REST API:

Link(Endpoint) used to share data between 2 different sources usually created by developers…
rest api data can be of many types like json,image,XML or can be anything but mostly it is JSON
just google some free api and we can get so many endpoints to test them

Accessing can be done by selecting “data set as rest api” and creating link service using “rest api web link address”.
Authentication mechanism for Link Service

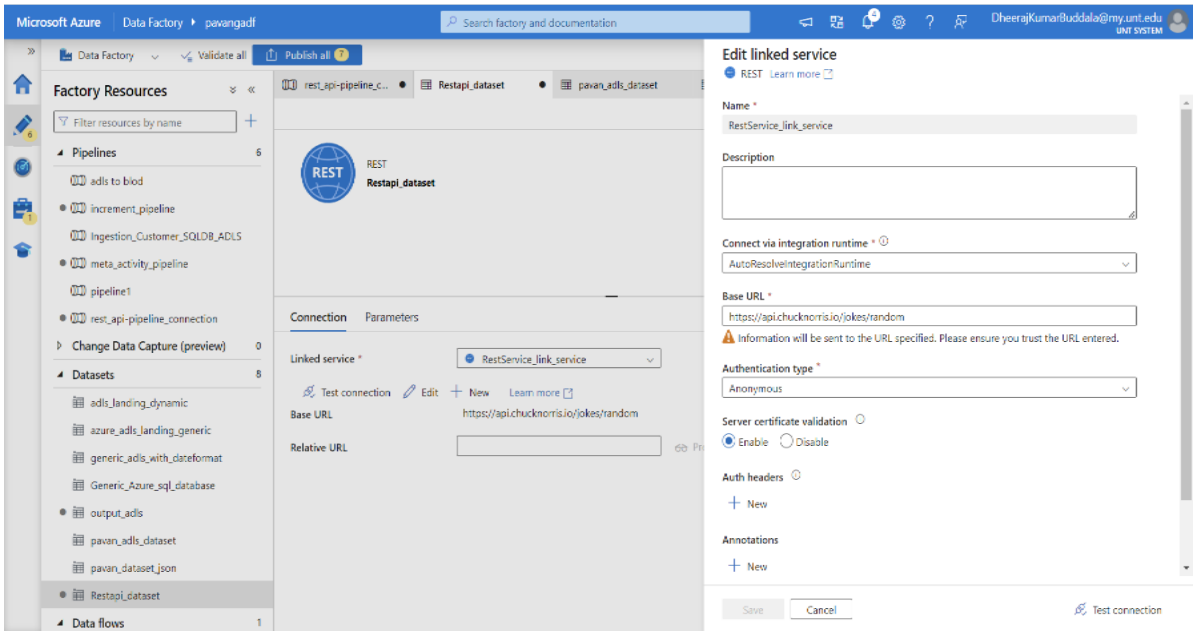
1.anonymous = anybody can access

2.basic = accessing through id and pass

3.OA = c=accessing through token id and pass all these id and pass details given by the developer who created the link.

Sometimes url or endpoints keep on changing that case we can make unchangeable link as base url and changing link as relative url in dataset where we can pass the values on runtime this mode makes link service flexible.

Request method: get is used if we want display parameters(ex: if we search java it will show java search in web address) passed in web address, post is used to not display parameter(ex:login request) generally this is given by developers only.

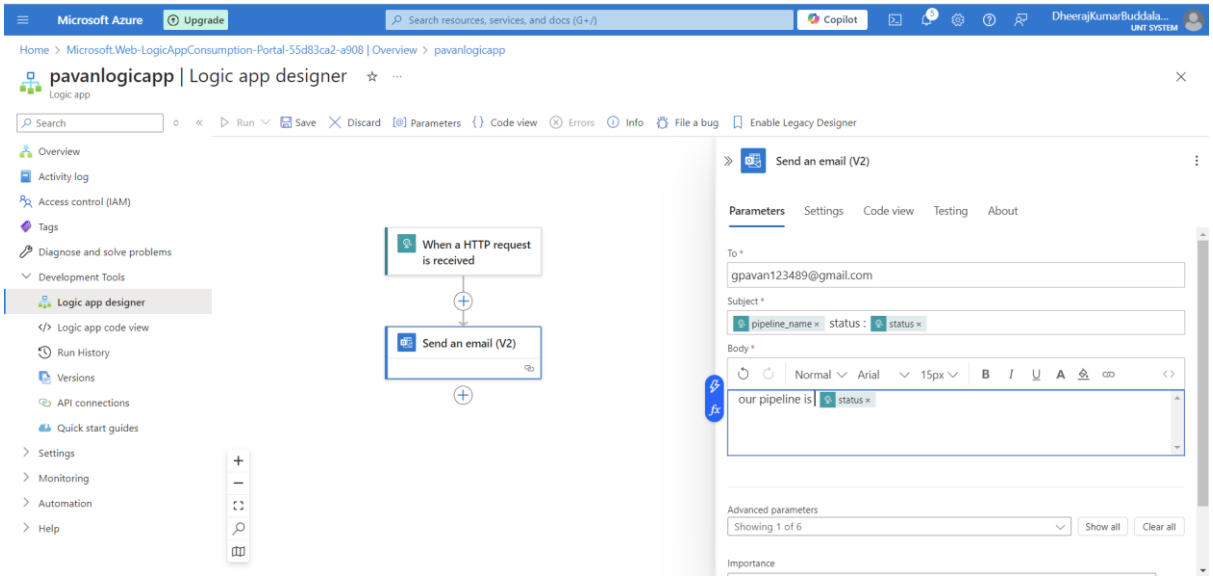


LOGIC APPS(High level enough):

It is a kind of azure service similar to azure key vault, ADF, storage account in order to use it create account same as u created for above it is used to create worlkflows.
this functionality can be used in many ways and can be accessed by the trigger point which are created in the logic app functionality.

example: sending or replying to mail based on trigger functionality(based on http request or anything) and also “dynamically use parmeters like pipeline name,status of activity through body we sent” → to use this we need to set the schema in logic app trigger(which can be done easily once you click create schema and provide your data receiving expectation)

most powerful azure service mostly used by developers but not much relevant to dataengineer



WEB ACTIVITY:

This activity used just to hit the URL once and nothing more can be used as trigger for logic apps (activity used whenever we need to call URL)

How to get notified?
in order to get notified we can use logic app and send the email but it is a time consuming process so we have **“alerts and metrices”** in monitor tab which help us to set the rules like pipeline succeeded or failed and allocate checking freq and all , and at last we can set remainder via sms or email

TRIGEERS:

Debug option allows the pipeline to run instantly,,, triggering options allows to run pipeline based on specific trigger it is 3 types.

Schedule Trigger: used to trigger pipeline based on time and can also set frequency of occurring of pipeline.
this trigger doesn’t care whether pipeline is succedded or not just runs it that’s it.

Storage key Trigger: here we can trigger the pipeline based on file creation and deletion in specific path and by also type of the file (like csv) (will be very useful)
use case: instead of using execute pipeline, we can just use one copy acticity and copy a dummy file and this file can be used as a triggering point to the another pipeline

Tumbling window Trigger: this usually does schedule trigger work but has extra features like dependency trigger(should be another tumbling window),concurrency level(max parllel

executions),retry option when failed,
Most importantly this trigger can only associated to one pipeline,
min schedule time should be 5 min unlike 1 min in schedule trigger.
use case: to check on any pipeline and to use any pipeline for small interval of time,

For dependency : offset window time is set based on which window time it should depend on main trigger, and window size represent for how many hours need to depend from offset “

1. **Without Offset**:
    - Window size: 1 hour
    - Triggers fire at: 12:00, 1:00, 2:00, 3:00, etc.
2. **With 10-minute Offset**:
    - Window size: 1 hour
    - Triggers fire at: 12:10, 1:10, 2:10, 3:10, etc.

Trigger CREATION can be done in pipeline level and also can create in manage tab,,, we can also stop the triggers in manage section

NOTE: One pipeline can have many triggers and one trigger can have multiple pipelines except tumbling window trigger. Debug until option allows pipeline to get executed upto that activity

Tumbling	Schedule
Minimum time 5 minute	1 minute
one to many	many to many
Add dependency (another tumbling window only)	
Have retry option	
Have concurrency option	

MONITOR:

It has multiple sections where in dashboard it shows how many pipeline ran and their success percentage and activities success percentage

Pipeline runs: shows the run of the pipline within 45 days and also shows trigger run pipelines

Trigger run: shows specific trigger run pipeline it also shows tigger pipeline ran based on type of trigger

Notification and metrices: used to notify used based on activity given by user like pipeline success or failure

NOTE:

Maximum data integration unit in copy activity gives flexible to opt us to use high-speed computational unit

DEVOPS:

Git (on prem version controller where we can control the different version of the code here everything will be within organization as server is also owned by organization) vs GitHub (cloud version controller here the server is not owned by organization).

ARM(azure resource manager) template is usually found in pipeline creation this template will be saved in the azure DevOps git

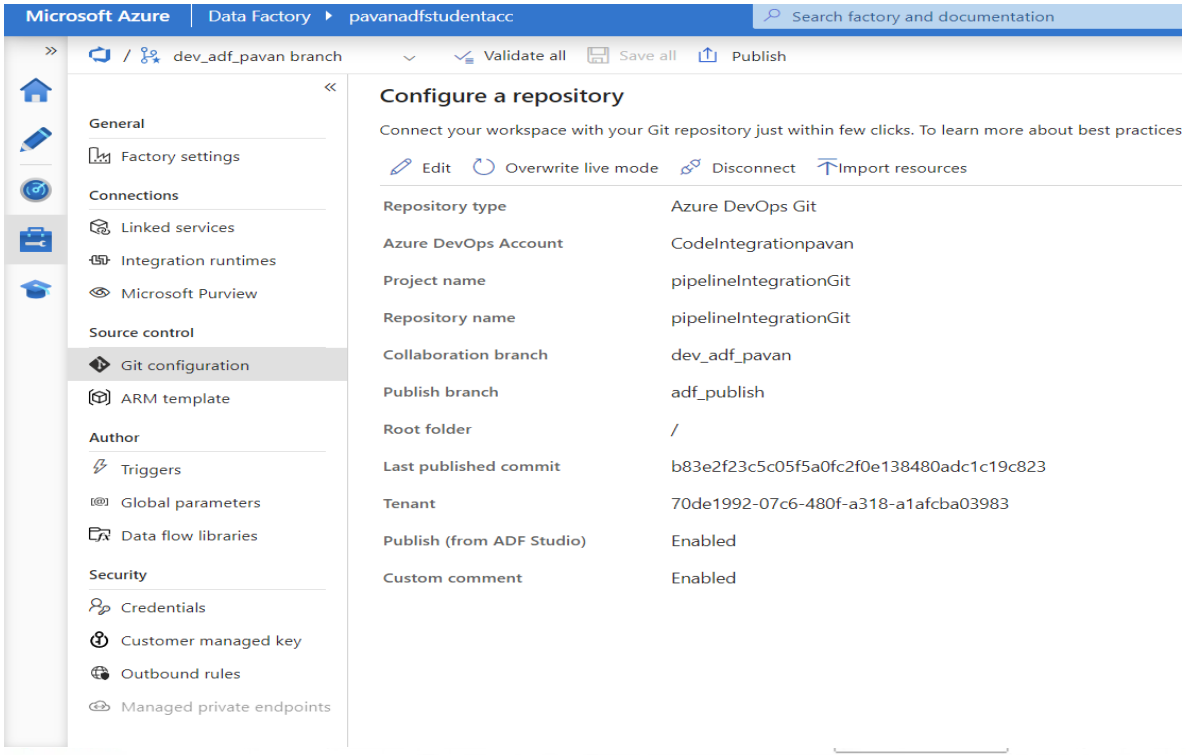
Creation of Azure Devops Git in ADF:

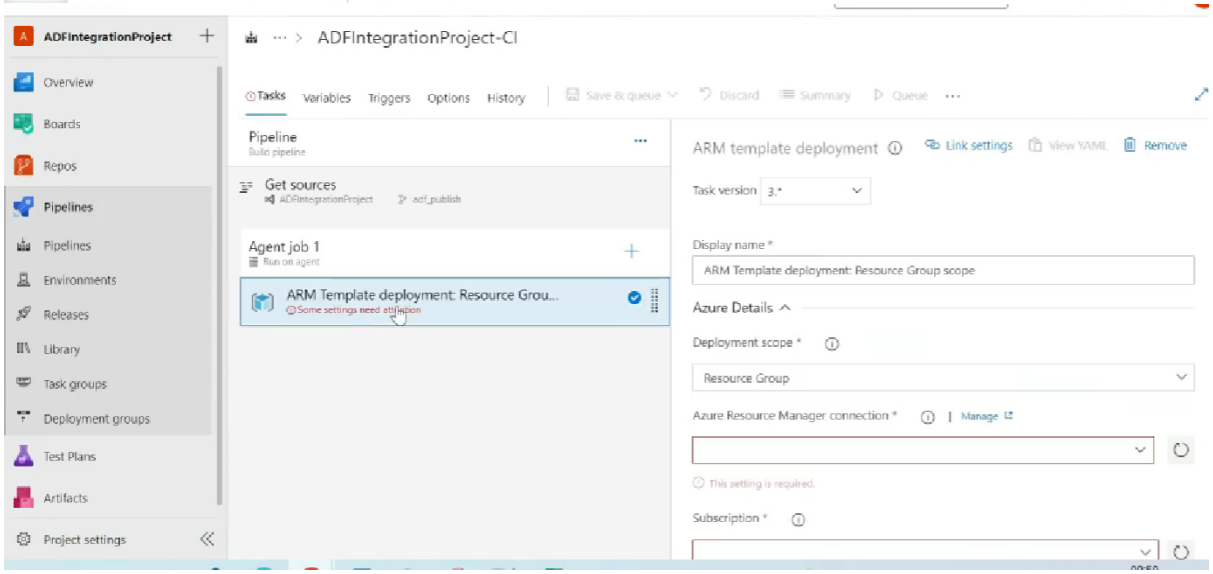
Manage-> git configure -> specifiy organization name, repository name, branch name to sync all pipeline codes into the branch.(integration process)
for practice: in azure DevOps create an organization name and then use those in git configure in ADF

Git contains many services like overview,dashboard,pipeline(deploy),boards(agile jeera),repos(repositories)… etc.,

after integration with git when we click on publish it will create one file(in publish branch) which contain everything including pipelines,dataset, link service etc., and this file we can use for deployment in future

Deployment Pipelines are created by using pipeline tab in azure devops repository,
from there empty job and select arm template and environment is selected by necessary options like resource group etc., in which our arm template code need to be deployed and used .
deployment parameters can be changed by changing in the parameters section(we can change parameters matching to another environment ex: production from dev) and then we can then give the deployment name and save the pipeline.





Note:

Publish branch contains all codes of dataset link into one code. Contains in linked templates, armtemplatefact.json mostly…….

Deployment mode: incremental only changed pipelines get deployed,, complete total pipelines get deployed irrespective of changes.

Best practice: maintaining 3 different azure accounts for different environments like dev,testing,prod is the best practice.

INTEGRATION RUN TIME(Generally called as “IR”):

Integration runtime is a software installed in a machine which gives the computational power to run the pipelines in ADF.

There are 3 types:

Note:

Generally we never gonna create IR since it is a costing issue probably done by infra team.

AZURE IR:

Auto resolve integration runtime it is generally default IR suggested by azure adf… when this is used it first try to allocates IR belongs to same destination region.

Else secondly tries to allocate present in near destination region or else if it is not able to identify destination region it allocates adf created region as IR region.

We can also set the region while creation if we need to avoid any compliance(privacy) issues.

SELF HOSTED IR ： (important)

This is used to connect to datasources that or on prem or that has private network (which are outside azure environment or network).

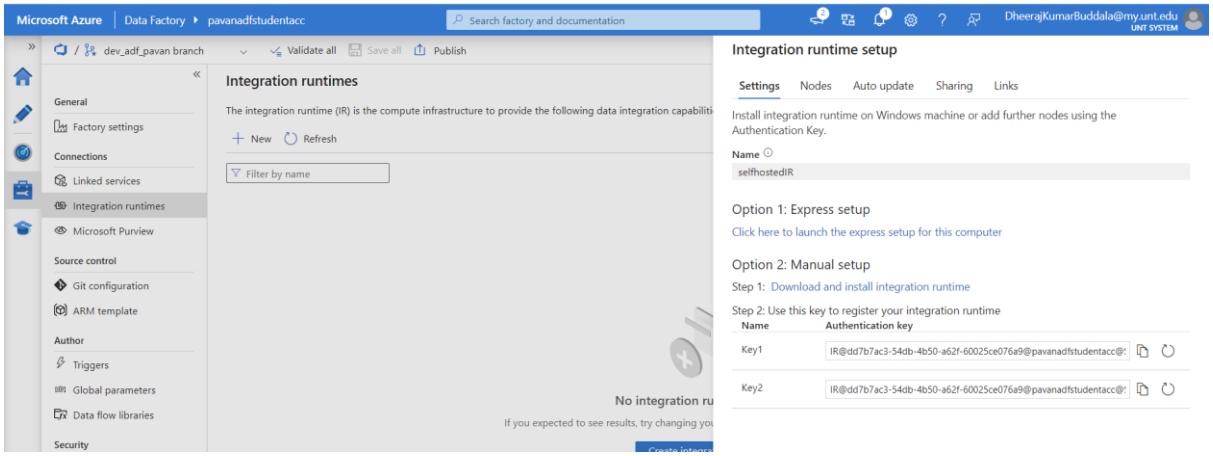
Generally to connect to private network we need to raise request to on-prem to allow request from our side but since our ip address changes frequently as resources are allocated by azure dynamically(based on region we specified while creation) to resolve this problem…

How SHIR is Used in ADF:

1. **Installation**: SHIR is installed on an on-premises machine or a virtual machine within a private network that needs to access ADF.
2. **Linked Services**: Once SHIR is installed, it's configured within ADF as a linked service. This allows ADF to interact with on-premises data sources through the SHIR.
3. **Pipelines**: When you create pipelines in ADF to move or transform data, SHIR acts as the bridge between ADF in the cloud and your on-premises or private data sources.
4. **Authentication**: SHIR uses credentials and authentication methods (like SQL Authentication or Windows Authentication) to securely connect to on-premises data sources, ensuring secure data access.

Data Movement: For example, if you're moving data from an on-premises SQL Server to Azure Blob Storage, the SHIR acts as the intermediary that moves data securely between these environments.

Setup is not done by us since it is a costing issue. But very important to know.



Note:

1. Node is just a machine name and can have multiple nodes(UP TO 4) connected with same self hosted IR

SSIS IR:

SQL server integration services – it is a microsoft etl product just as adf where pipelines can be created, SSIS packages are developed using sql server data tools or visual studio.

SSIS IR used to execute SSIS Packages both in cloud environment or private network ,,, this IR is specifically given to execute SSIS packages.
specifically designed for compatibility(SSIS)

Note:

Linked- self host: we can share the IR created in one adf account to another adf account.

Big data: (HADOOP = HDFS+MAP REDUCE)

after digitalization(ex: website creation etc.,) managing the huge amount of data, data engineer roles has came to the lime light

There is no proper definition for big data but generally referred as huge amount of data where it ranges from few gbs to few terabytes.

There are three types of data:

Structured: has fixed columns and this data is analyzed using a simple SQL query.(MYSQL,oracle)

Semi-structured: somewhat fixed columns will be there but its not that strict to be followed, will depend on data input.(MongoDB, JSON)

Unstructured: no specific structure for data it is completely random.(textfiles,logs,images,videos)

Note:

As a data engineer its our responsibility to gather relevant data and keep it in one file using pipelines and also sometimes need to do analytics to get insights of data which will give better results.

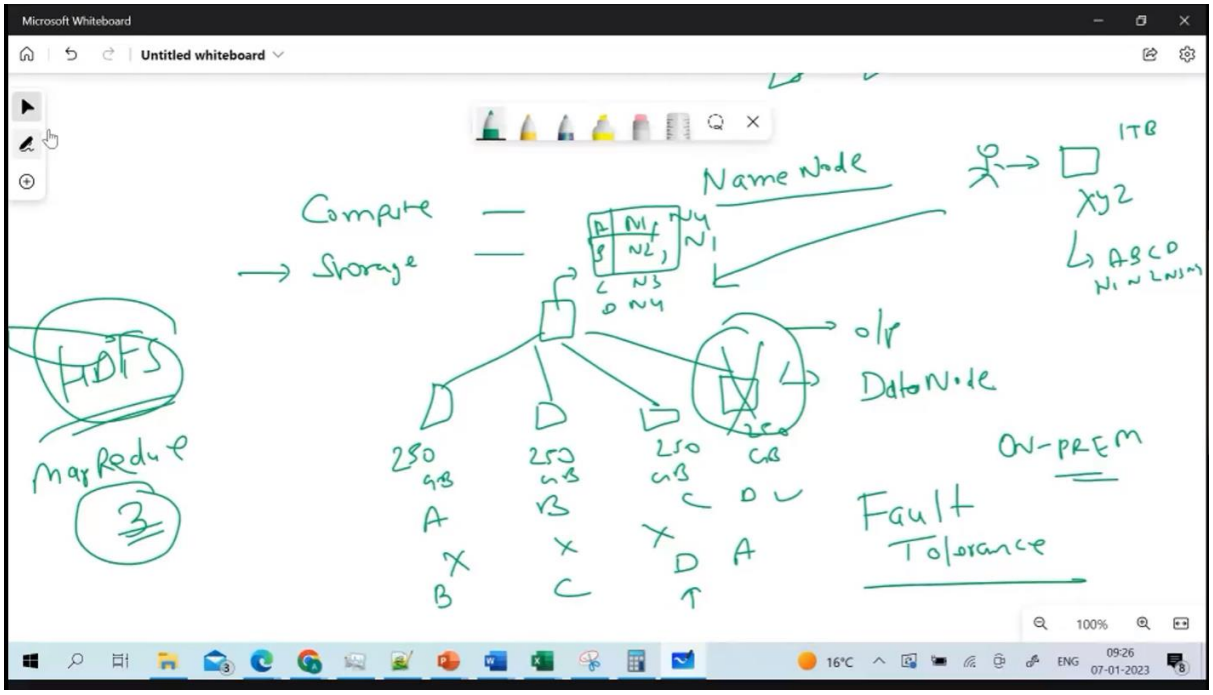
To analyze data we need 2 main things
1.Storage
2.Compute
both are required to process huge amount of data,, node is a machine/laptop whereas cluster is a group of machines that are interconnected.

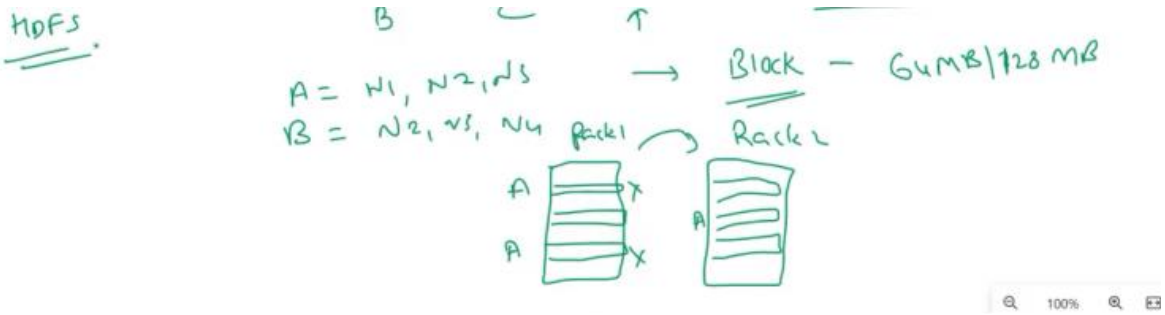
HDFS working(storage of data):
This framework was developed before existence of cloud, it was developed based on machines on-prem.

Hadoop uses HDFS(Hadoop distributed file system) which distribute the big file into different blocks and give it to the different nodes(machines to work properly) Hadoop uses clusters(collection of machines to run this process)
To remove the Fault Tolerance(loosing data if one machine is down) each node contains multiple block copies of data(distributed data so there will be no loss in data and we can analyze entire file completely),,,,, each Node generally contains default 3 copies of same data in multiple Nodes(in same cluster) and default block size of data is 64mb/128mb but these are customizable according to needs.
ex: machine(Node) A can have data copy of machine B,D

“The details of which set of data(block of data) allocated to which machine(node) will be in the “**name node**” and the node where actual data resides is called as **data node**”.

Note:
Hadoop is such a smart such that it will try to duplication data between servers keeping Fault Tolerance in mind





MAP REDUCE(MR Jobs logic based):

Data locality: try to run code where data is already in local, this will decrease or avoid the cost of data transfer
ex: when we formed a cluster the new data is automatically divided into the cluster machines(using HDFS) rather than cluster machines copying data after some times

Here we have 2 phases:
1.mapping phase: where we write code that code is mapped to local machines and run the data analytics on logic given by us
2.reduce phase: here we write code to “integrate the data” from mapping phase.

And these mr job containing code for map and reducing is passed through the cluster where passes through different machines our distributed data is stored.

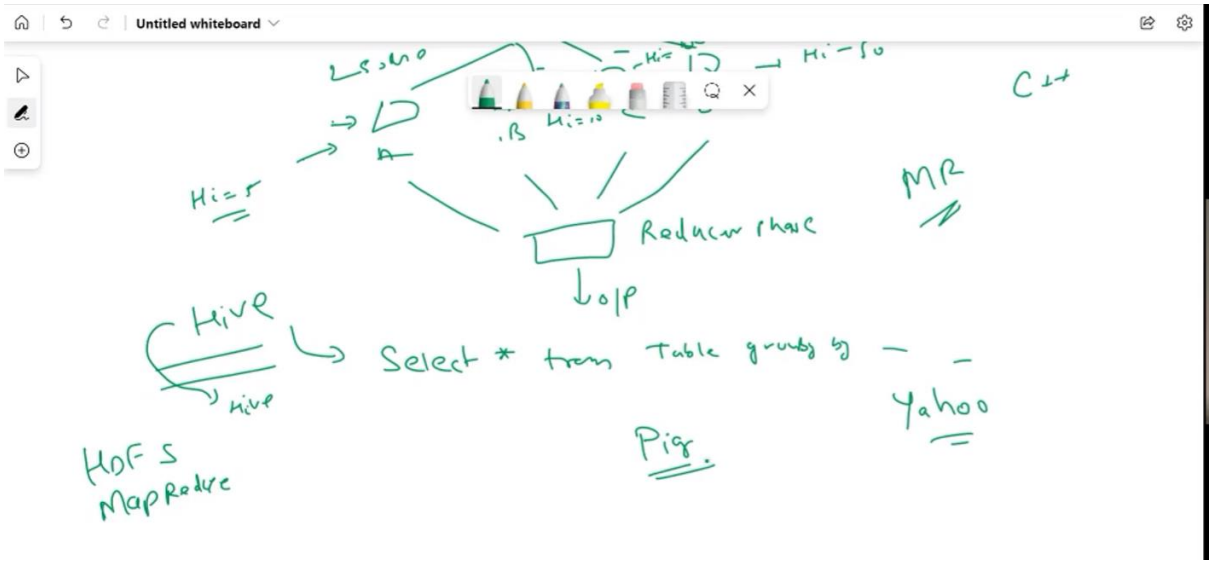
Mr job language support – java,python..

Mr job code writing is complex(to write hello word we need to write 50 lines of code) so in order to overcome this situation hive(founded by facebook) and pig(founded by yahoo) are being used as interpreter where write the code in sql using hive it will automatically convert it into MR CODE.

Another tool named “Sqoop” used to copy data from hdfs to rdms database.

Note:

Both mr and hdfs both are written on java.



SPARK/PYSPARK/DATABRICKS:

Hadoop Drawbacks and Solution:

1. MR job code is more complex (NOT userfriendly to understand)
2. MR job may contains sequence of steps and output of every step will be stored in harddisk (instead of RAM – Main Memory), so every time intermediate result of each step is stored in the hard disk.

   SPARK can resolves the these drawbacks from Hadoop,it can replace the drawback of the “MR JOBS” (computational Drawback of Hadoop Framework).
   here output of files will remain in main memory unlike in hard disk and could be again used there itself and do analytical . but unlike the Hadoop distribution frame work, spark supports ADLS,HDFS,S3(storage solutions) for the data distribution.

   SPARK(Open source—anybody can install and run in their system) supports python,java,scala,r programming languages.
   pyspark is nothing but writing spark code using python language.

   “DATABRICKS” is developed by spark developed employees with more optimization and 5 times faster than the spark it provides the “spark cloud solution” provider where we can run spark code without installing spark software. And also has some additional features.

Data Bricks Properties:

We can create databricks community account for free(1 year) but most probability use azure databricks paid account in project.

Notebook:
Notebook creation is nothing but a note pad which is useful to write codes AND it is auto saved.
it works same as google collab like cells concept

Cluster(compute):
We can create databricks cluster from compute option where we can configure properties like no of machines, memory size etc., and data bricks has its own runtime called data bricks run time.

Workspace:
Workspace is nothing but having directories or folders of all the files we have created just like adls.
Workflows used to give specific role based controls.
Repos(premium feature): connecting to git hub
Data:
Data is used to see tables or dbfs (data base file system is just like adls, amazon s3 storage sol for spark).
Here we can insert new data as well.

Note:
Databricks runtime is just like integration runtime gives computational power for spark cluster.
Databricks doesn't support java like spark.

Notebook Features:
have view: used to represent the notebook in different ways like results only, standard, side by side
history: the history is tracked here (auto saving features is enabled)
code comment: we can give the code comment here for documentation
edit common like same insert and delete cells find and replace
File can help export or download the code and import code common functionalities

Spark syntax and functions:

Python code: df = spark.read.csv(""). (# to comment)
spark(session object).read.csv → spark is the session object to read spark utilities, read.csv is reading file
In Spark whenever you read you will get the data frame(it is the representation of data in terms of table).
data frames are immutable(you can point variables to new data frame though same as python concept)
ctrl + space → for auto suggestions

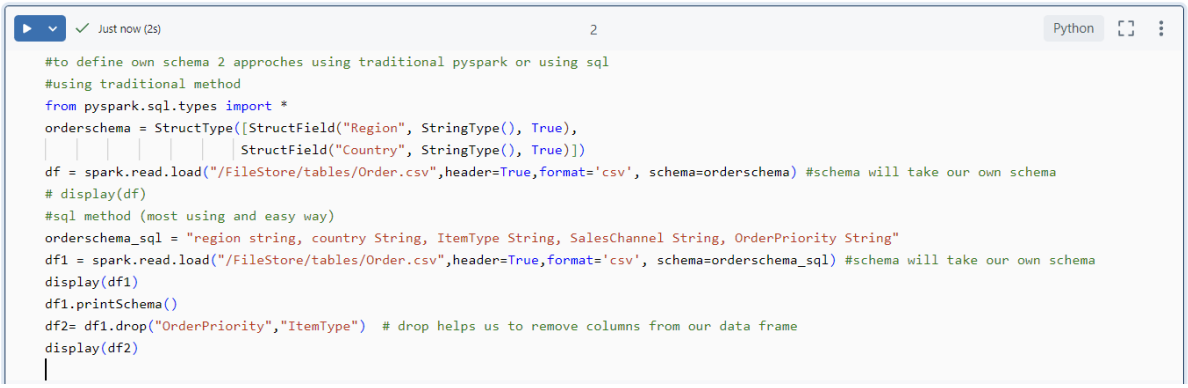
df.show() → display first 20 rows.
display(df) → gives proper table format. (spark specified function)

Header = True, sep=',' allocates the first line as a header based on comma as separator.

df.printschema() → print schema of the data.
infer schema = True → Analyze the entire data and gives the correct data type(used for dynamic schema or we may not aware of schema).

<u>Defining own schema:</u>
we can define own schema structure: we have two approaches, traditional programmatic pyspark and sql approach
Pyspark:
from pyspark.sql.types import *
orderSchema = StructType([StructField("Region", StringType() ,True)])
SQL:
orderSchema = 'Region String ,Country String ,ItemType String'

df = spark.read.load("/FileStore/tables/Order.csv",format="csv", header=True, schema=orderSchema)
df.printSchema()



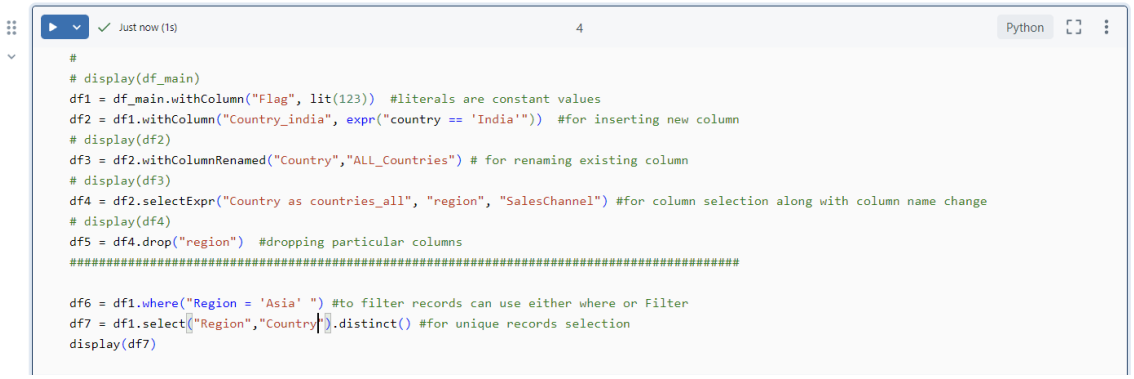
creating data frame manually:
selecting columns in data frame: (can do in 5 different ways but all does the same thing pick one ur

choice)

```
#creating data frame manually
# we need columns(schema) and rows for creating data frame
# from pyspark.sql import *
from pyspark.sql.functions import *
row1 = Row("Pavan", 1, 50000)
row2 = Row("Sai", 2, 75000)
employee_schema = ("name String, employee_id Integer, employee_salary Integer")
df = spark.createDataFrame([row1, row2], schema=employee_schema)
display(df)

#for selecting particular columns 5 methods
df1 = df.select("name", "employee_id") #for selecting particular multiple columns (popularly used)
df2 = df.select(col("name")) #drawback -- col,column,expr only can select 1 argument
df3 = df.select(column("name"))
df4 = df.select(expr("name")) #popularly used
df5 = df.select(df.name,df.employee_salary)
display(df5)
# display(df4)
```

adding column, changing column name and dropping the column
To filter particular records
To pull unique records(distinct)



Note:

Everything is case sensitive and use small letters mostly.
Infer schema will take more time and computational power.

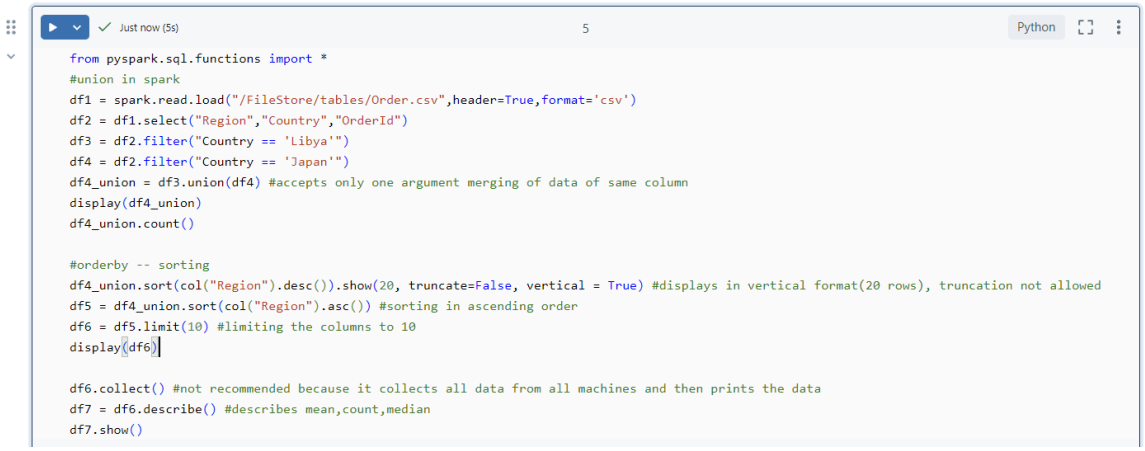
---

parquet files has better compression and readability ratio for cluster machines compared to other file formats so this is most highly used and default format for spark.

Union of dataframes(df1.union(df2)) (merge the data in same column)
Orderby → sorting data in the data frame (sort, orderby can adjust asc,desc)
show(100,truncate=False,vertical=true) → can display top 100, without data truncation and can see the data in vertical manner as well)
limit,count → data frame functions

Collect → print all the records basically collects all records from all nodes which may result to the memory out in driver node not recommended to use at all

Describe → gives the dataframe which has parameters like mean,median,max,min etc.,

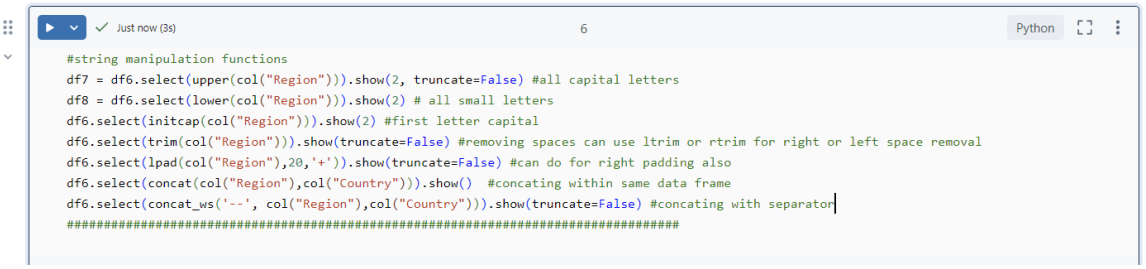


```
from pyspark.sql.functions import *
#union in spark
df1 = spark.read.load("/FileStore/tables/Order.csv",header=True,format='csv')
df2 = df1.select("Region","Country","OrderId")
df3 = df2.filter("Country == 'Libya'")
df4 = df2.filter("Country == 'Japan'")
df4_union = df3.union(df4) #accepts only one argument merging of data of same column
display(df4_union)
df4_union.count()

#orderby -- sorting
df4_union.sort(col("Region").desc()).show(20, truncate=False, vertical = True) #displays in vertical format(20 rows), truncation not allowed
df5 = df4_union.sort(col("Region").asc()) #sorting in ascending order
df6 = df5.limit(10) #limiting the columns to 10
display(df6)

df6.collect() #not recommended because it collects all data from all machines and then prints the data
df7 = df6.describe() #describes mean,count,median
df7.show()
```

String and date functions:
upper,lower,initcap(camelCase),concat(using separator concat_ws),trim(removing the spaces), padding
---



```
#string manipulation functions
df7 = df6.select(upper(col("Region"))).show(2, truncate=False) #all capital letters
df8 = df6.select(lower(col("Region"))).show(2) # all small letters
df6.select(initcap(col("Region"))).show(2) #first letter capital
df6.select(trim(col("Region"))).show(truncate=False) #removing spaces can use ltrim or rtrim for right or left space removal
df6.select(lpad(col("Region"),20,'+')).show(truncate=False) #can do for right padding also
df6.select(concat(col("Region"),col("Country"))).show()  #concating within same data frame
df6.select(concat_ws('--', col("Region"),col("Country"))).show(truncate=False) #concating with separator
################################################################################
```

spark.range(n) → create a data frame with n no of rows
current_date, current_datetimestamp → gives current date and time
date_sub, date_add →(date addition and subtraction)
to_date → change data to date datatype(if data format doesn't match replace with NULL)
can extract only hour,month,week using week,hour,day

```
from pyspark.sql.functions import *
#date manipulation functions
#range describes how many rows should be in new_data frame
date_df = spark.range(10).withColumn("Date_current",current_date()).withColumn("Time_stamp",current_timestamp())
date_df.show(truncate=False)
date_df.select(date_add("Date_current", 5), date_sub("Date_current", 5)).show()  #date addition or subtraction by constant

#converting to date data type
const_df = spark.range(2).withColumn("Date",lit("2024-05-20"))
const_df.show()
const_df.printSchema()
updated_df = const_df.select(to_date(col("Date")).alias("new_date")) #to_date converts string to date format,data should be date format
updated_df.printSchema()
updated_df.select(date_add(col("new_date"),5).alias("new_date")).show()
updated_df.select(month(col("new_date"))).show() #year,month,day can be extracted
updated_df.select(dayofmonth(col("new_date"))).show()
#can also extract the minute,hour, second if it is a time stamp format
```

Operations in Data bricks:
There are 2 Types:
Transformations, Actions

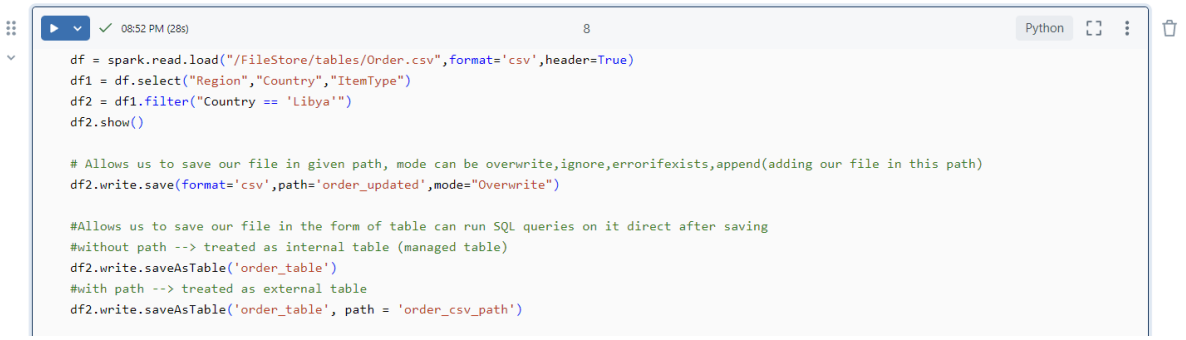
Transformations:
Changing the Data frame, can be Anything GroupBy, select etc.,
In general , Transformations are lazy in nature
ex: instead of going to market every time for products getting all products at one shot at evening. (spark has built in Transformation optimization) it will help it to use the existing data frame data while creating new data frame it assumes this is not final output…
df1,df2,df3 here df3 can be formed by using df1, df2 data

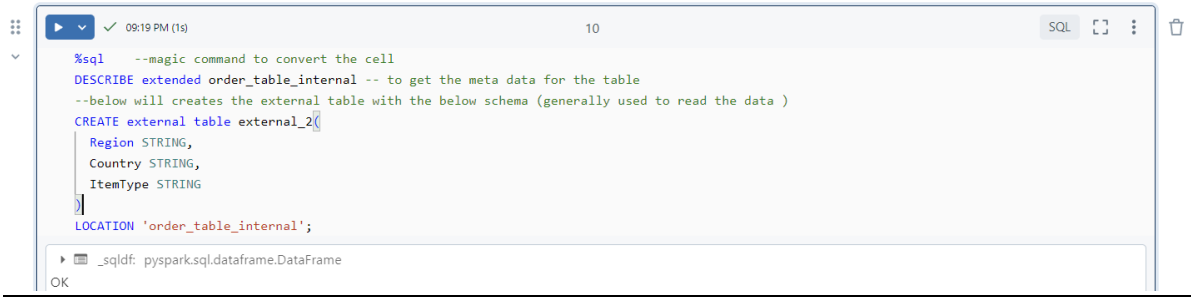
Action:
Here spark job is triggered,, here we are doing some actions on our output like (show,save,take etc.,)

<u>Storing output Data:</u>
File Format:
df.write.save(path="Folder_Location", mode='',format='')
→ For saving the data frame in data bricks(DBFS), default mode is overwrite, we do have append,ignore,error if exists mode types.
Table Format:
df.write.saveAsTable("Table_name") → we can run sql commands on these table directly
data stored in (/users/hive/Warehouse)
Here data is still stored in terms of File but to us it appears as table

<u>Table is saved using 2 methods:</u>
Managed(Default-Internal): Here data is deleted when we dropped the table(spark has full control).
External(Path is given): Here data is not deleted even when we drop the table(because table data is deleted from meta store but file is untouched—spark has no control on external path)
describe extended "Table_name" to know properties of ur table

External table can be done in 2 types:
one giving file path, another creating external table





Meta store:

whenever we create tables in spark

The meta data of the tables will be stored in as Meta store, by default it uses debry database

Magic Commands → %sql can converts cell to sql(Any language) eventhough notebook is python

Note:

In spark we can write sql command either on direct sql notebook or converting cell to sql or using spark.sql("").

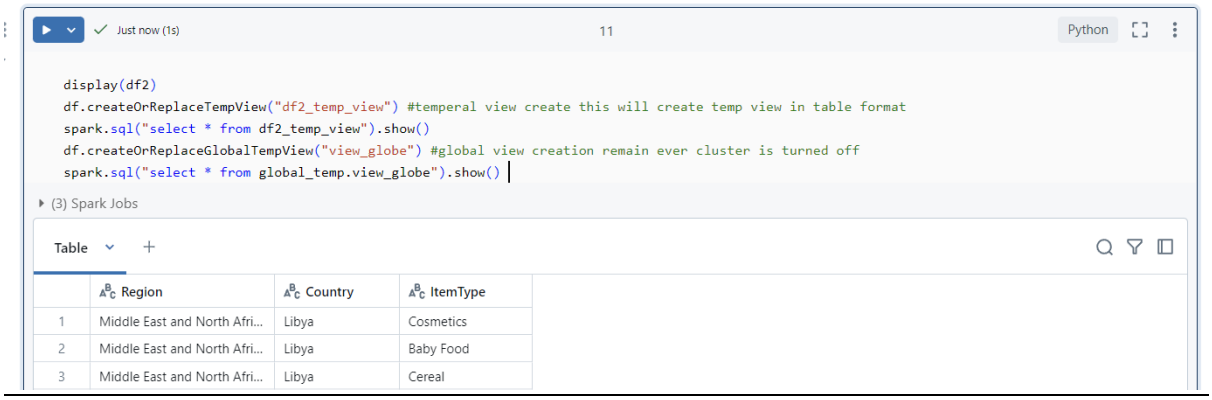
Views:

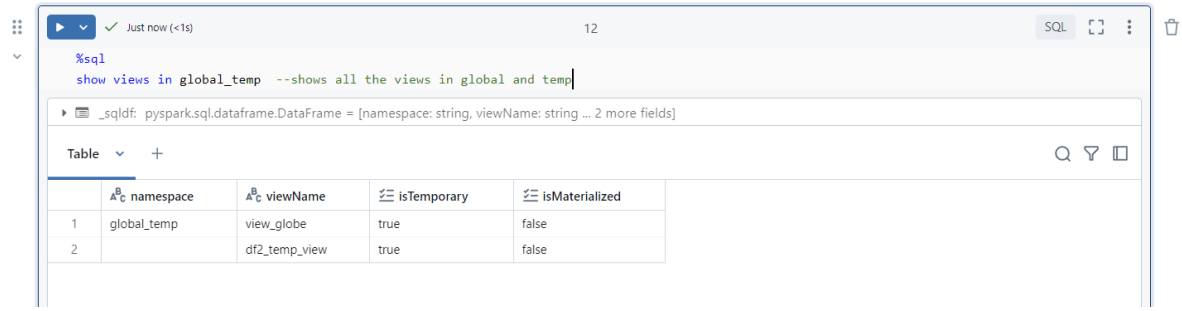
They allow you to treat the data within a DataFrame as if it were a table in a relational database, enabling SQL-like queries directly on DataFrame data. Views are not stored physically in the filesystem or database but exist only for the duration of the Spark session

Types : 1.temparory, 2.permanent

If ur files are present even after cluster is disconnected it is called permanent view else called temporary view.

Temporary views are 2 types
1.localtemp, 2.globaltemp ->
local view available only In that session or notebook.
Global can be available across different notebooks.
→df.CreateorReplaceTempview("view_name"),,, select * from view_name
→df.CreateorReplaceGlobalTempview(view_name),,, select * from global_temp_view_name
→ show views;





Note:
If data changed in temp_view it doesn't affect the original data frame it was created

<u>MOUNTING:</u>

Attaching databricks to adls is called mounting this is one time process done through config Notebook even though cluster is disconnected mounting will still be on.

Mounting is done through container level (1 mount = 1 container mapping).
dbutilities(databricks specified helps to mount)
dbutils.help → used to get the available options of this utilities -- mounting(dbutils.fs.mount) if forgot the path can check this in this help

dbutils.fs.mount(source="source_link(container,storageaccount)", mount="mounting_path", extra_configs="SAS token")

%fs → magic command for file system
ls mount_point → (list mount_point) gives all files or folders in our container

Mounting can be done either user SAS(privacy concerns exposing SAS) or AAD

Connecting through aad(azure active directory):

AAD allows to pass through it and access the container, For this create app registration(user id will be generated), generate secret for application created, use app id, secret id, tenant id for mounting and more importantly give access to this app created in storage account IAM services.

Even though it can resolve the problem of direct SAS token visibility, still app userid, secret are publicly visible In order to overcome the problem we can use key vault.

Mounting of ADLS contianer Using Application Registration

```
[ ] # Container name
    # Stoage account name
    # Tennant Id
    # Application Id
    # Secret value
    # Mount point name (it could be anything /mnt/.....)

    #Code Template

    configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": "<application-id>"",
           "fs.azure.account.oauth2.client.secret": "<service-credential-key>",
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/<tennant-id>/oauth2/token",
           "fs.azure.createRemoteFileSystemDuringInitialization": "true"}

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


    #Example code


    configs = {"fs.azure.account.auth.type": "OAuth",
           "fs.azure.account.oauth.provider.type": "org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider",
           "fs.azure.account.oauth2.client.id": "d9c97ba4-8943-4b06-b9d5-eb239baf9113",
           "fs.azure.account.oauth2.client.secret": "uwsajdbwj3u392smsknas",
           "fs.azure.account.oauth2.client.endpoint": "https://login.microsoftonline.com/3cc553a5-eec4-4b67-9863-7e488c9fa455/oauth2/token",
           "fs.azure.createRemoteFileSystemDuringInitialization": "true"}
```

Connecting through key vault:

Secret Scope: used to connect to key vault and fetch the secret pass by secret name

Need to create scope in azure databricks(scope creation not allowed for community databricks) extend databricks url after # with secrets/createScope (hidden menu in Azure data bricks) and then create secret scope,

in key vault we can store sas token and pass it using secret name

Mounting of ADLS contianer Using SAS token with secret scope and key vault

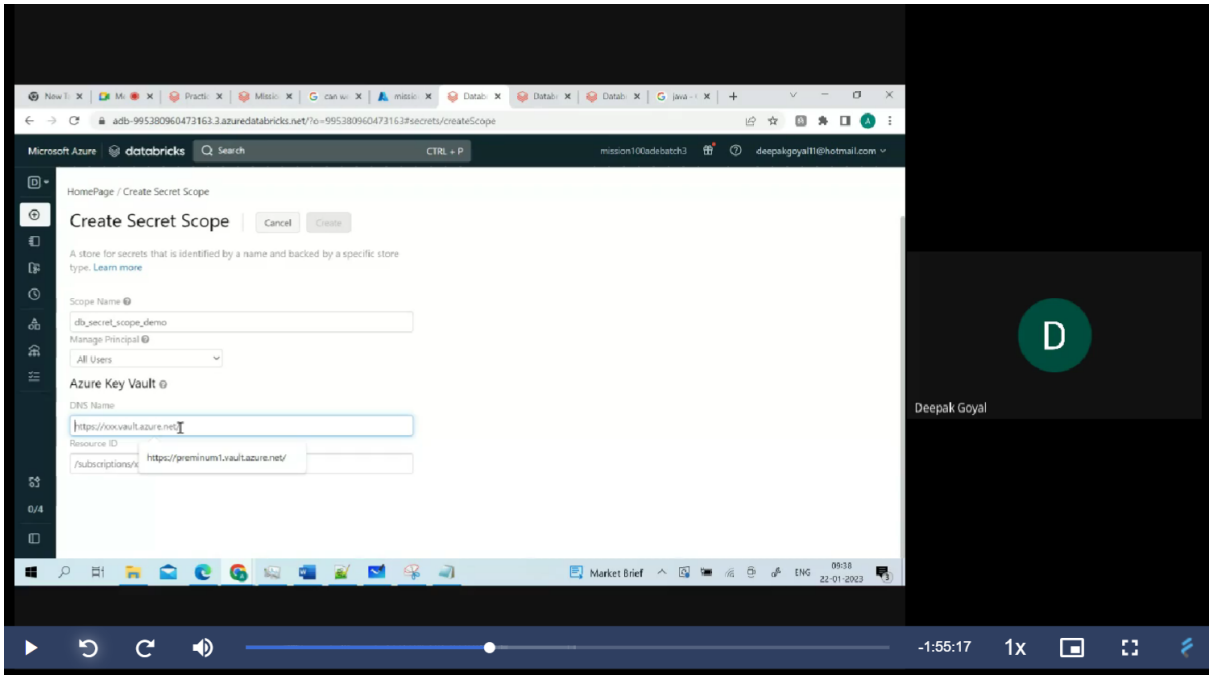
```
# Container name
# Stoage account name
# Mount point name (it could be anything /mnt/.....)
# Databricks secret scope name
# Azure key vault key name containing the secret info

#code Template


dbutils.fs.mount( source = 'wasbs://<conatiner-name>@<storage-account-name>.blob.core.windows.net',
                  mount_point= '<mount-point-name>', extra_configs ={'fs.azure.sas.landing.mission100batch2adls.blob.core.windows.net': dbutils.secrets.get(scope="<databricks-secret-sc



# Example will be like this
dbutils.fs.mount( source = 'wasbs://landing@mission100batch2adls.blob.core.windows.net',
                  mount_point= '/mnt/abc2', extra_configs ={'fs.azure.sas.landing.mission100batch2adls.blob.core.windows.net':dbutils.secrets.get(scope="dbscopemissionbatch2",key="dbse
```


Microsoft Azure
databricks
Search
CTRL + P
HomePage / Create Secret Scope
Create Secret Scope
Cancel
Create
A store for secrets that is identified by a name and backed by a specific store type. Learn more
Scope Name
db_secret_scope_demo
Manage Principal
All Users
Azure Key Vault
DNS Name
https://xx.vault.azure.net
https://preminum1.vault.azure.net/
Resource ID
/subscriptions/x
0/4
Market Brief
ENG
09:38
22-01-2023
D
Deepak Goyal
-1:55:17
1x