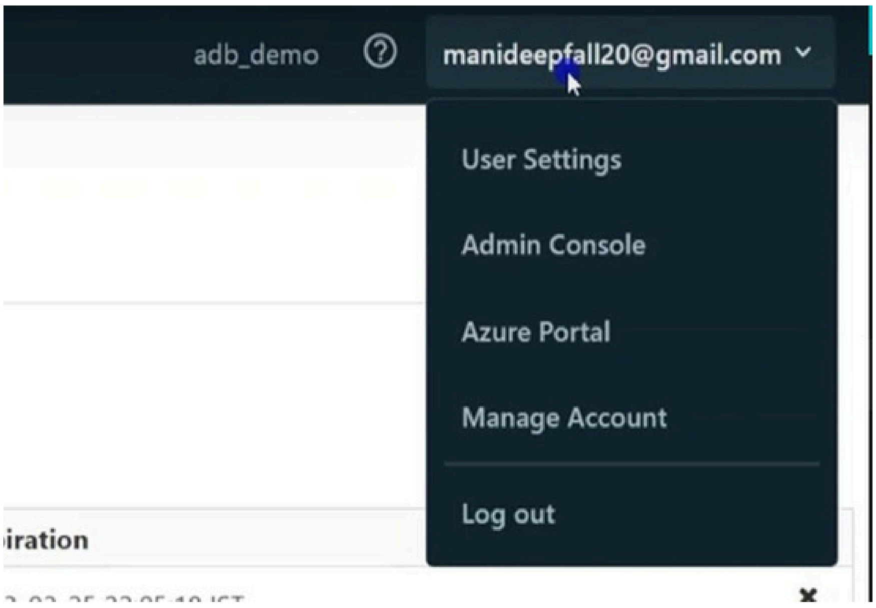
Python Version *

2

Worker options

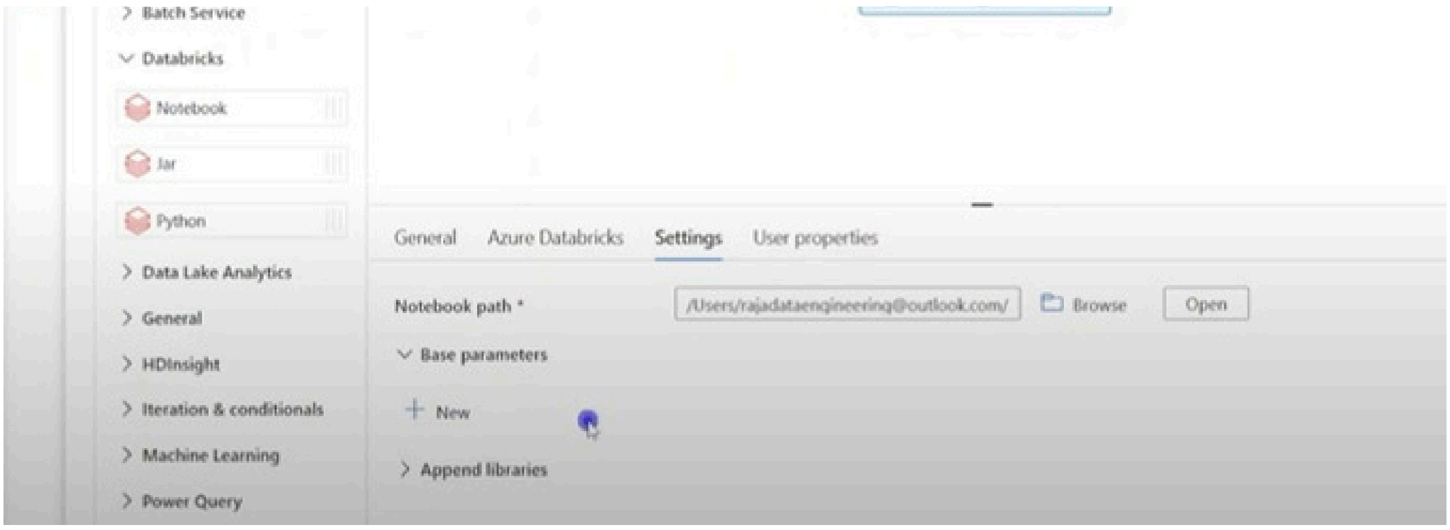
(● Fixed) ○ Autoscaling

- Here we need to provide a access token over here . To get that go back to databricks console ->



- Now go to user settings and generate a new token and also we have to give a meaningful name. Copy the token and paste over there in the azure .

- Now under settings select the notebook by clicking on browse .Over there if we have any input parameters , we have to give them under 'Base parameters'.



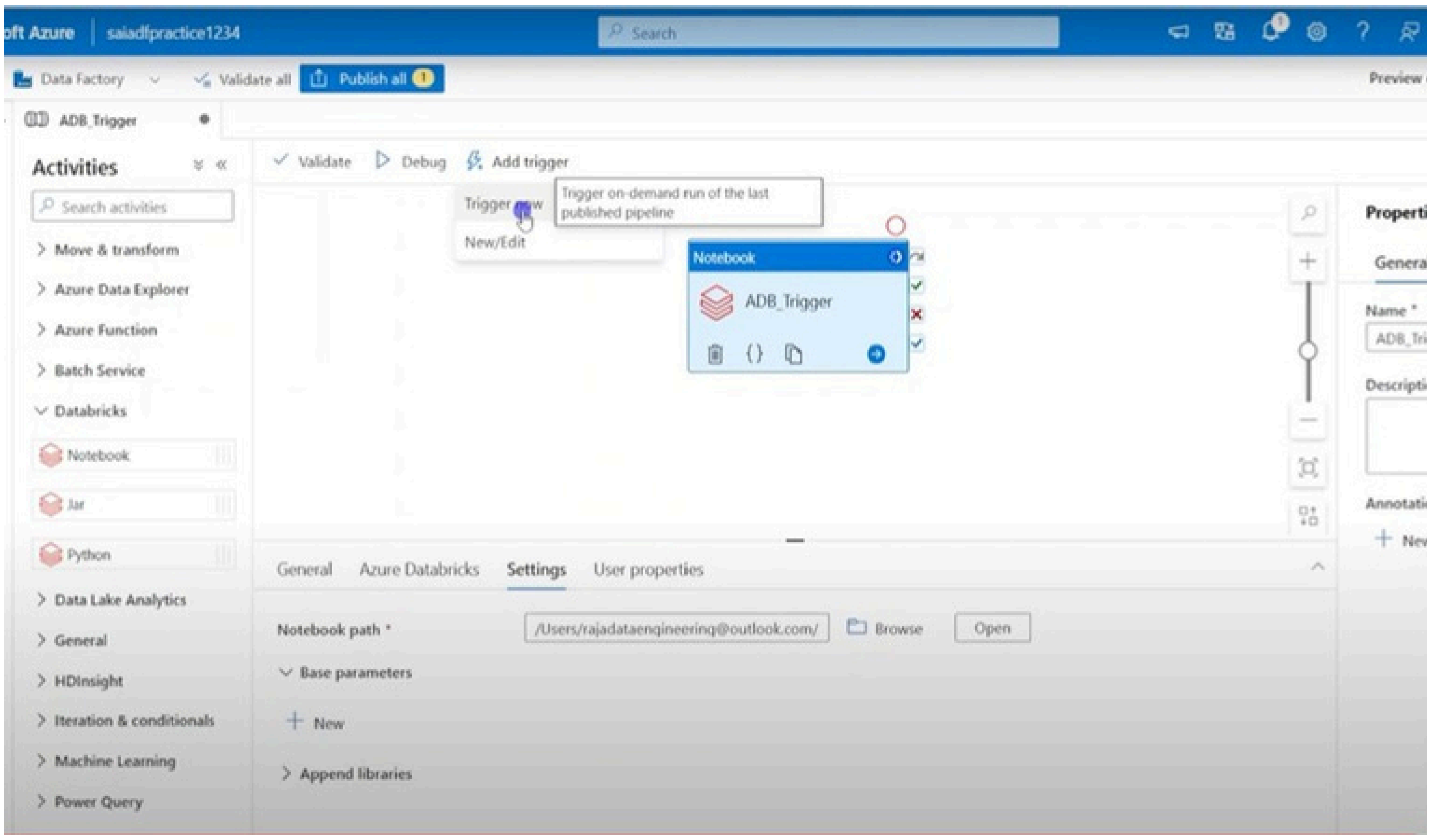
Notebook path * /Users/rajadataengineering@outlook.com/ Browse Open

Base parameters

+ New

Append libraries

- Now we can trigger our pipeline by clicking on Add trigger



Trigger on-demand run of the last published pipeline

Notebook path * /Users/rajadataengineering@outlook.com/ Browse Open

Base parameters

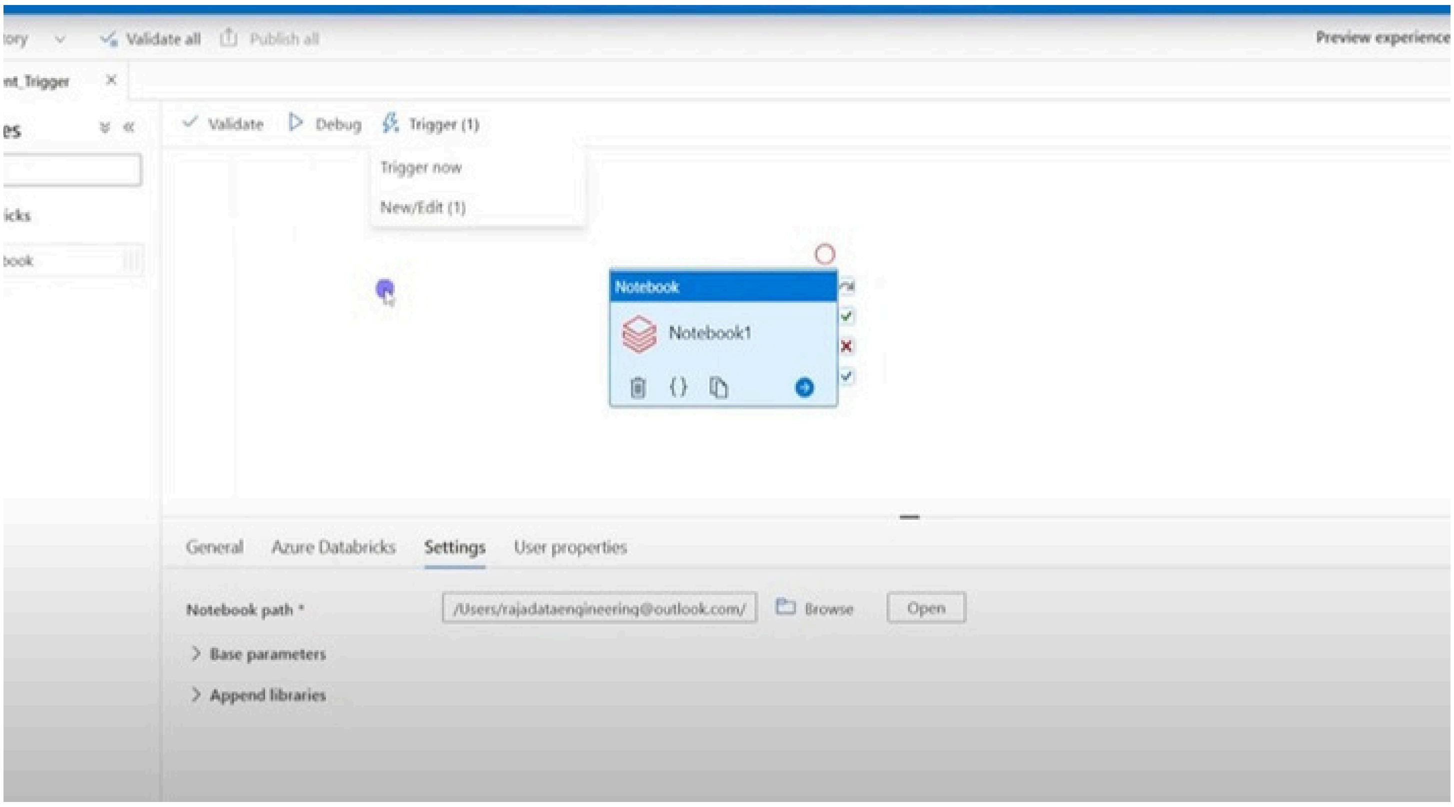
+ New

Append libraries

- But if we want to add a schedule based trigger click on Add trigger and then select 'New Edit'

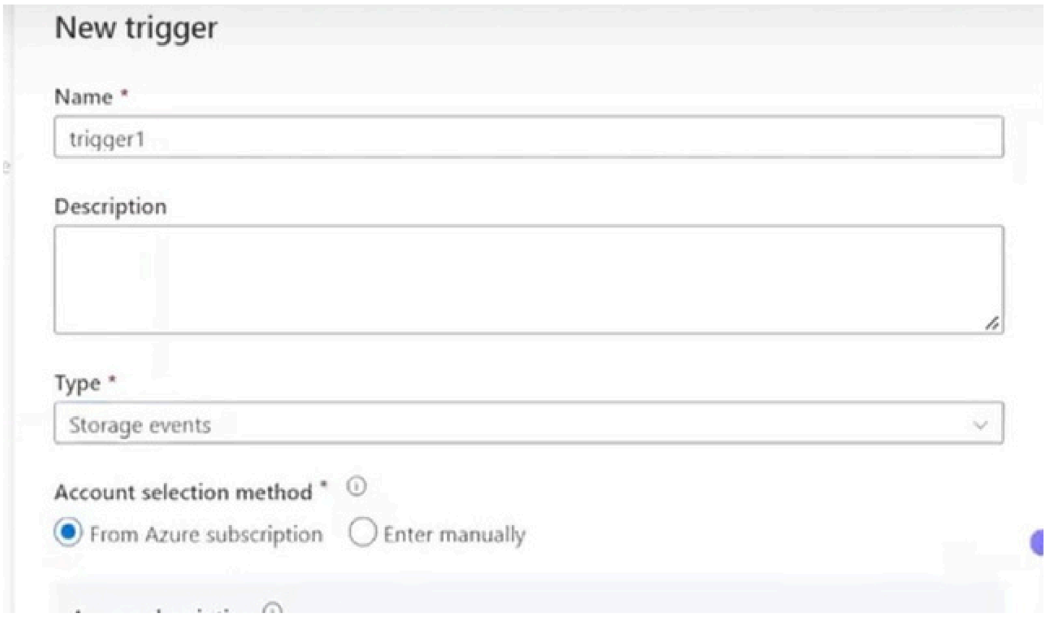
Notebook Scheduling through Event Based Trigger using Azure Data Factory

- Lets consider a simple demo.The process is we read the latest file from the azure datalake storage and create a data frame and store that back in a azure sql db. Here we have to create the pipeine as how we did in the previous case . All the configurations that we did in the previous example up to triggerig must be done over here as well.The name of the container that we have created under data lake storage is ‘demo’



Notebook path * /Users/rajadataengineering@outlook.com/

- Go to New Edit



New trigger

Name *

trigger1

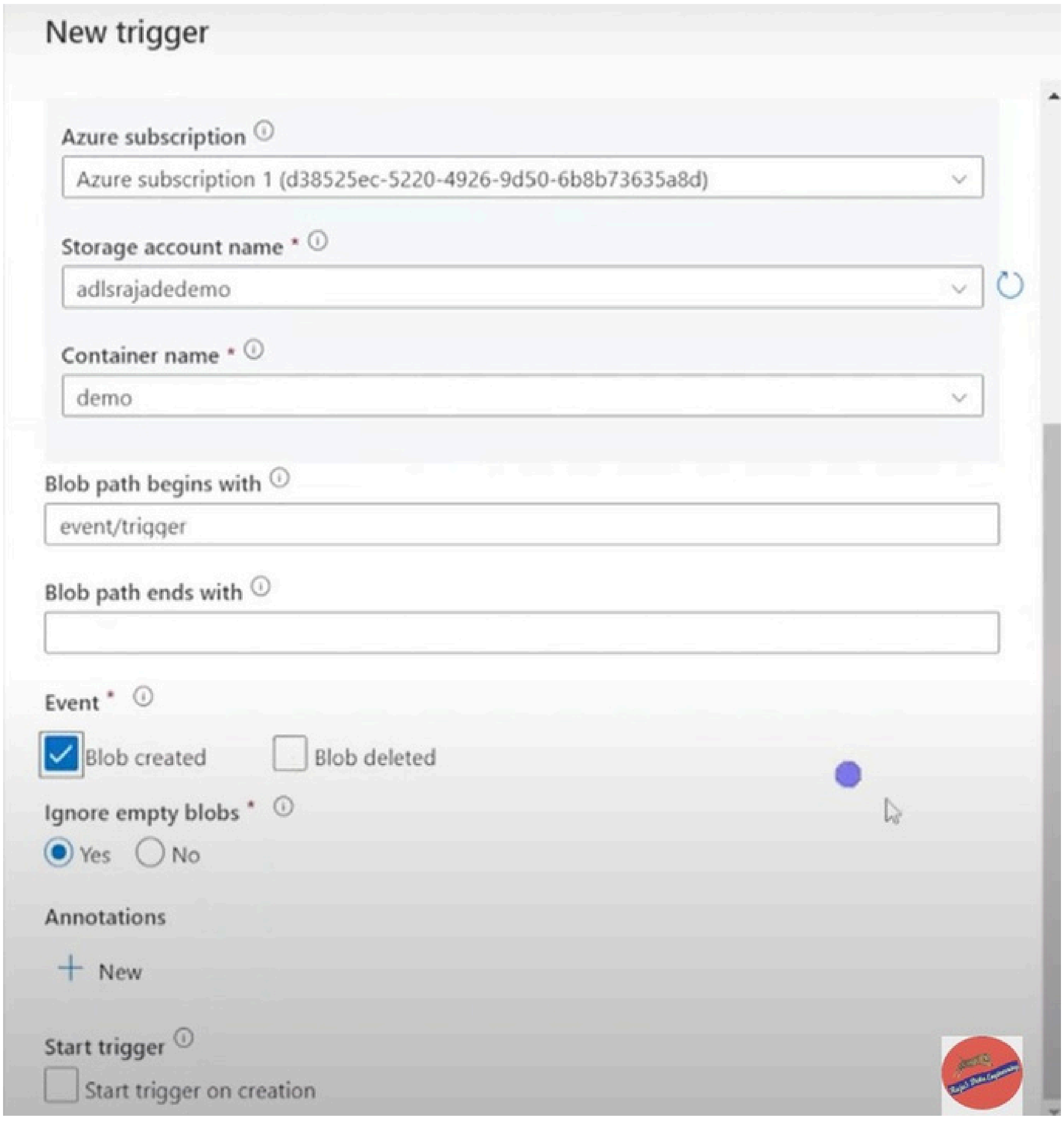
Description

Type *

Storage events

Account selection method *

[✓ From Azure subscription] Enter manually



New trigger

Azure subscription

Azure subscription 1 (d38525ec-5220-4926-9d50-6b8b73635a8d)

Storage account name *

adlsrajadedemo

Container name *

demo

Blob path begins with

event/trigger

Blob path ends with

Event *

[✓ Blob created] Blob deleted

Ignore empty blobs *

[✓ Yes] No

Annotations

+ New

Start trigger

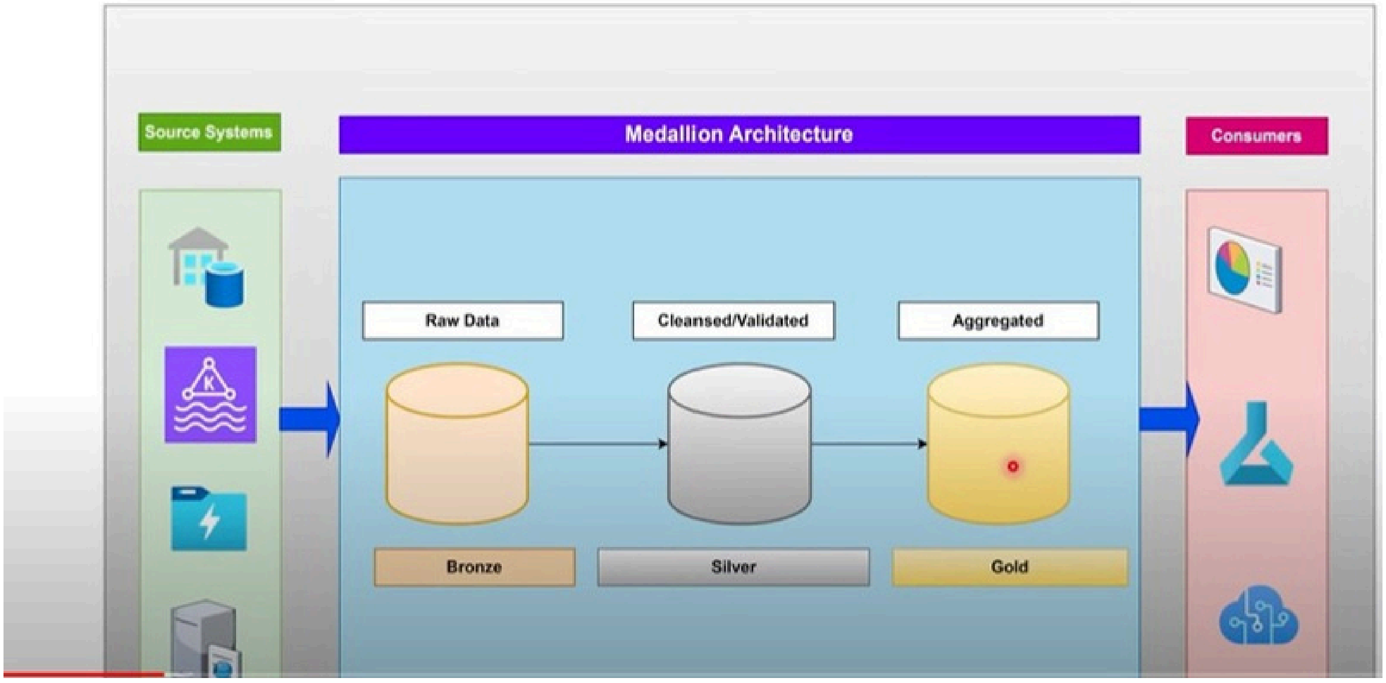
Start trigger on creation

- Now whenever a file is getting created our event will be triggered

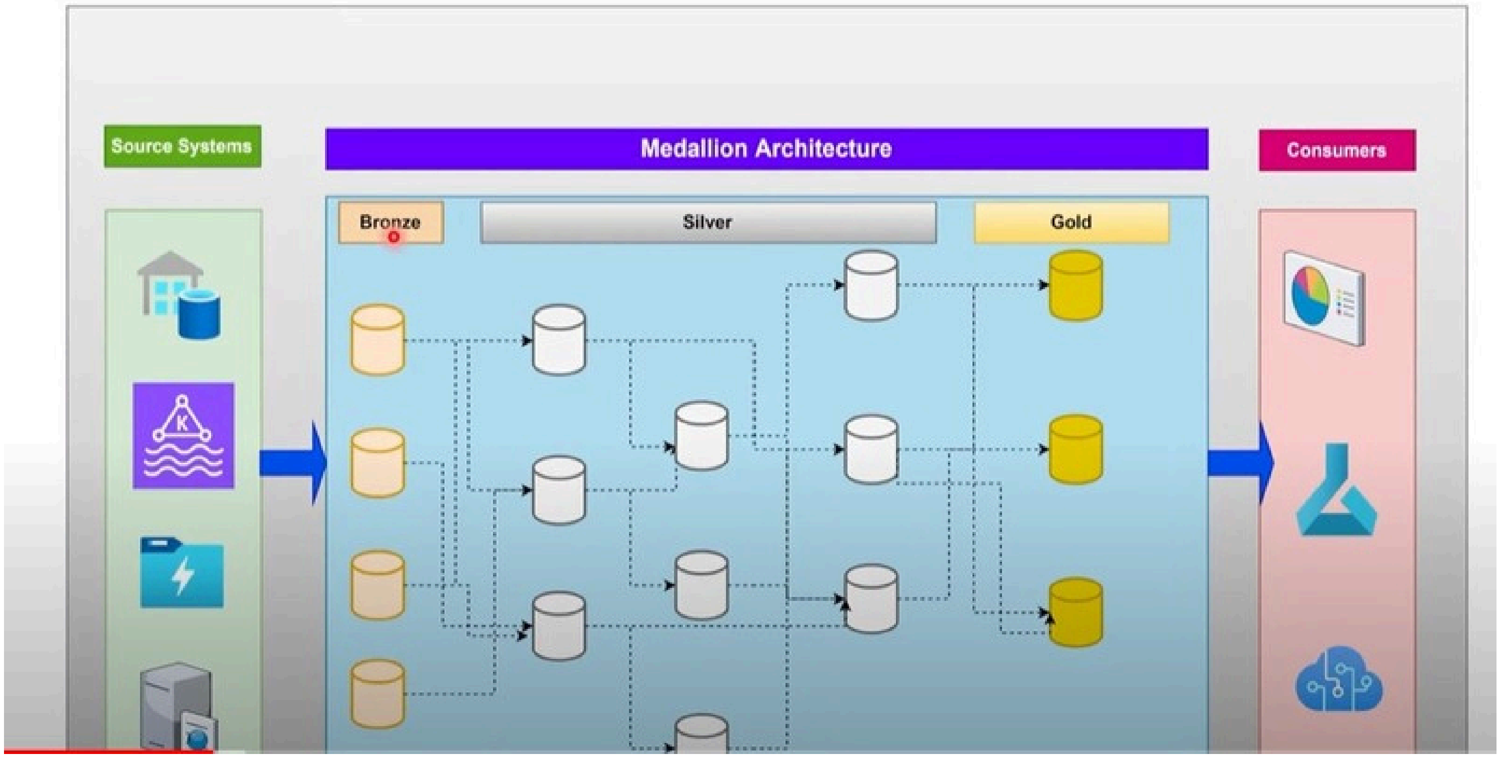
- Now go to azure control and search 'subscription' and go inside that and then go for resource providers , within that search 'microsoft.event.grid' and select that and make it registered.

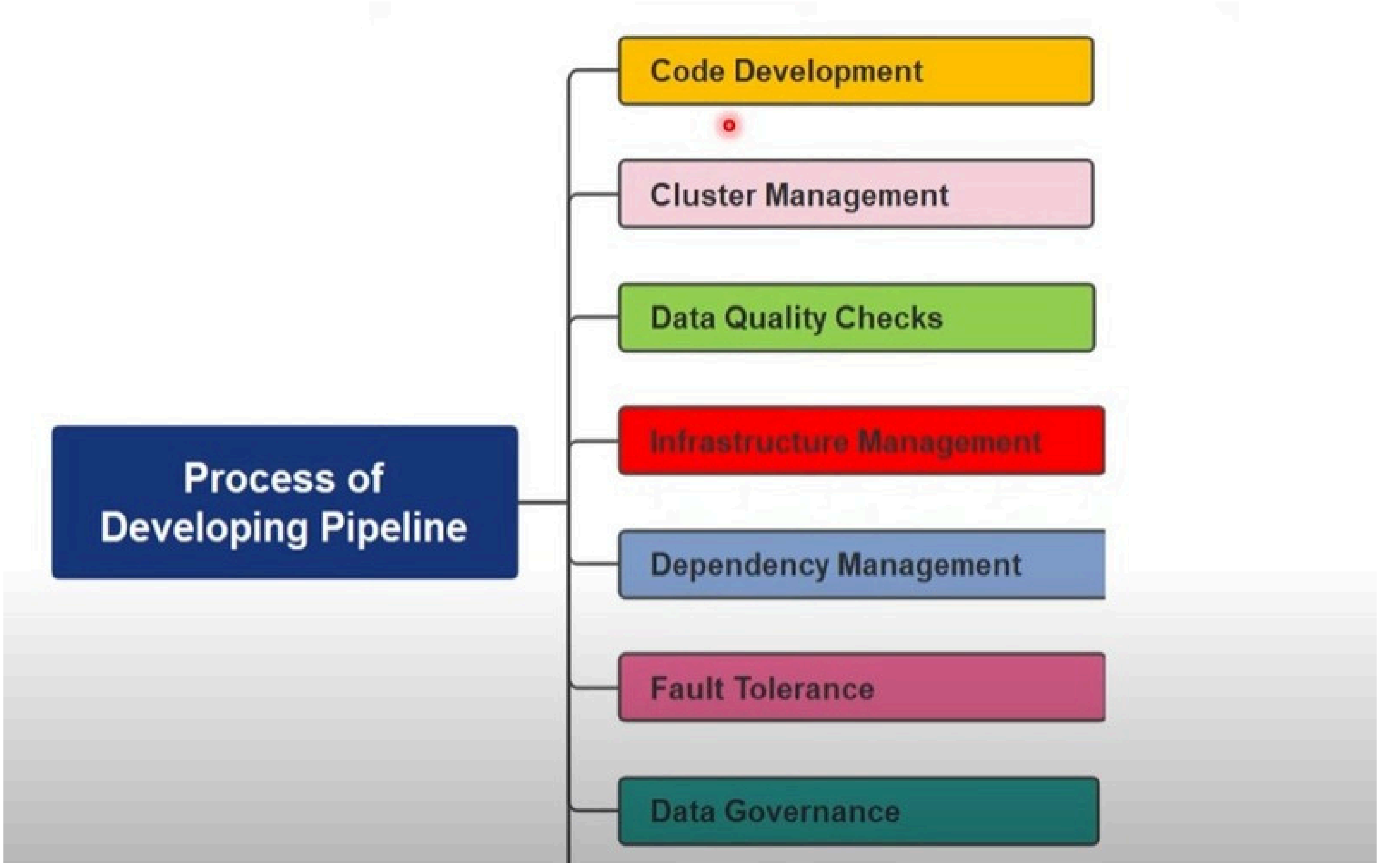
Delta Live Table

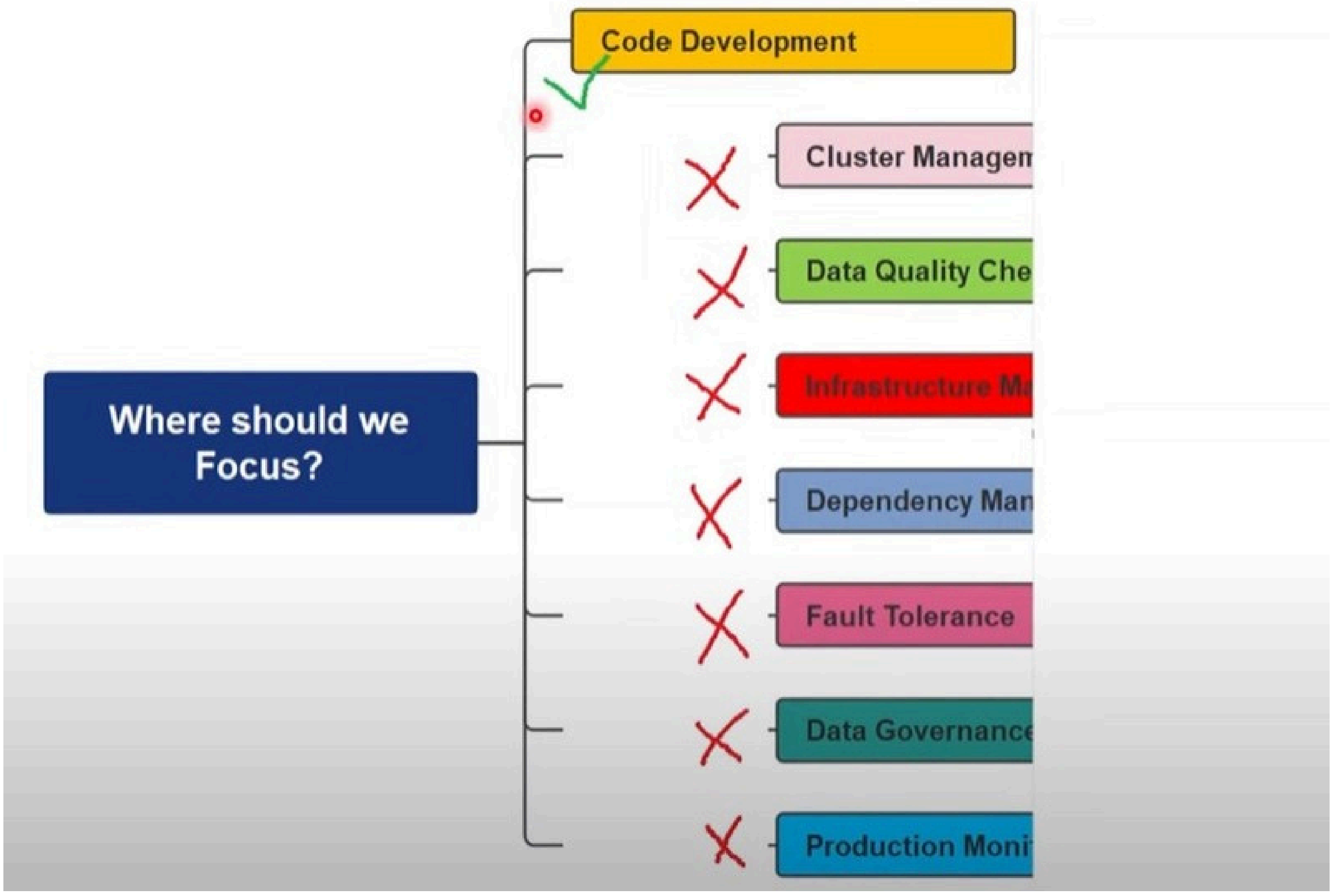
Medallion Architecture

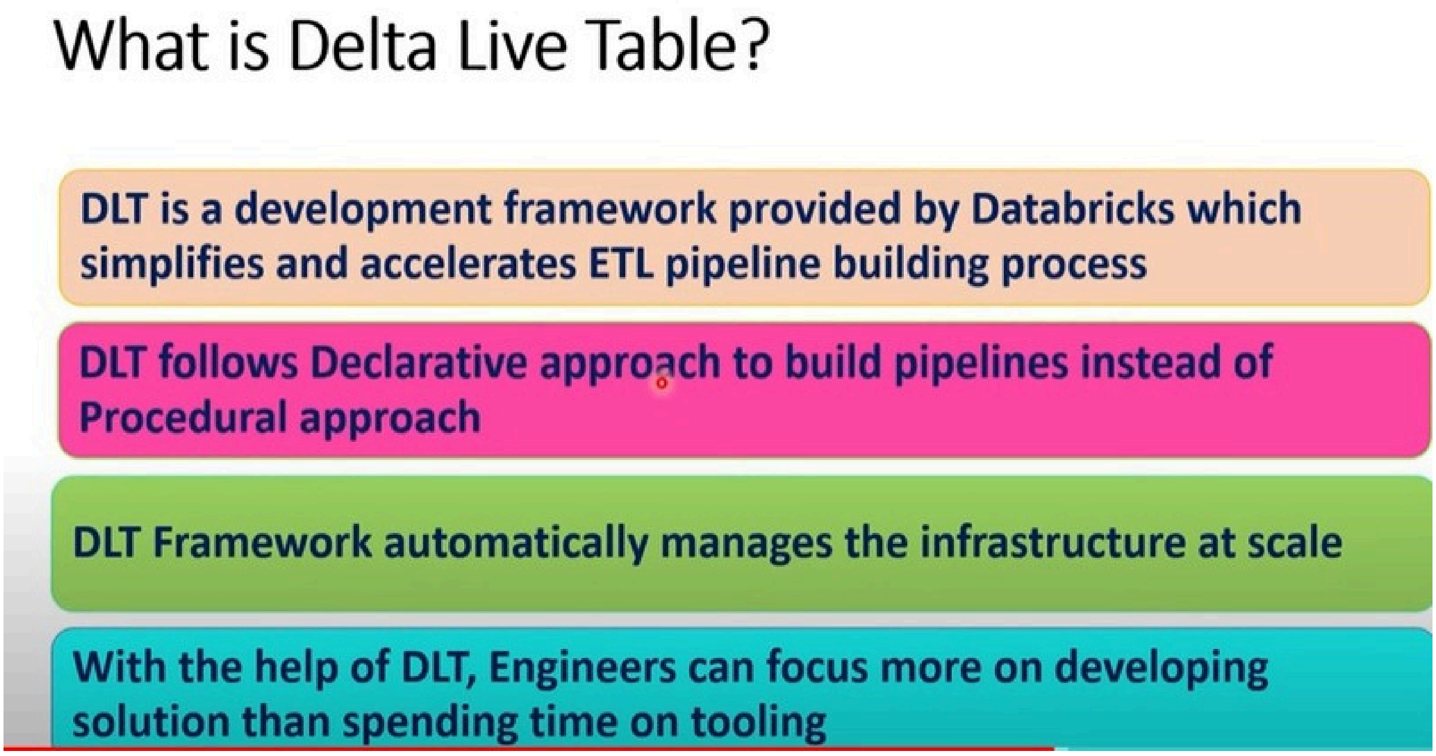


Complexity of Medallion Architecture




Process of Developing Pipeline
Code Development
Cluster Management
Data Quality Checks
Infrastructure Management
Dependency Management
Fault Tolerance
Data Governance


Where should we Focus?
Code Development
✓
Cluster Managem
✗
Data Quality Che
✗
Infrastructure Ma
✗
Dependency Man
✗
Fault Tolerance
✗
Data Governance
✗
Production Moni
✗



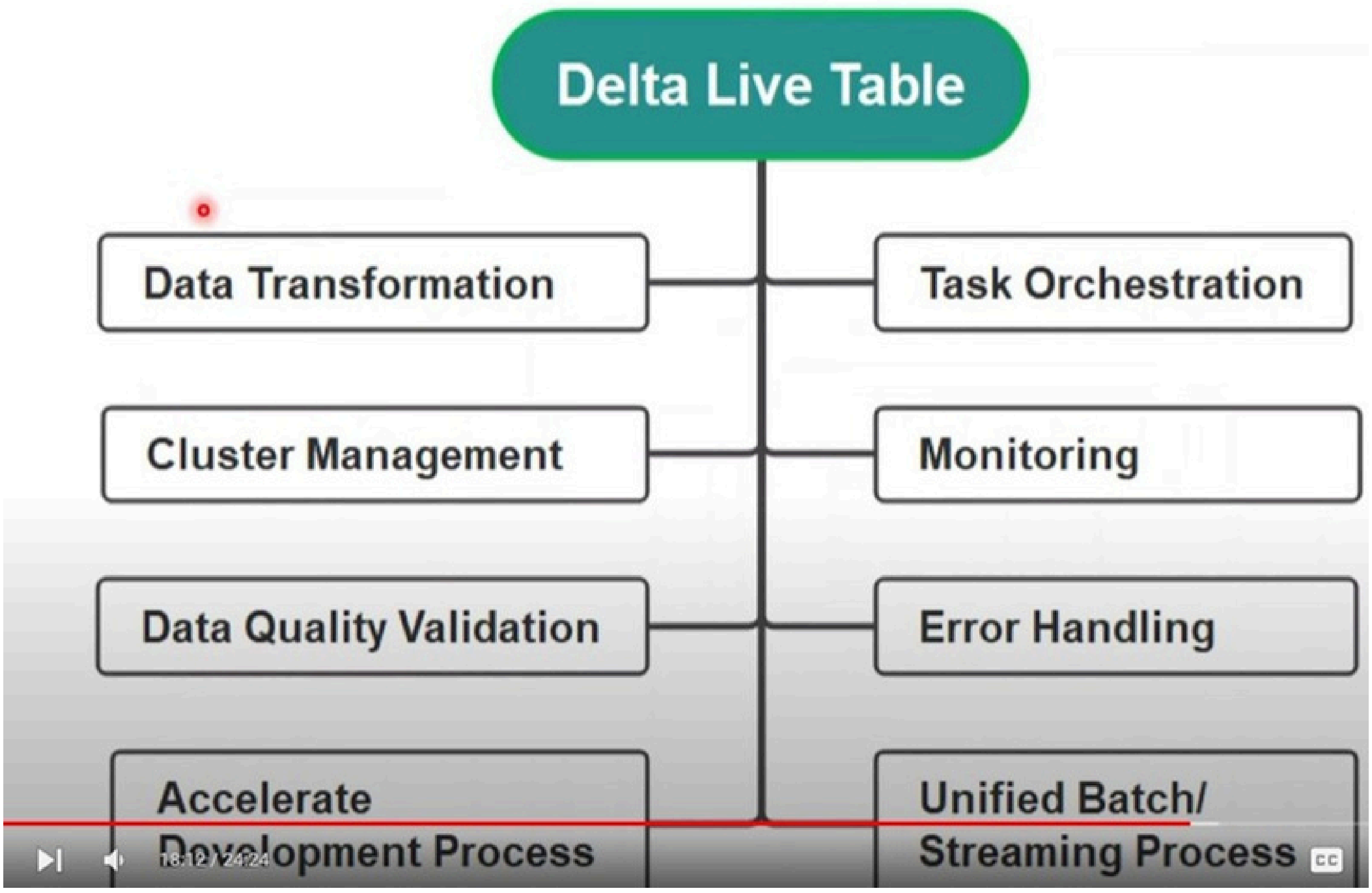
What is Delta Live Table?

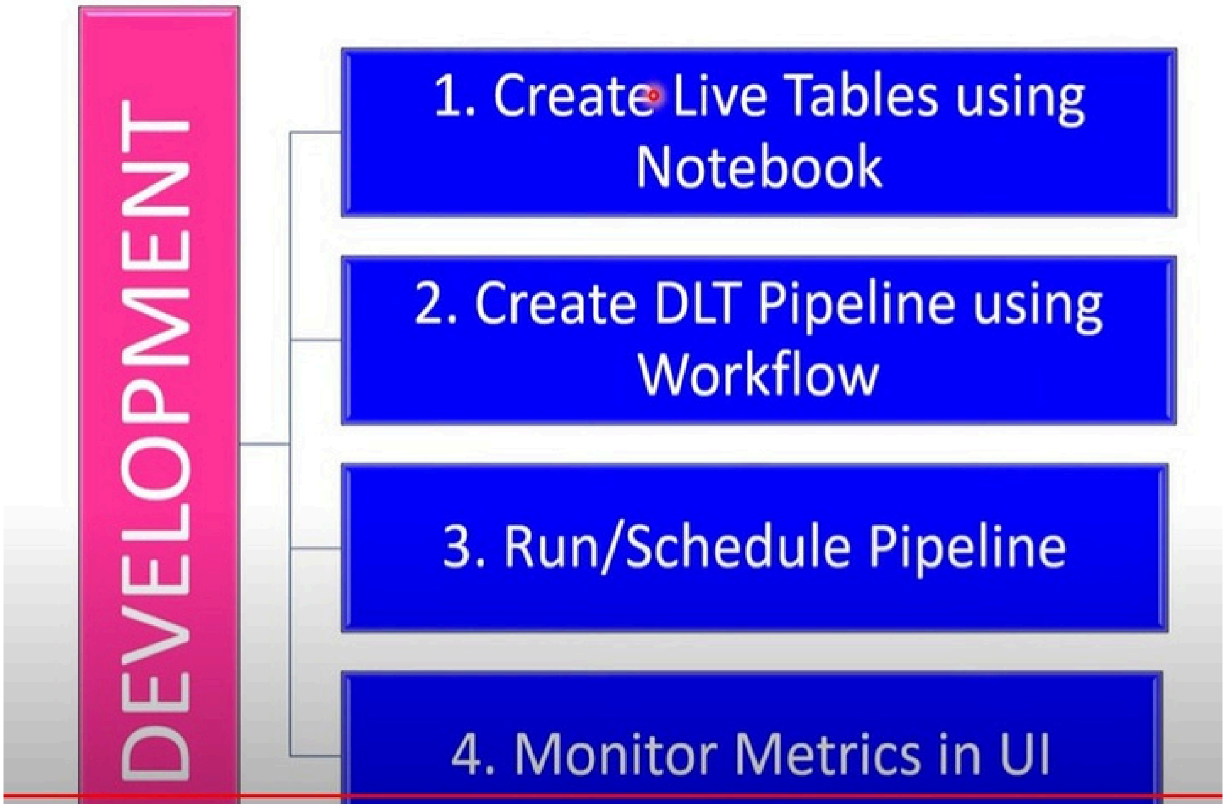
DLT is a development framework provided by Databricks which simplifies and accelerates ETL pipeline building process

DLT follows Declarative approach to build pipelines instead of Procedural approach

DLT Framework automatically manages the infrastructure at scale

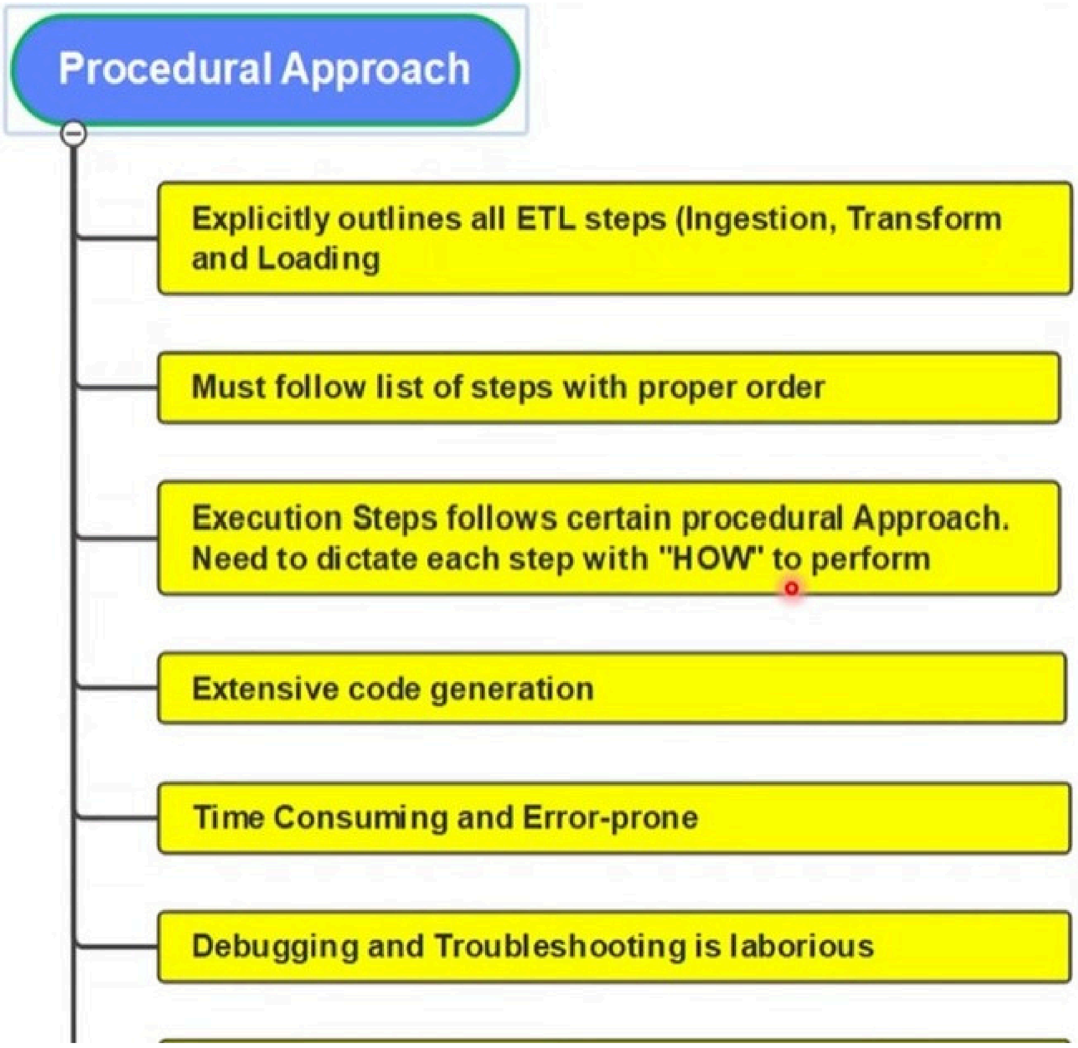
With the help of DLT, Engineers can focus more on developing solution than spending time on tooling





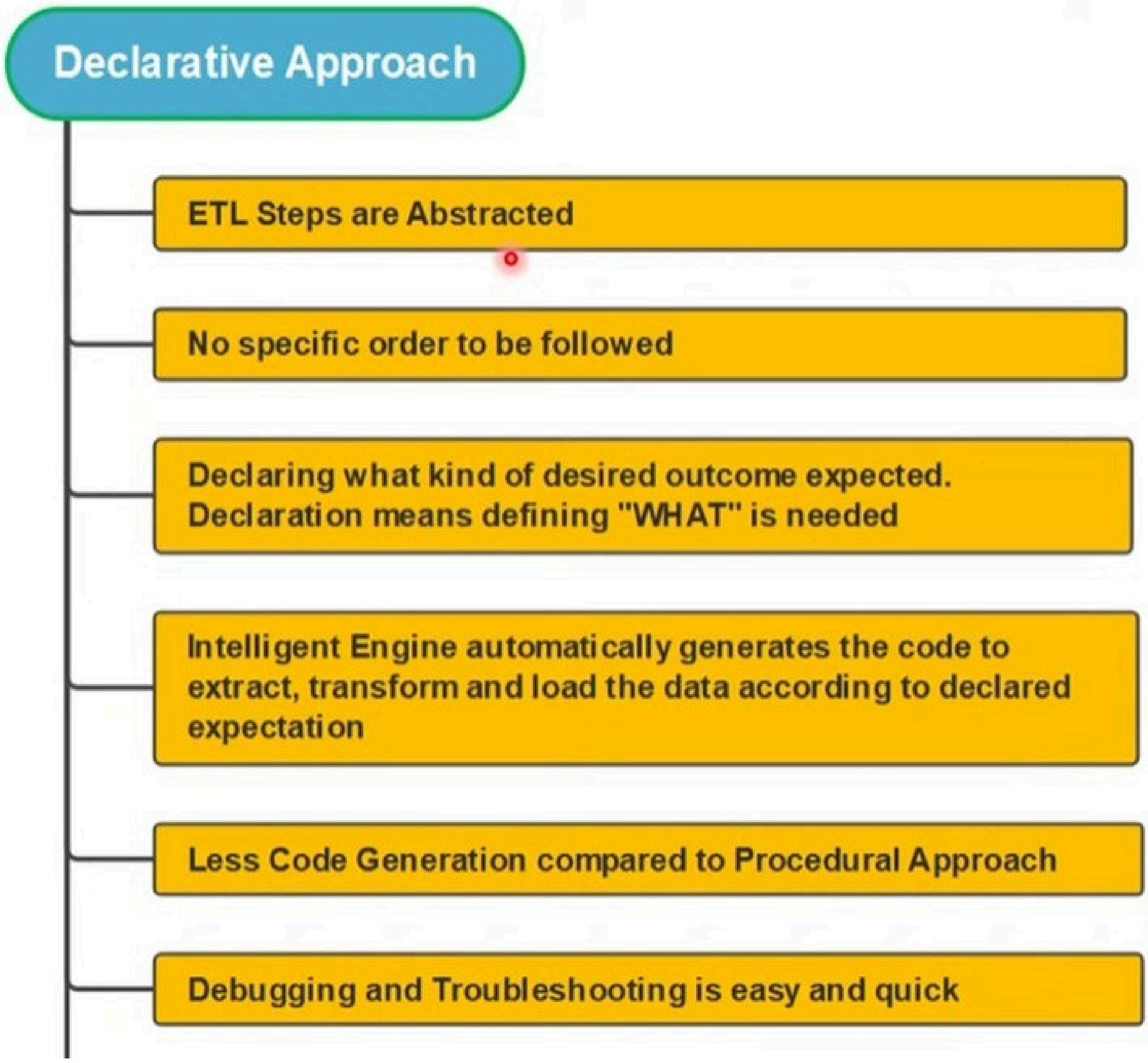
DEVELOPMENT

1. Create Live Tables using Notebook
2. Create DLT Pipeline using Workflow
3. Run/Schedule Pipeline
4. Monitor Metrics in UI



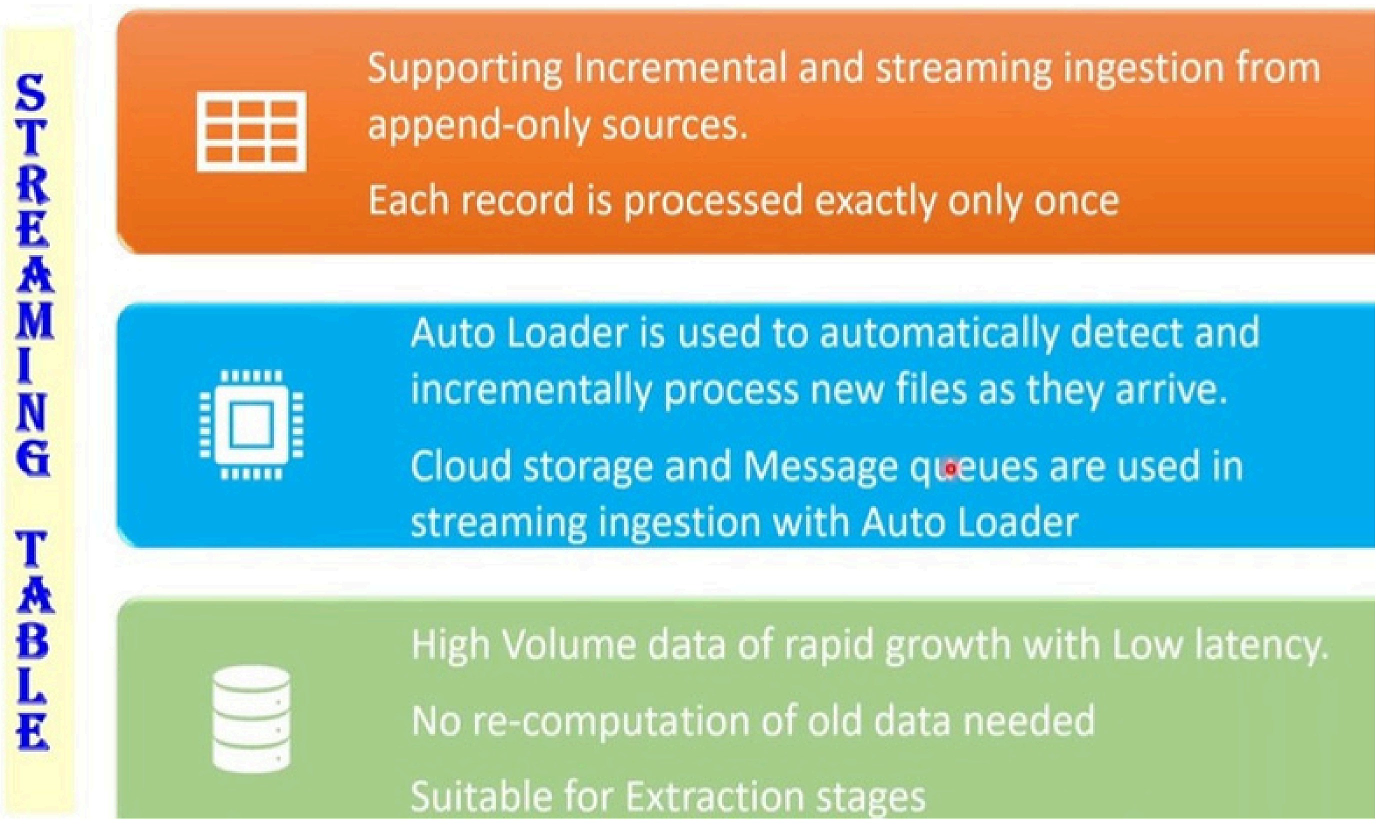
Procedural Approach

- Explicitly outlines all ETL steps (Ingestion, Transform and Loading
- Must follow list of steps with proper order
- Execution Steps follows certain procedural Approach. Need to dictate each step with "HOW" to perform
- Extensive code generation
- Time Consuming and Error-prone
- Debugging and Troubleshooting is laborious



Declarative Approach

- ETL Steps are Abstracted
- No specific order to be followed
- Declaring what kind of desired outcome expected. Declaration means defining "WHAT" is needed
- Intelligent Engine automatically generates the code to extract, transform and load the data according to declared expectation
- Less Code Generation compared to Procedural Approach
- Debugging and Troubleshooting is easy and quick



STREAMING TABLE

Supporting Incremental and streaming ingestion from append-only sources.

Each record is processed exactly only once

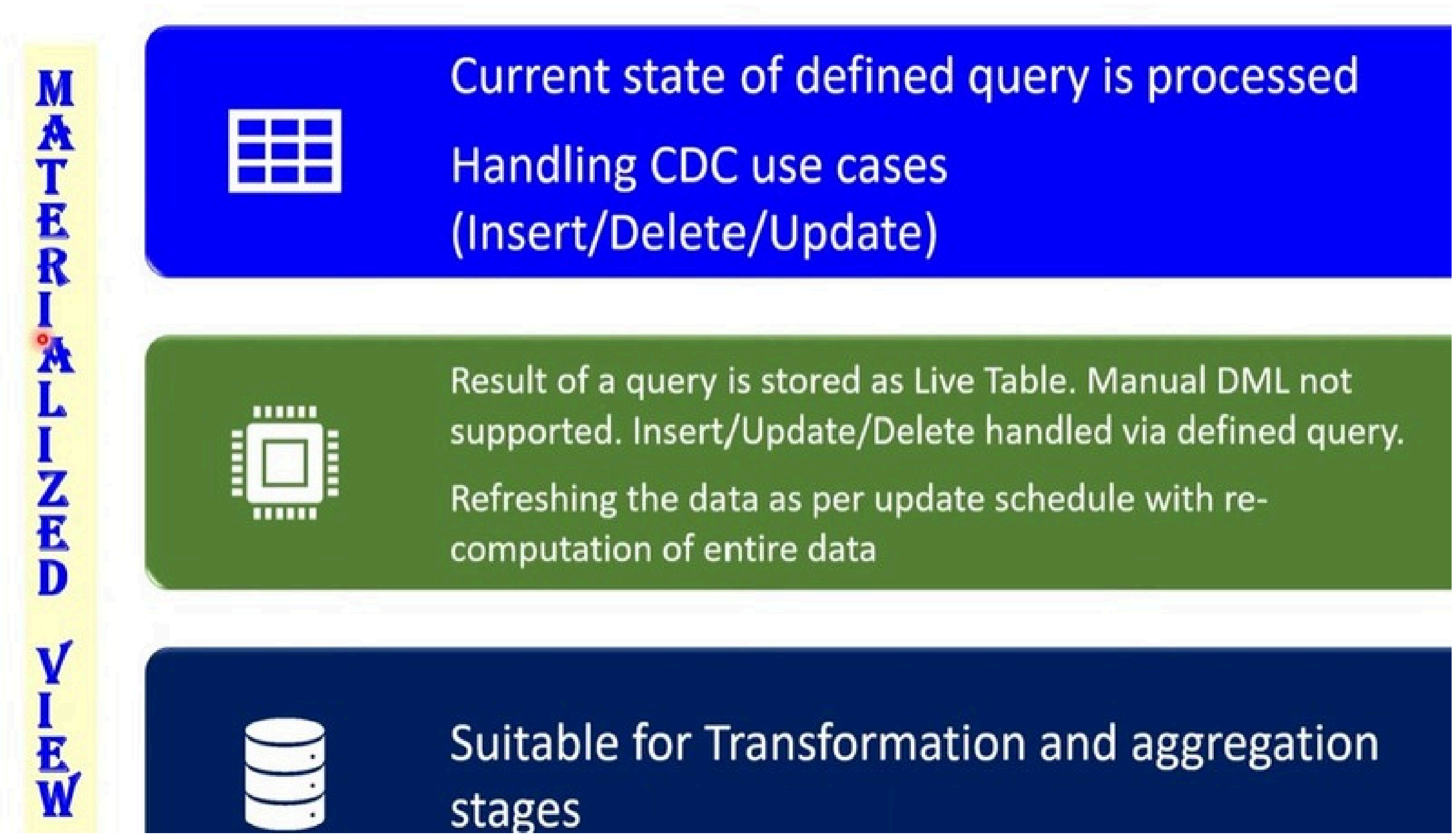
Auto Loader is used to automatically detect and incrementally process new files as they arrive.

Cloud storage and Message queues are used in streaming ingestion with Auto Loader

High Volume data of rapid growth with Low latency.

No re-computation of old data needed

Suitable for Extraction stages



MATERIALIZED VIEW

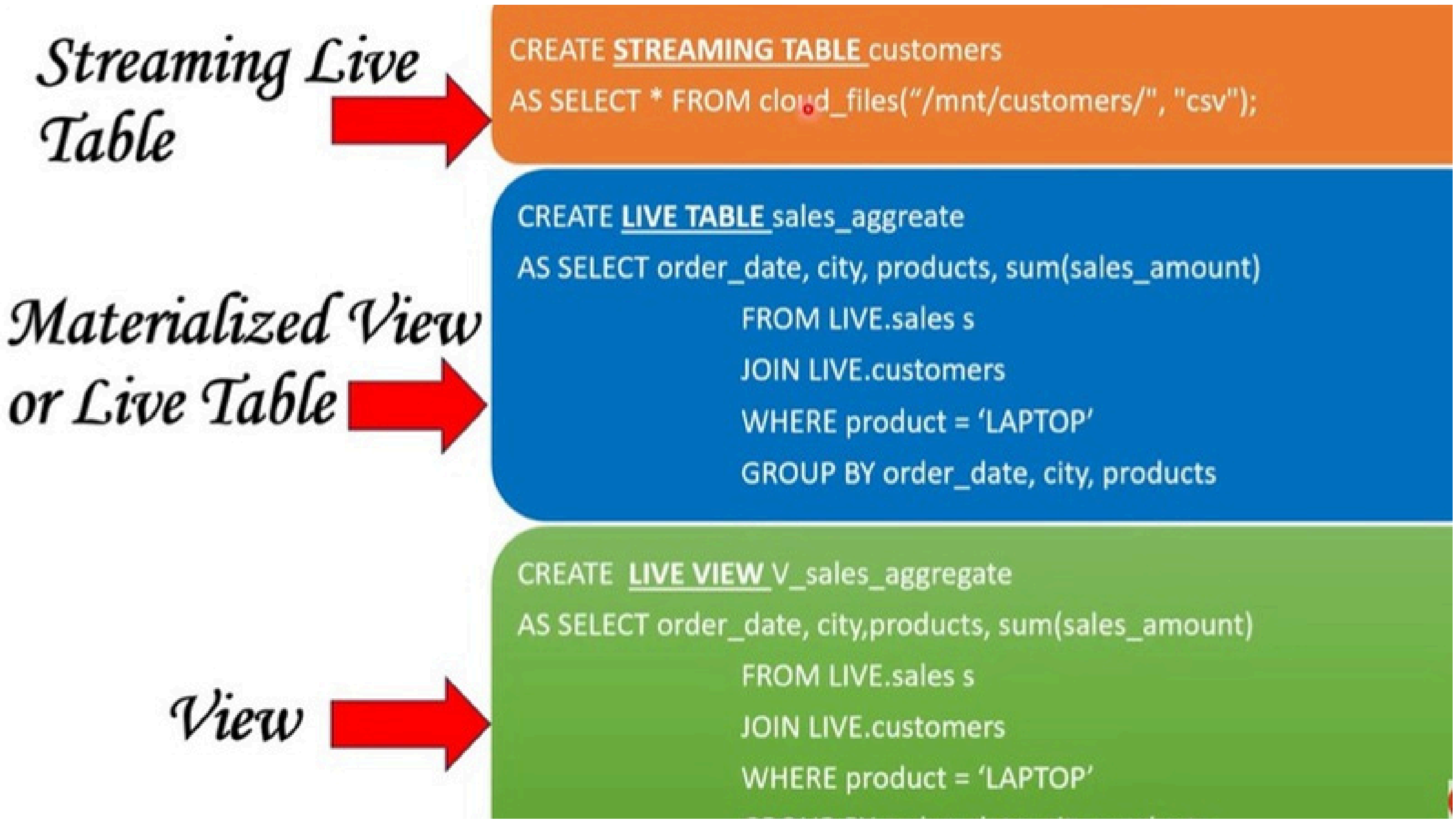
Current state of defined query is processed

Handling CDC use cases (Insert/Delete/Update)

Result of a query is stored as Live Table. Manual DML not supported. Insert/Update/Delete handled via defined query.

Refreshing the data as per update schedule with re-computation of entire data

Suitable for Transformation and aggregation stages



Streaming Live Table

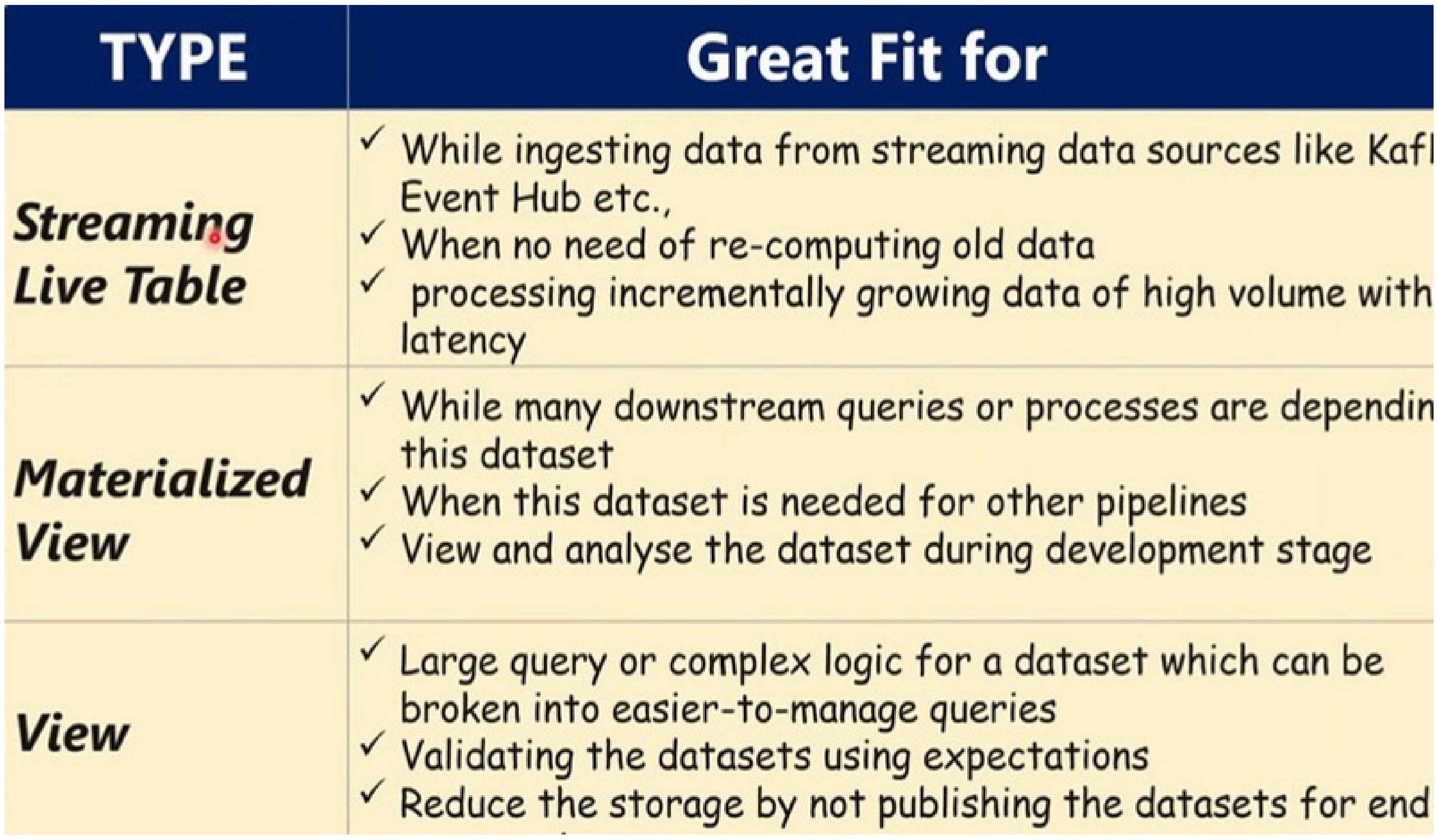
```
CREATE STREAMING TABLE customers
AS SELECT * FROM cloud_files("/mnt/customers/", "csv");
```

Materialized View or Live Table

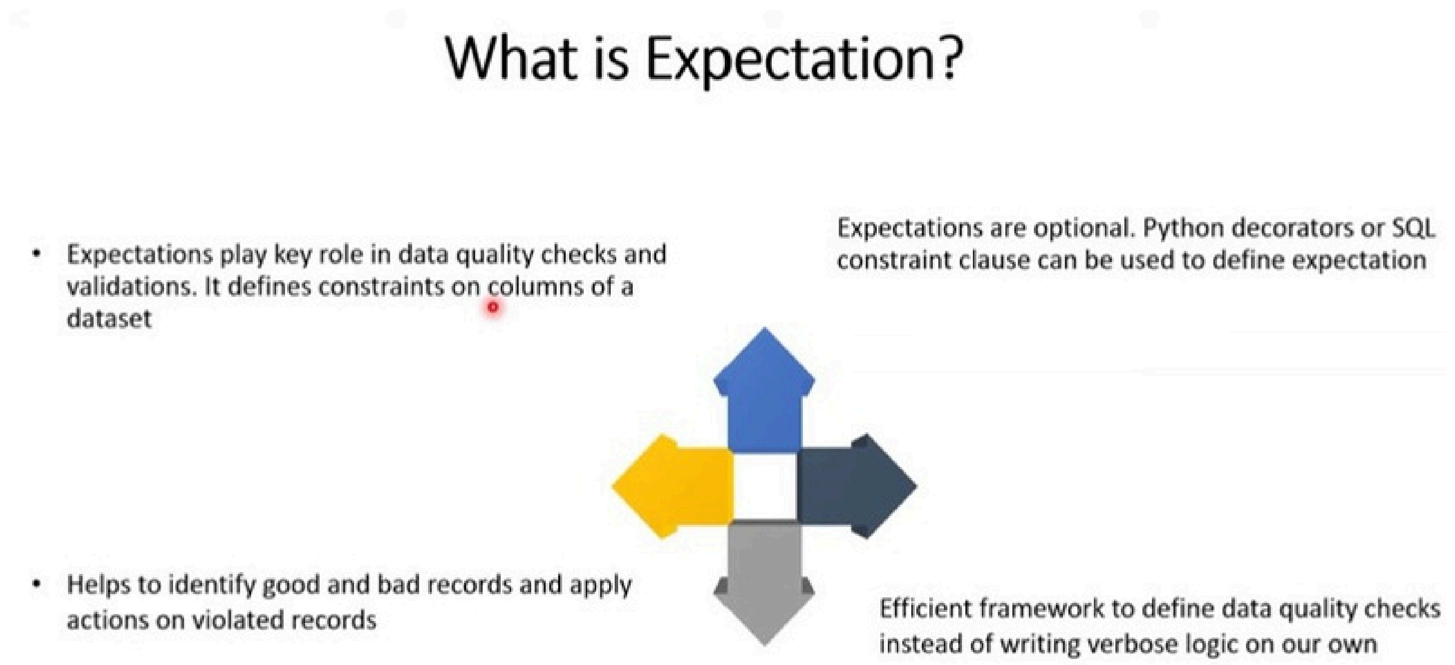
```
CREATE LIVE TABLE sales_aggreate
AS SELECT order_date, city, products, sum(sales_amount)
            FROM LIVE.sales s
            JOIN LIVE.customers
            WHERE product = 'LAPTOP'
            GROUP BY order_date, city, products
```

View

```
CREATE  LIVE VIEW V_sales_aggregate
AS SELECT order_date, city,products, sum(sales_amount)
            FROM LIVE.sales s
            JOIN LIVE.customers
            WHERE product = 'LAPTOP'
```



TYPE	Great Fit for
Streaming Live Table	✓ While ingesting data from streaming data sources like Kafl Event Hub etc., ✓ When no need of re-computing old data ✓ processing incrementally growing data of high volume with latency
Materialized View	✓ While many downstream queries or processes are dependin this dataset ✓ When this dataset is needed for other pipelines ✓ View and analyse the dataset during development stage
View	✓ Large query or complex logic for a dataset which can be broken into easier-to-manage queries ✓ Validating the datasets using expectations ✓ Reduce the storage by not publishing the datasets for end



What is Expectation?

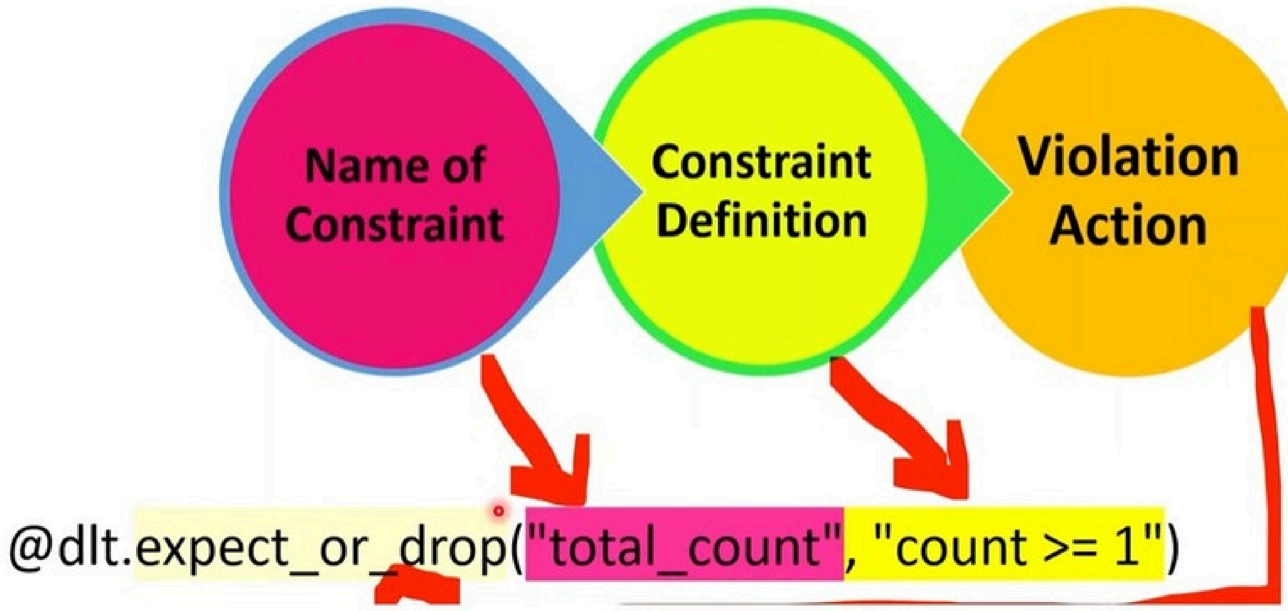
- Expectations play key role in data quality checks and validations. It defines constraints on columns of a dataset

Expectations are optional. Python decorators or SQL constraint clause can be used to define expectation

- Helps to identify good and bad records and apply actions on violated records

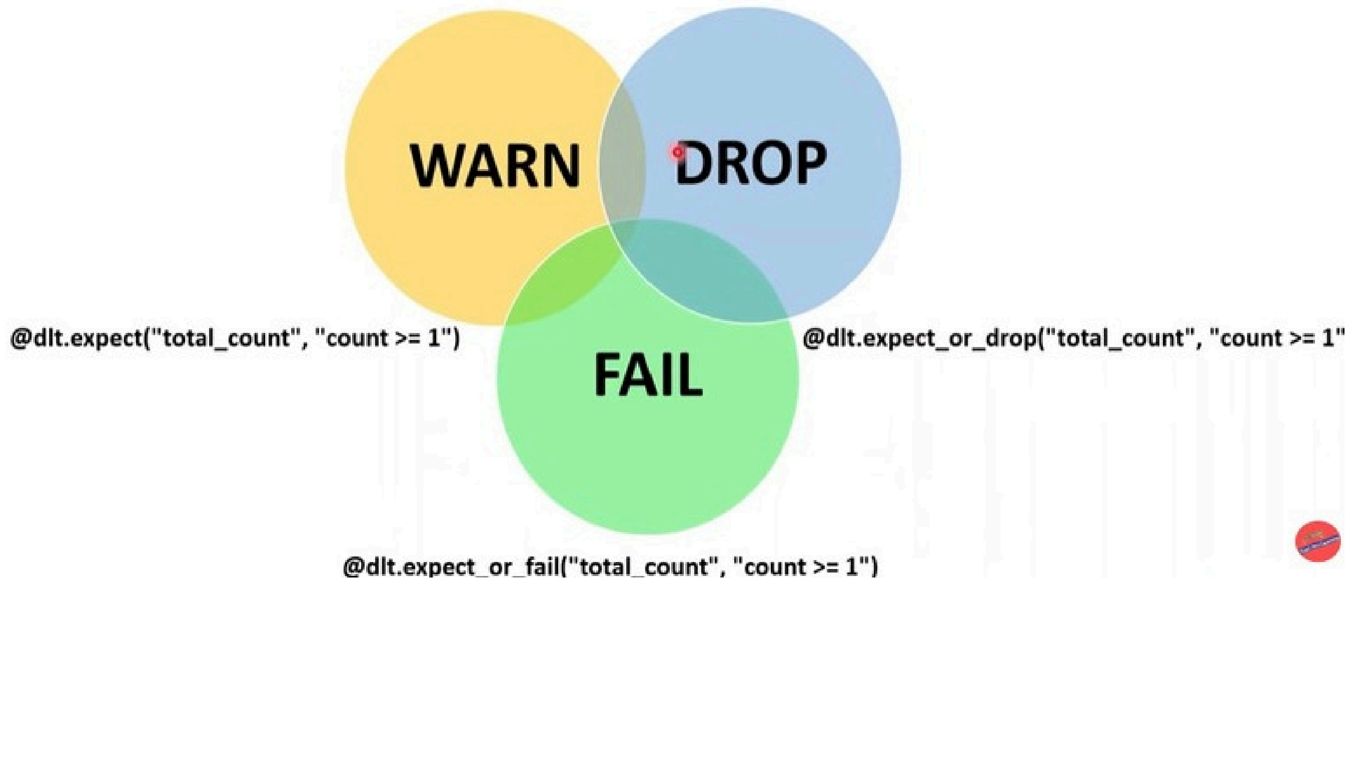
Efficient framework to define data quality checks instead of writing verbose logic on our own

Components of Expectation



@dlt.expect_or_drop("total_count", "count >= 1")

Violation Action



@dlt.expect("total_count", "count >= 1")

@dlt.expect_or_drop("total_count", "count >= 1"

@dlt.expect_or_fail("total_count", "count >= 1")