Azure Databricks Mastery:

Hands-on project

with

Unity Catalog,

Delta Lake,

Medallion Architecture

Azure Databricks Free notes

Azure Databricks end to end project with Unity Catalog

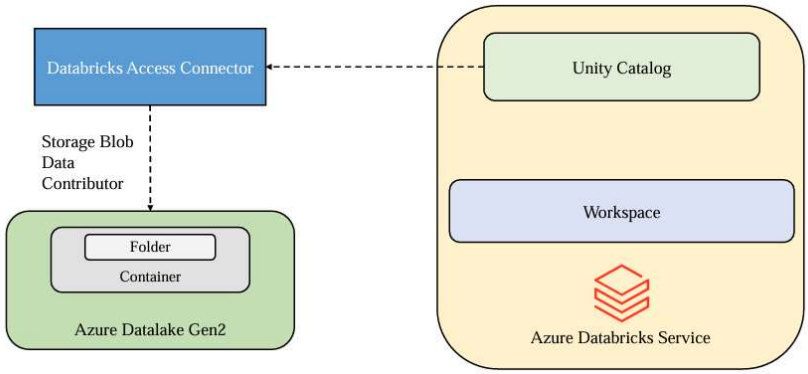
Azure Databricks Mastery: Hands-on project with Unity Catalog, Delta lake, Medallion Architecture

Day 1: Create DataBricks resource using Azure Portal

Environment Setup: Login to your Azure Portal

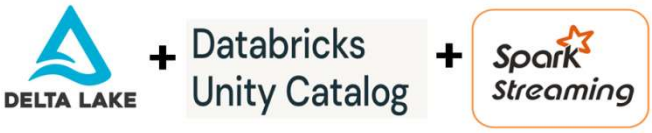
Step 1: Creating a budget for project: search and type budget, "ADD" on Cost Management, "Add Filter" in "Create budget", select Service Name: Azure Databricks in drop down menu.

Step 2: Set alerts as well in next step. Finally click on "Create".



Step 3: Create a Databricks resource, for "pricing tier", click here for more details: https://azure.microsoft.com/en-us/pricing/details/databricks/

Hence select for Premium (+ Role based access controls), skip "Managed Resource Group Name", not any changes required in "Networking", "Encryption", "Security", "Tags" also.



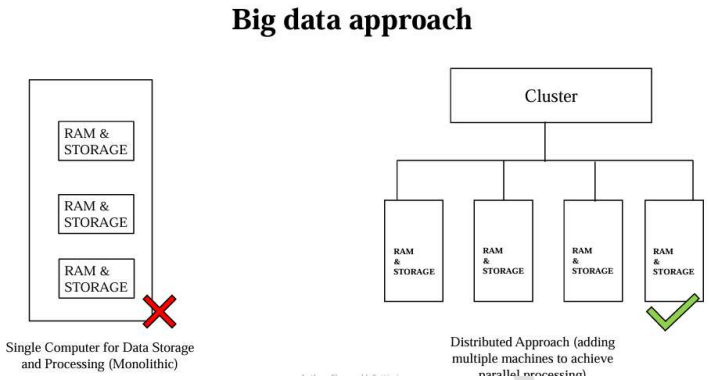
Continuous Integration + Continuous Deployment

Step 4: Create a "Storage Account" from "Microsoft Vendor", select "Resource Group" as previous one, "Primary Service" as "ADLS Gen 2", select "Performance" as "standard", "Redundancy" as "LRS", not any changes required in "Networking", "Encryption", "Security", "Tags" also.

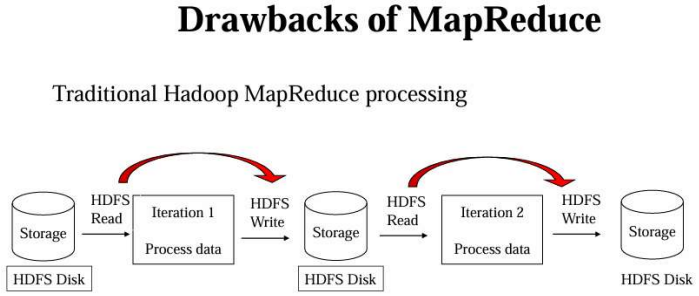
Step 5: Walkthough on databricks Workspace UI: click on "Launch Workspace" or go through URL: looks like https://_____azuredatbricks.net, Databricks keep updating UI, click on "New" for "Repo" as CI/CD, "Add data" in "New", "Workflow" are just like Pipeline at high level, "Search" bar for searching also.

- **Databricks admin types: https://learn.microsoft.com/en-us/azure/databricks/admin/**

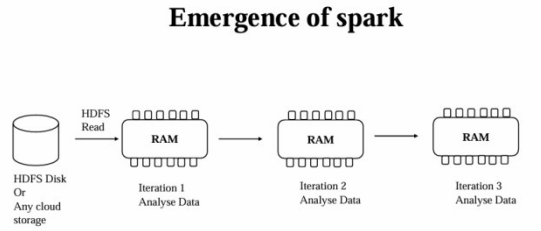
Theory 1: What is Big Data approach?: Monolithic is used for Single Computer and distributed Approach using Cluster which is group of computers.



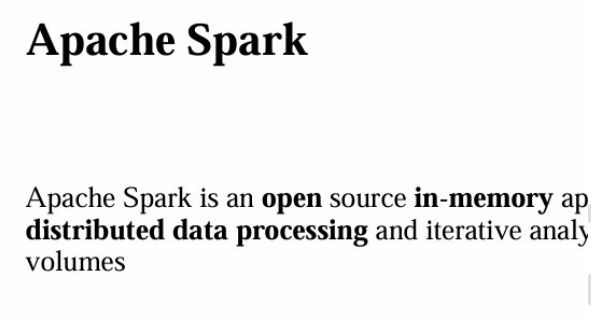
Theory 2: Drawbacks of MapReduce: In HDFS, in the each iteration, Read and Write operation from disk which will take place high I/O disk costs, developer also have to write complex program, Hadoop is only single super Computer.



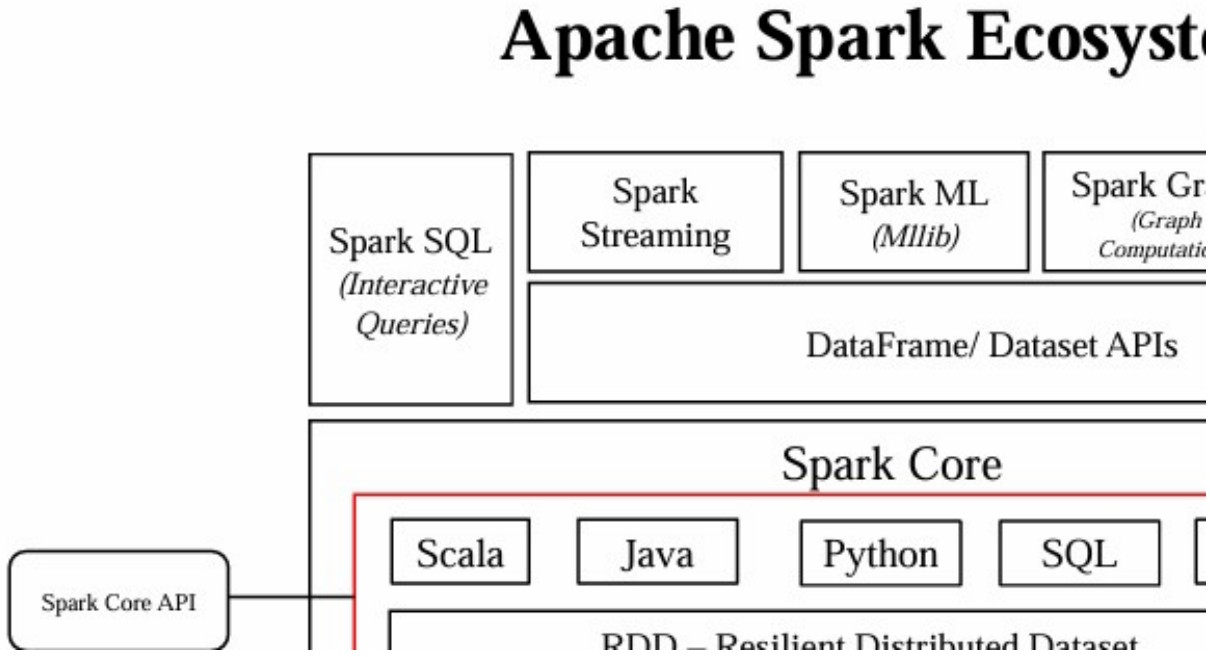
Theory 3: Emergence of Spark: First it uses HDFS or Any cloud Storage then further process takes place in RAM, it uses in-memory process which is 10-100 times faster than Disk based application, here database is detached from memory and process aloof.



Theory 4: Apache Spark: it is an in-memory application framework.



Theory 5: Apche Spark Ecosystem: Spark Core, special data structure RDD, this is collection of items distributed across the compute nodes in the cluster, these will be processed in parallel, but RDDs are difficult to use for complex operations and they are difficult to optimize , now we are making use of Higher level APIs and libraries like Data Frames and Data Set APIs. Also, uses other high level APIs like Spark SQL, Spark Streaming, Spark ML etc.



In the real time, we do not use RDD but higher level APIs to do our programming or coding, data frame APIs to interact with spark and these data frames can be invoked using any languages like Java, Python, SQL or R and internally spark has two parts: set of core APIs, and the Spark Engine: this distributed Computing engine is responsible for all functionalities, there is an OS which will manage

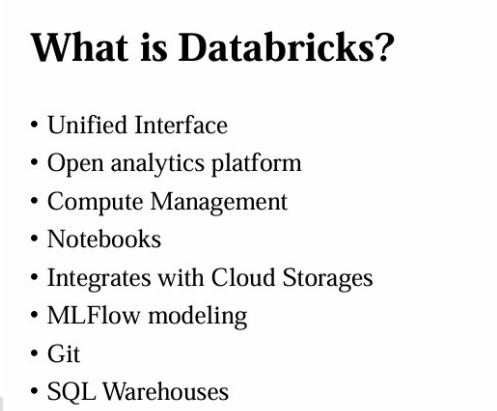
this group of computers (cluster) is called Cluster Manager, In Spark, there are many Cluster Managers in which you can use like YARN Resource Manager or Resource standalone, Mesos or Kubernetes.

So, Spark is a distributed data processing solution not a storage system, Spark does not come with storage system, can be used like Amazon S3, Azure Storage or GCP.

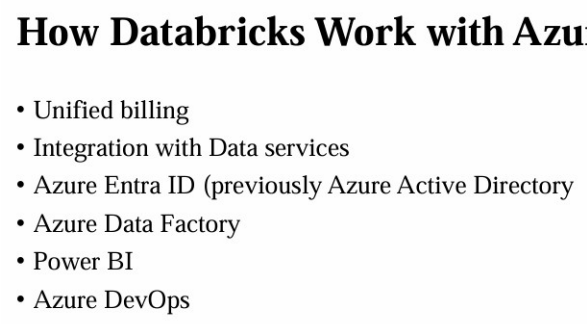
We have Spark Context, which is Spark Engine, to break down the task and scheduling the task for parallel execution.

So, what is Databricks? The founders of the Spark developed a commercial product and this is called Databricks to work with Apache Spark in more efficient way, Databricks is available on Azure, GCP and AWS also.

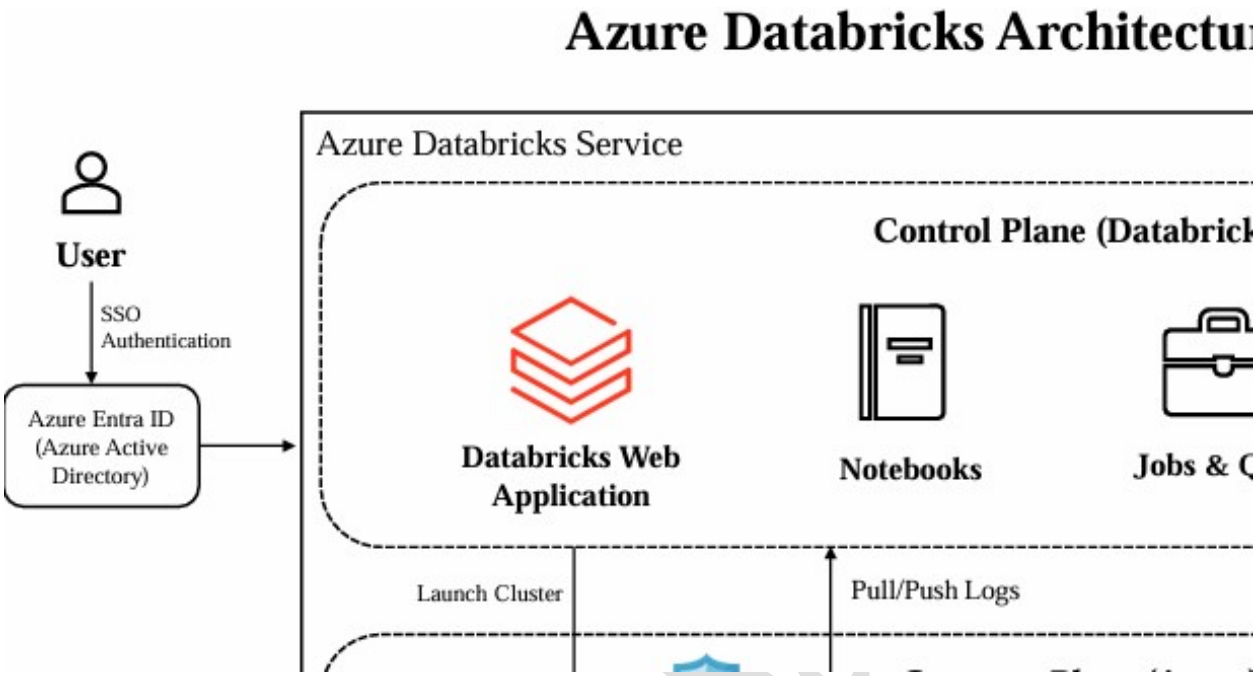
Theory 6: What is Databricks?: DB is a way to interact with Spark, to set up our own clusters, manage the security, and use the network to write the code. It provides single interface where you can manage data engineering, data science and data analyst workloads.



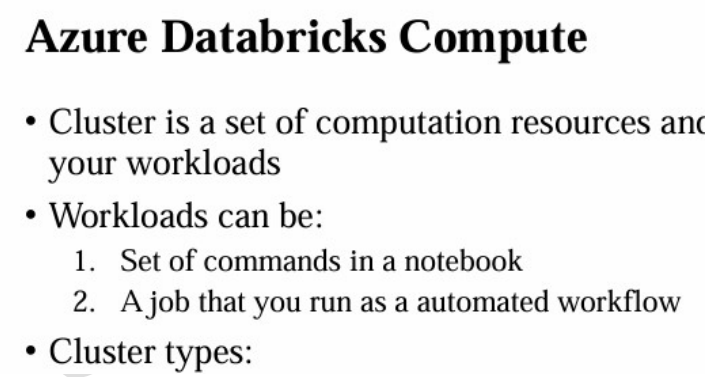
Theory 7: How Databricks Works with Azure? DB can integrate with data services like Blod storage, Data Lake Storage and SQL Database and security Entra ID, Data Factory, Power BI and Azure DevOps.



Theory 8: <u>Azure Databricks Architecture:</u> Control plane is taken care by DB and Compute Plane is taken care by Azure.



Theory 9: Cluster Types: All purpose Cluster and Job cluster. Multi-node cluster is not available in Azure Free subscription because it's allowed to use only maximum of four CPU cores.



In DB workspace: (inside Azure Portal), "create cluster", select "Multi-node": Driver node and worker node are at different machines. In "Access mode", if you will select "No isolation shared" then "Unity Catalogue" is not available. Always uncheck "Use Photon Acceleration" which will reduce your DBU/h, can be seen from "Summary" pane at right top.

Cluster Types

1. **All purpose Cluster**
   - To interactively run the commands in your notebook
   - Multiple users can share such clusters to do collaborat
   - You can terminate, restart, attach , detach these cluster
   - You can choose
     - Multi-node cluster = Driver node and executor nodes w
     - Single node cluster = Only there will be a single driver

Theory 10: Behind the scenes when creating cluster: click on "Databricks" instance in Azure portal before clicking on Databricks "Launch Workspace", there is "Managed Resource Group": open this link; there is a Virtual network and Network security group and Storage account.

Cluster Access modes

Access Mode	Visible to user	UC Support	Supported Lang
Single user	Always	Yes	Python, SQL, Sc
Shared	Always (**Premium plan or above required**)	Yes	1. Python (on D Runtime 11.1 2. SQL, 3. Scala (on Un enabled clust Databricks R

This Storage account is going to store Meta Data of it, we will see Virtual Machine, when we will create any compute Resource, now go to Databricks workspace, create any compute resource and then come back here, will find some disks, Public IP address and VM. For all these, we will be charged as DBU/h.

Cluster Runtime version:

- Databricks Runtime is the set of core components that run on your clusters

So which version to use?

- **For all purpose compute:**
  - Databricks recommends using the latest Databricks Runtime version.
  - Using the most current version will ensure you have the latest optimizations and most up-to-date compatibility between your code and preloaded packages.
- **For Job compute:**
  - As these will be operational workloads, consider using the Long Term Support (LTS) Databricks Runtime version.
  - Using the LTS version will ensure you don't run into compatibility issues and can thoroughly test your workload before upgrading.
- **For ML Workloads:**
  - For advanced machine learning use cases, consider the specialized ML Runtime version.

Author: Shanmukh Sattiraju

Cluster policies (in Unity Catalog)

- Policies are a set of rules configured by admins
- These are used to limit the configuration options available to users when they create a cluster
- Policies have access control lists that regulate which users and groups have access to the policies.
- Any user with unrestricted policy can create any type of cluster

Stop our compute resource, nothing is deleted in Azure portal, but when we will click on Virtual Machine, then that will show not "start". But if you delete compute resource from Databricks workspace, check your Azure portal again, will find all resources i.e. disks, Public IP address and VM etc are deleted.

Day 2: Understanding notebook and Markdown basics : Hands-on

Note: this part can be executed in Databricks Community edition, not necessarily to be run in Azure Databricks resource

```
%md
### Heading 3
#### Heading 4
##### Heading 5
###### Heading 6

####### Heading 7
-----------------------------------------------------------------
%md
# This is a comment
-----------------------------------------------------------------

%md
```

1. HTML Style Blod
2. Astricks style **Blod**

%md

Italics style

%md

`print(df)` is the statement to print something

```
This
is  multiline
code
```

%md

- one
- two
- three

%md
To highlight something

 Highlight this

%md
![Profile Pic](https://media.licdn.com/dms/image/C4E03AQGx8W5WMxE5pw/profile-displayphoto-shrink_400_400/0/1594735450010?e=1705536000&v=beta&t=_he0R75U4AKYCbcLgDRDakzKvYZybksWRoqYvDL-alA)

%md
Click on [Profile Pic](https://media.licdn.com/dms/image/C4E03AQGx8W5WMxE5pw/profile-displayphoto-shrink_400_400/0/1594735450010?e=1705536000&v=beta&t=_he0R75U4AKYCbcLgDRDakzKvYZybksWRoqYvDL-alA)

Day 3: DataBricks in Notebook - Magic Commands : Hands-on

Magic commands in Databricks: if any SQL command is to be executed then select 'SQL'.

Note: this part can be executed in Databricks Community edition, not necessarily to be run in Azure Databricks resource

1. Select 'Python' from top and type
print('hello')
#Comments
Default language is Python

Magic commands

- You can use multiple languages in one notebook
- You need to specify language magic command at the beginning of a cell.
- By default, the entire notebook will work on the language that you choose at the top

Magic command	Language	Description
%python	Python	Execute a Python query against Spark Context.
%scala	Scala	Execute a Scala query against Spark Context.
%sql	Spark SQL	Execute a SparkSQL query against Spark Context.
%r	R	Execute a R query against Spark Context.

2. %scala
print("hello") will work and also #comments will also not work.
For comments in Scala use //Comments
---

3. Comments in SQL -- Comments
now in %sql
select 2+5 as sum

4. in %r
x <-"Hello"
print(x)
---

5. There are much more magic commands in DB.
%fs ls
List all things in all the directories inside DBFS ie Databricks File System.

---

6. Know all the Magic commands available:
type:
%lsmagic
---

7. Summary of Magic commands: You can use multiple languages in one notebook and you need to specify language magic commands at the beginning of a cell. By default, the entire notebook will work on the language that you choose at the top.

DBUtils:

 # DBUtils: Azure Databricks provides set of utilities to efficiently interact with your notebook.
Most commonly used DBUtils are:
1. File System Uttilities
2. Widget Utilities
3. Notebook Utilities
---

1. What are the availble utilities?
 # just type:
dbutils.help()
---

2. Lets see File System Utilities
%md
# File System Utilities
# click new cell:
# type:
dbutils.fs.help()

Ls utility

what are available list in particular directory: Enable DBFS, click on "Admin setting" from right top, click on "Workspace Settings",
scroll down, enable 'DBFS File Browser', now you can see 'DBFS' tab, after clicking on 'DBFS' tab, some set on folders are there,
You will find "FileStore" in left pane in “Catalog” button, somewhere, copy path from "spark API format",
path = 'dbfs:/FileStore'

dbutils.fs.ls(path)

why ls, see just above from dbutils.fs.help() details.

# remove any directory:
# just copy following addres from above:such as FileInfo(path='dbfs:/FileStore/temp/', name='temp/', size=0, modificationTime=0)
dbutils.fs.rm('dbfs:/FileStore/CopiedFolder/',True)

# True is added bcs if this file is not exisiting than it will just reply 'True'
# just check directory list again, that file has been removed.
dbutils.fs.ls(path)

---

mkdir

why heading are important bcs, left side "Table of Contents" are there, which showing all the headings

dbutils.fs.mkdirs(path+'/SachinFileTest/')

# list all files so that we can see newly created directory is there or not?
dbutils.fs.ls(path)

put: Inside a folder lets put something,

dbutils.fs.put(path+ '/SachinFileTest/test.csv','1, Test')
---

also check using manual "DBFS" tab
head : read the file conten, which we just written,
filepath = path+ '/SachinFileTest/test.csv'
dbutils.fs.head(filepath)

Copy: Move this newly created file from one location to another
source_path = path+ '/SachinFileTest/test.csv'
destination_path = path+ '/CopiedFolder/test.csv'
dbutils.fs.cp(source_path,destination_path,True)

# display content from recently pasted values
dbutils.fs.head(destination_path)

```
---------------------------------------------------------------------


# same activity can be done by right click of that file *.csv
# with "Copy path", "Move", "Rename", "Delete"

---------------------------------------------------------------------

# Move is cut and paste/move
# copy is just copy and paste

source_path = path+ '/FileTest/test.csv'
destination_path = path+ '/MovedFolder/test.csv'
dbutils.fs.mv(source_path,destination_path,True)
---------------------------------------------------------------------
```


```
# remove folder
dbutils.fs.rm(path+ '/MovedFolder/',True)
dbutils.fs.help()
---------------------------------------------------------------------
```

Day 4: DBUitls -Widget Utilities : Hands-on

Note: this part can be executed in Databricks Community edition, not necessarily to be run in Azure Databricks resource

Why Widgets: Widgets are helpful to parameterize the Notebook, imagine, in real world you are working in heterogeneous environment, either in DEV env, Test env or Production env, to change everywhere, just parameterize the notebook, instead of hard coding the values everywhere.

Details: Coding:

```
# what are vailavle tools, just type:
dbutils.widgets.help()
------------------------------
%md
## Widget Utilities
------------------------------
%md
## Let's start with combo Box
### Combo Box
dbutils.widgets.combobox(name='combobox_name',defaultValue='Employee',choices=['Employee','Developer','Tester','Manager'],label="Combobox Label ")
------------------------------
# Extract the value from "Combobox Label"
emp=dbutils.widgets.get('combobox_name')

# dbutils.widgets.get retrieves the current value of a widget, allowing you to use the value in your Spark jobs or SQL Queries.

print(emp)
type(emp)
------------------------------
# DropDown Menu
dbutils.widgets.dropdown(name='dropdown_name',defaultValue='Employee',choices=['Employee','Developer','Tester','Manager'],label="Dropdown Label")
------------------------------
# Multiselect
dbutils.widgets.multiselect(name='Multiselect_name',defaultValue='Employee',choices=['Employee','Developer','Tester','Manager'],label="MultiSelect Label")
------------------------------
# Text
dbutils.widgets.text(name='text_name',defaultValue='',label="Text Label")
```


```
------------------------------
dbutils.widgets.get('text_name')
```

```
# dbutils.widgets.get retrieves the current value of a widget, allowing you to use the value in your Spark jobs or SQL Queries.


------------------------------

result = dbutils.widgets.get('text_name')
print(f"SELECT * FROM Schema.Table WHERE Year = {result}")
------------------------------
# go to Widget setting from right, change setting to "On Widget change"--> "Run notebook", now entire notebook is getting executed

print('execute theseeeSachin ')
```

Day 5: DBUtils - Notebook Utils : Hands-on

Note: this part can be executed in Azure Databricks resource, not in Databricks Community edition, otherwise it will give like: To enable notebook workflows, please upgrade your Databricks subscription.

Create a compute resource with Policy: “Unrestricted”, “Single node”, uncheck “Use Photon Acceleration”, select least node type,

Now go to Workspace-> Users-> your email id will be displayed, add notebook from right, click on “notebook” rename as

Notebook 1: “Day 5: Part 1: DBUtils Notebook Utils: Child”

```
dbutils.notebook.help()
-------------------------
a = 10
b = 20
-------------------------

c = a + b
-------------------------
print(c)
-------------------------
# And I'm going to use the exit here. So basically what exit will do is it is going to execute all the commands before that. And it is going to come here. And if ever there is an exit command, it is going to stop executing the notebook at that particular point and it is going to return the value, whatever you are going to enter here.
dbutils.notebook.exit(f'Notebook Executed Successfully and returned {c}')

# We are going to access this notebook in another Notebook
-------------------------
print('Test')
```

Notebook 2: “Day 5: Part 2: DBUtils Notebook Utils: Parent”

```
print('hello')
-------------------------
```

```
dbutils.notebook.run('Day 5 Part 1 DBUtils Notebook Utils Child',60)
60 is timeout parameter


Click on “Notebook Job”, will lend you to “Workflow”, where it is executed as job, there are two kinds of clusters, one is interactive and another is “Job”, it’s executed as a “Job”, under “Workflow”, check all “Runs”.
```

Now "clone" Notebook 1: "Day 5: Part 1: DBUtils Notebook Utils: Child" and Notebook 2: "Day 5: Part 2: DBUtils Notebook Utils: Parent" and rename as "Day 5: Part 3: DBUtils Notebook Utils: Child Parameter" and "Day 5: Part 4: DBUtils Notebook Utils: Parent Parameter"

Notebook 3: "Day 5: Part 1: DBUtils Notebook Utils: Child Parameter"

```
dbutils.notebook.help()


---------------------------
dbutils.widgets.text(name='a',defaultValue='',label = 'Enter value of a ')
dbutils.widgets.text(name='b',defaultValue='',label = 'Enter value of b ')
---------------------------
a = int(dbutils.widgets.get('a'))
b = int(dbutils.widgets.get('b'))
```

 # The dbutils.widgets.get function in Azure Databricks is used to retrieve the current value of a widget. This allows you to dynamically incorporate the widget value into your Spark jobs or SQL queries within the notebook.

```
---------------------------
c = a + b
---------------------------
print(c)
---------------------------
dbutils.notebook.exit(f'Notebook Executed Successfully and returned {c}')
```

Notebook 4: "Day 5: Part 4: DBUtils Notebook Utils: Parent Parameter"

```
print('hello')
-------------------
dbutils.notebook.run(Day 5: Part 1: DBUtils Notebook Utils: Child Parameter',60,{'a' : '50', 'b': '40'})
# 60 is timeout parameter
```

 # go to Widget setting from right, change setting to "On Widget change"--> "Run notebook", now entire notebook is getting executed

On right hand side in "Workflow" → "Runs", there are Parameters called a and b.

Day 6: What is delta lake, Accessing Datalake storage using service principal

- ✓ **Introduction to section Delta Lake: Delta is a key feature in Azure Databricks designed for managing data lakes effectively. It brings ACID transactions to Apache Spark and big data workloads, ensuring data consistency, reliability, and enabling version control. Delta helps users maintain and track different versions of their data, providing capabilities for rollback and audit.**
- ✓ **In this section, we will dive into Delta Lake, where the reliability of structured data meets the flexibility of data lakes.**
- ➢ **We'll explore how Delta Lake revolutionizes data storage and management, ensuring ACID transactions and seamless schema evolution within a unified framework.**

- ✓ Discover how Delta Lake enhances your data lake experience with exceptional robustness and simplicity.
- ✓ We'll cover the key features of Delta Lake, accompanied by practical implementations in notebooks.
- ✓ By the end of this section, you'll have a solid understanding of Delta Lake, its features, and how to implement them effectively.

- ✓ ADLS != Database, in RDBMS there is called ACID Properties which is not available in ADLS.

Data Lake came forward to solve following drawback of ADLS:

- ✓ Drawbacks of ADLS:

1. No ACID properties
2. Job failures lead to inconsistent data
3. Simultaneous writes on same folder brings incorrect results
4. No schema enforcement
5. No support for updates
6. No support for versioning
7. Data quality issues

- ✓ What is Delta Lake?
- It is an Open-source framework that brings reliability to data lakes.
- Brings transaction capabilities to data lakes.
- Runs on top of your existing data lake and supports parquet.
- Delta Lake is not a data warehouse or a database.
- Enables Lakehouse architecture.

A. Datawarehouse can work only on structure data, which is first generation evolution. However it is supporting ACID properties. One can delete, update and perform data governance on it.

Datawarehouse cannot handle the data other than structure cannot serve a ML use cases.

B. Modern data warehouse architecture: There is Modern data warehouse architecture, which includes usage of Data Lakes for object storage, which is cheaper option for storage, this also called two tier architecture.

So the best features would be first one.

It supports the any kind of data can be structured or unstructured, and the ingestion of data is much faster. And the data lake is able to scale to any extent. And let us see what the drawbacks here are.

Like we have seen, Data Lake cannot offer the acid guarantees, it cannot offer the schema enforcement, and a data lake can be used for ML kind of use cases, but it cannot serve for BI use case, a BI use case is better served by the data warehouse.

That is the reason we are still using the data warehouse in this architecture.

C. Lakehouse Architecture: Databricks gave a paper on Lakehouse, which proposed the solution by just having a single system that manages both the things.

So Databricks has solved this by using Delta Lake. They introduced metadata, which is transaction logs on top of the data lake, which gives us data warehouse like features.

So Delta Lake is one of the implementation that uses the Lakehouse architecture. If you can see in the diagram there is something called metadata caching and indexing layer. So under the hood there will be data lake on the top of the data lake. We are implementing some transaction log feature where that is called the Delta lake, which we will use the Delta Lake to implement Lakehouse architecture.

So let's understand about the Lakehouse architecture now. So the combination of best of data warehouses and the data lakes gives the Lakehouse where the Lakehouse architecture is giving the best capabilities of both.

If you can see the diagram, Data Lake itself will be having an additional metadata layer for data management, which having a transaction logs that gives the capability of data warehouse.

So using Delta Lake we can build this architecture. So let's see more about the Lakehouse architecture now. So coming to this we have the data lake and data warehouse which are architecture we have seen. And each is having their own capabilities.

Now Data Lake House is built by best features of both. Now we can see there are some best elements of Data Lake and there are best elements of Data Warehouse. Lake House also provides traditional analytical DBMs management and performance features such as Acid transaction versioning, auditing, indexing, caching, and query optimization.

Create Databricks instances (with standard Workspace otherwise Delta Live tables and SQL warehousing will be disabled) and ADLS Gen 2 instances in Azure Portal.

Hands-on: Accessing Datalake storage using service principal:

"Day 6 Part 1 Test+access.ipynb"

Source Link: Tutorial: Connect to Azure Data Lake Storage Gen2 - Azure Databricks | Microsoft Learn

Inside ADLS Gen 2, create a ADLS Gen 2 with name "deltadbstg", create a container with name "test", inside this container add a directory with name "sample", upload a csv file name "countires1.csv".

Inside Databricks instances: Create a compute resource with Policy: "Unrestricted", "Single node", uncheck "Use Photon Acceleration", select least node type.

Go give permission, we have unity catalogue.

- **Go to Azure Entra ID (previously Azure Active directory), inside it, going to create some service principle, click on "App Registration" on left hand side where you can create an app. Click on "New Registration",**
- **Give name: "db-access", leave other settings as it is. Copy "Application (client) ID" and "Directory (tenant) ID" from "db-access" overview.**
- **Eg: Application ID: dbf11b5d-9d81-4cc8-82f3-f14da29c8c98**
- **Directory (tenant) ID: 076af1d9-53a7-48e4-9c43-8ae1c16db3e2**
- **Also copy secret key from left, "certificates & secrets" from left, click on "+ New client secret", give "Description" as "dbsecret" and click on "Add".**
- **Copy the "Value" from "dbsecret" now.**
- **Eg: Client Secret Value or service credential:"HnF8Q~braIA_bVu6IVtDXrvVYzm_YCncVouGtcb."**
- **Note three keys "Application (client) ID" and "Directory (tenant) ID" and "Value" from "dbsecret" i.e. secret ID in a text notebook.**
- **Inside notebook, secret ID is "service credential",.**
- **To give access to data storage, goto ADLS Gen 2 instances in Azure Portal, go to "Access Control (IAM)", click on "+Add", click on "+Add Role Assignment", search for "Storage Blob Data Contributor", click on storage blob contributor" and "+select members", type service principle which is "db-access". Select, finally Review and Assign.**

Replace everything here:

```
service_credential = dbutils.secrets.get(scope="<scope>",key="<service-credential-key>")

spark.conf.set("fs.azure.account.auth.type.<storage-account>.dfs.core.windows.net", "OAuth")

spark.conf.set("fs.azure.account.oauth.provider.type.<storage-account>.dfs.core.windows.net",
"org.apache.hadoop.fs.azurebfs.oauth2.ClientCredsTokenProvider")

spark.conf.set("fs.azure.account.oauth2.client.id.<storage-account>.dfs.core.windows.net",
"<application-id>")
```

spark.conf.set("fs.azure.account.oauth2.client.secret.<storage-account>.dfs.core.windows.net", service_credential)

spark.conf.set("fs.azure.account.oauth2.client.endpoint.<storage-account>.dfs.core.windows.net", "https://login.microsoftonline.com/<directory-id>/oauth2/token")

Hands on 2: Drawbacks of ADLS – practical:

Day 6 Part 2 .+Drawbacks+of+ADLS.ipynb

- **Create new directory in "test" container with name "files" and upload csv file "SchemaManagementDelta.csv"**

This hands on showing that using data lake we are unable to perform Update operation. Only in delta lake this operation is supportive.

Even using spark.sql, are unable to perform Update operation. This is one of Drawbacks of ADLS.

Versioning is also not available in ADLS, which is Drawbacks of ADLS.

Hands on 3: Creating Delta lake:

Day 6 Part 3 +Drawbacks+of+ADLS+-+delta.ipynb

Hands on 4: Understanding Transaction Log:

Day 6 Part 4 Understanding+the+transaction+log.ipynb

> **Transaction logs tracks changes to delta table and it is responsible for bringing the ACID compliance for Delta lake.**

Day 7: Creating delta tables using SQL Command

Day 07 Part 1 pynb

- **Change default language to "SQL", then Create schema with name "delta", before further code, where exactly we can see this table, go to "Catlog", there are two defaults catalogues, "Hive metastore" and "samples". This is not "Unity catalog".**
- **The Hive metastore is a workspace-level object. Permissions defined within the hive_metastore catalog always refer to the local users and groups in the workspace. Hence Unity catalog cannot manage the local hive_metastore objects like other objects. For more refer https://docs.databricks.com/en/data-governance/unity-catalog/hive-metastore.html#access-control-in-unity-catalog-and-the-hive-metastore.**

- Schema with name “delta” is created in “Hive metastore” catalogue, this is schema not database.
- Create table with name “’delta’.deltaFile” , any table which you are creating be default is a delta table in Databricks. Check again Schema with name “delta” which is created in “Hive metastore” catalogue, here one symbol with delta is also showing.
- To find exact location of this delta table: Go to “Cataloge”-> “Hive metastore” -> “delta” -> “deltaFile”-> “Details”-> “Location”.
- There is no parquet file, means we haven’t inserted any data.

Day 07 Part 2 pynb file

Day 07 Part 3 pynb file

Day 07 Part 4 pynb file

- ✓ Create “SchemaEvol” in “Test” containers (this is part of Day 6), before running this file upload two csv files, “SchemaLessCols.csv” and “SchemaMoreCols.csv” in “SchemaEvol” directory in “Test”.
- ✓ Schema Enforcement or Schema Validation: Let’s take a delta table, which is maintained strictly, we are ingesting data into this table on daily basis. In one ingestion, if a new data is coming with new “Column” which is not available in this schema.
- ✓ Delta Lake uses Schema validation on “Writes”.
- ✓ Now on a fine day during the data ingestion some data comes with a new column which is not in the schema of our current table, which is being overwritten to the location where our delta lake is present.
- ✓ Now, generally, if you are using the data lake and I'm mentioning it again to prevent any confusion, I mean the data lake, not the delta lake, the general data lake will allow the overwriting of this data and we will lose any of our original schema.
- ✓ Like we have seen in the drawback, we try to overwrite to the location where we lost our data and it allowed the write.

- ✓ But coming to the Delta Lake, we have a feature called schema enforcement or Schema validation, which will check for the schema for whatever the data that is getting written on the Delta Lake table.
- ✓ If the schema does not match with the data which we are trying to write to the destination, it is going to reject that particular data.
- ✓ It will cancel the entire write operation and generates an error stating that the schema is not matching the module of the schema.
- ✓ Validation is to safeguard the delta lake that ensures the data quality by rejecting the writes to a table that do not match the table schema.
- ✓ A classic example is you will be asked to scan your IDs before entering your company premises, so that is going to check if you are the authorized person to enter this.
- ✓ Similarly, schema enforcement acts as a gatekeeper who checks for the right data to enter to the Delta Lake.
- ✓ Now, how does this schema enforcement works exactly?

- ✓ So to understand this, Delta Lake uses the schema validation on writes, which means all the new writes to the new table are checked for the compatibility with the target table.
- ✓ So during the right time it is going to check for the schema compatible or not.
- ✓ If the schema is not compatible, data is going to cancel. The delta lake cancels the transaction altogether.

- ✓ No data is being written, and it raises an exception to let the user know about the mismatch. And there are certain rules on how the schema enforcement works.
- ✓ And let us see on what conditions the incoming data will be restricted in writing to the delta table. So let's see about the rules now.
- ✓ So it cannot contain any additional columns like we have seen before.
- ✓ If the incoming data is having a column more than the one defined in the schema, it is treated as a violation to the schema enforcement.
- ✓ But if it is having less number of columns than the target table, it is going to allow the write by giving the null value to the existing columns where there is no data for this particular table.
- ✓ But if the incoming data is having more number of columns, it is going to cancel that insert. Now there is one more rule where it cannot have the different data types. If a delta table's column contains the string data, but the corresponding column in the data frame incoming is having the integer data, the schema enforcement will raise an exception and it will prevent the write operation entirely.
    - Now how is this schema enforcement useful?
    - Because it is such a stringent check, schema enforcement is an excellent tool to use as a gatekeeper to get a clean, fully transformed data set that is ready to use for production or consumption.
    - It is typically enforced on tables that directly fed into the machine learning algorithm by dashboard or data analytics or visualization tools, and schema enforcement is used for any production system that is requiring highly structured, strongly typed semantics checks.
    - And it's enough with this theory.
    -

- Trying to append more columns using code. Extra column is “Max_Salary_USD”.
- Source with fewer columns will accept.
-

Day 07 Part 5 pynb file

- ➢ Schema Evolution: Schema evolution in Databricks Delta Lake enables the flexible evolution of table schemas, allowing changes such as adding, removing, or modifying columns without the need for rewriting the entire table. This flexibility is beneficial for managing changes in data structures over time.
- ➢ So schema evolution is a feature that allows the user to easily change the tables current schema to accommodate the data changing over time.
- ➢ Most commonly, it is used when performing an append or an override operation to automatically adapt the schema to include one or more columns.
- ➢ “mergeSchema”,”True” enables this Schema Evolution in Delta tables.

Day 07 Part 6 pynb file

Audit Data changes & Time Travel: "Time travel" in Delta Lake enables users to query a historical snapshot of the data at a specific version, facilitating data correction or analysis at different points in time.

Day 07 Part 7 pynb file

- **Vacuum Command:**

- **If you are getting a very high storage cost now, if your organization wish to delete the old data like a 30 days old data, you can make use of the vacuum command.**
- **Now let's see how we can implement this. Now in order to know how many files will be deleted, you can make use of the dry run feature in the vacuum.**
- **So let's see how you can implement this. Now I am going to use some feature called dry Run.**
- **So it is not actually going to delete any kind of data. It is just going to show us how many files will be deleted. So it will ideally give a list of first thousand files that will be deleted and it will not actually delete.**
- **It will just show what files will be deleted. Now, by default, the retention period of this vacuum command is seven days. So any data that is having the age of more than seven days that will be deleted by default using this vacuum command.**
- **So we just created our table and we just inserted few records, but we haven't have any data which is older than seven days.**
- **Now, if I just try to run this particular command, it is going to show me nothing. And now you can see he's not returning any results because we are not having any data, which is post seven days old, which is the retention period of this particular vacuum command.**

- **Now for testing purpose, if you want to delete the data, which is less than seven period of time than the default duration, you can make use of the retain command.**
- **There is restriction, just make "retentionDuration" to True.**

Day 07 Part 8 pynb file

Convert to Delta:

Day 8: Understanding Optimize Command – Demo

- ✓ **Optimize command in Databricks primarily reduces the number of small files and compacts data, improving query performance and storage efficiency.**
- ✓ **So on a high level, the optimize command will help us to compact multiple small files into a single file.**
- ✓ **So this is one of the optimization feature in the delta lake which the name itself indicates the optimize. Now the use of this particular command is to compact multiple files into a single file.**
- ✓ **If you are aware of the small file problem in the spark, where if you have 100 small files, where each transaction is creating a file, if you want to read the content of each and every file, the time to open each and every file is more than reading the actual file.**
- ✓ **It need to open the file, it need to read the content, and it need to close the file. So the time to open and close each and every file is more than actual content reading. So this is why the optimize command helps in a way where it can just combine all the active files into a single file.**
- ✓ **I am mentioning something like active files, because sometimes there are inactive files where. Let us try to understand what exactly is this active and what is an inactive.**
- ✓ **So let's see by taking an example. So we'll be doing some transformations on our data in our Delta lake transformations are nothing but the operations like inserts, deletes and updates and etc..**
- ✓ **So each action or a transformation will be treated as a commit and it will create the parquet file.**
- ✓ **Along with that it will create the delta log files. So imagine we are creating a empty table because we are doing an empty table creation. It is also an operation where the operation is recorded as a create table.**
- ✓ **And it is not going to create any parquet file, but it is going to create a delta log.**

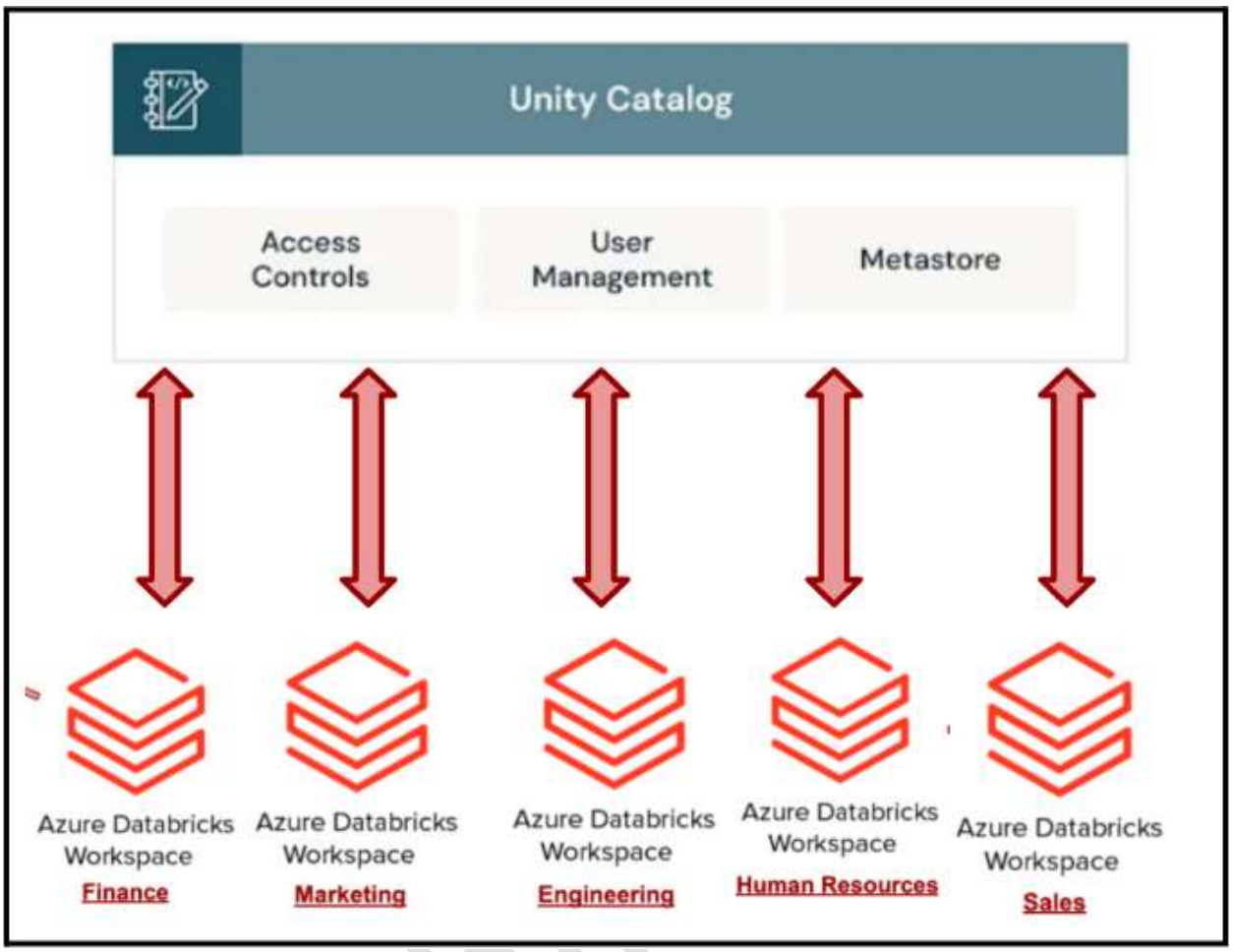
Details: to be added

Day 9: What is Unity Catalog: Managed and External Tables in Unity Catalog

Understanding the Problem Unity Catalog Solves

Unity Catalog: bringing order to the chaos of the cloud. It is a data governance tool that provides a centralized way to manage data and AI assets across platforms.

Unity Catalog: a powerful tool designed to centralize and streamline data management within Databricks.



Unity Catalog centralizes all metadata, ensuring that user data definitions are standardized across the organization. This means that the marketing, customer service, and product development teams all have access to a single, consistent source of truth for user data. By providing a unified view of all data assets, **Unity Catalog** makes it easier for teams to access the information they need without having to navigate through multiple systems.

The marketing team can easily access support interaction data, and the product development team can view user engagement metrics, all from a single platform. This unified approach reduces administrative overhead, enhances data security, and ensures that data is accurate and compliant, supporting better data-driven decision-making and driving business success.

1. **Typical Databricks Setup**
    - Organizations use Databricks workspaces for projects (e.g., Azure Databricks).
    - User management is required for assigning roles (developers, leads, managers).
    - Each workspace has a Hive metastore storing managed tables and defined access controls.
2. **Challenges in Managing Multiple Workspaces**
    - Multiple environments (dev, UAT, prod) and business units/projects require several workspaces.

- Duplication of user management, access controls, and cluster creation policies across workspaces.
- Lack of centralized governance for managing users, auditing, and metadata.
- Difficult to track who has access to specific data across all workspaces.

3. **Need for a Centralized Solution**
   - A central system for governance, audit, and metadata management is required.

How Unity Catalog Solves the Problem

1. **Centralized Governance**
   - Manages user access, metadata, and governance for multiple workspaces centrally.
   - Provides visibility and control over access permissions across all workspaces.
2. **Unified Features**
   - **Access Controls**: Define and enforce who can access what data.
   - **Lineage**: Track how data tables were created and used.
   - **Discovery**: Search for objects like tables, notebooks, and ML models.
   - **Monitoring**: Observe and audit object-level activities.
   - **Delta Sharing**: Share data securely with other systems or users.
   - **Metadata Management**: Centralized management of tables, models, dashboards, and more.

Summary

Unity Catalog is a centralized governance layer in Databricks that simplifies user and metadata management across multiple workspaces. It enables unified access control, data lineage, discovery, monitoring, auditing, and sharing, ensuring seamless management and governance in one place.

Hands-on:

Step 1: In Azure Portal, create a Databricks workspace and ADLS Gen2, add these two Databricks workspace and ADLS Gen2 in "Favorite" section.

Step 2: search for "Access connectors for Azure Databricks", create "New", only give resource group name and Instance name "access-connectors-sachin" here, you need not to change anything here.

Click on “Go to Resource”. Now in “Overview”, “Resource ID”, can use this “Resource ID” while creating the Metastore.

Step 3: Create ADLS Gen2, “deltadbstg”-> “test”-> “files”->”SchemaManagementDelta.csv”. Now give access of this Access connectors to ADLS Gen2, go to ADLS Gen2, go to “Access Control IAM” from left pane, click on “Add”-> “Add Role Assignment”-> search for “Storage Blob Data Contributor”, in “Members”, select “Assign Access to”->”Managed Identity” radio button, “+Select Members”-> select “Access connectors for Azure Databricks” under “Managed identity” drop down menu-> “Select”-> “access-connectors-sachin” -> “Review+Assign”.

- Now using this managed identity, our Unity catalog or Metastore can access this particular storage account. And the reason why we are doing is we need to have a container where that is going to be accessed by the unity catalog to store its managed tables, and we will see that in upcoming lectures.

Step 4: To use Unity Catalog: following are pre-requisites:

1. Must have Databricks Premium Workspace
2. Configure Metastore (Done in Step 3)
3. Attach premium workspace to Metastore
4. Note: https://accounts.azuredatabricks.net/ , in case you are not able to access "Manage Account" setting in my Azure Portal, click here:
5. a. https://learn.microsoft.com/en-us/answers/questions/1843839/not-able-to-access-manage-account-setting-in-my-az
6. b. https://learn.microsoft.com/en-us/answers/questions/1344479/unable-to-login-accounts-azuredatabricks-net-and-c
7. c. https://learn.microsoft.com/en-us/azure/databricks/admin/#account-admins

   Now go to Databricks, we need to start a creating a meta store, meta store is top level container in the unity catalog, go “Manage Account” under “Sachindatabricks name” from right top -> “Catalog” from left pane, “create meta store”, provide “Name” as “metastore-sachin”,”Region”(can create one meta store in single region), “ADLS Gen2 path” (go to ADLS Gen2-> create container-> “Add Directory”, paste <container_name>@<storage_account_name>.dfs.core.windows.net/<directory_Name> Or test@deltadbstg123456.dfs.core.windows.net/files), “Access connector ID” (go to “Access connectors”-> “access-connectors-sachin” -> copy that “resource ID”)-> “Create”.
   Attach with any workspaces. “Enable Unity Catalog?”-> “Enable”.

Step 5: Create the required users to simulate the real time environment: go to "Microsoft Entra ID"-> "Add"-> "users" from left pane-> "new user"-> "create new user"-> give any name to "User Principal Name"-> give any display Name "Sachin_Admin" ->give custom password not "Auto generate password"-> not required to change any "Properties", "Assignments"-> click on "Create".

- **Now login from https://accounts.azuredatabricks.net/ using username and password from Step 5. Using unity catalog, we can access to this user. My username is : sa******xgmail.onmicrosoft.com and password is K***r3**#**

Step 6: Create one more user as in Step 5, now we two new users.

- **So these are two new users we have created for this session, where we will try to simulate the real time environment by giving them required access to understand the roles and responsibilities clearly, because user management is something, if you are in a project, generally there will be an admin who can do this, but in real times they will expect you to handle this by your own. A data engineer must also be aware who can access what.**
- **First user will be Workspace admin and second will be developer.**

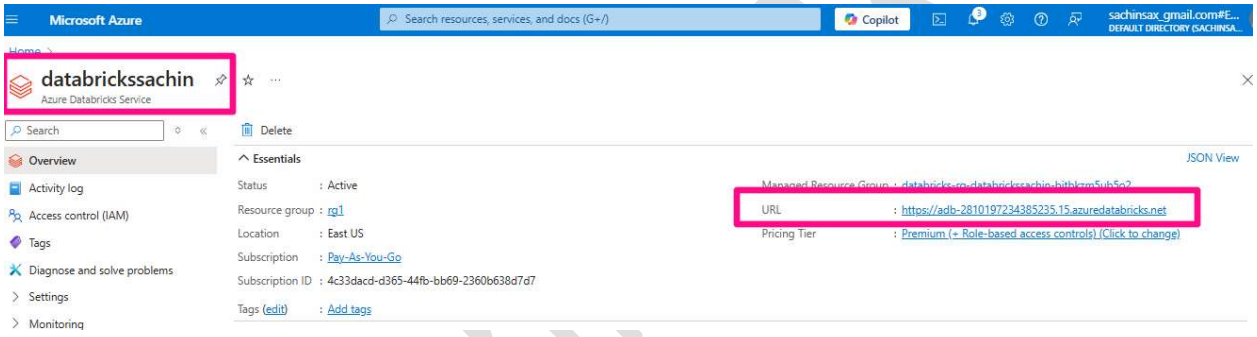
Step 7: Now in Databricks portal, click on "Manage Account", from right top, this Databricks portal is created neither by Workspace admin nor developer, in order to add user, click on "User Management" from left pane, we need to add both Workspace admin and developer.

- **Click on "Add User"-> paste email id from "Microsoft Entra ID" -> "User Principal name", can give any "first name" and "last name" as "Workspace admin". Now add developer "User Principal name" in same way.**
- **Click on "Setting" from left, -> "User Provisioning"-> "Set up user provisioning".**
- **Open a "incognito window" mode to open** https://portal.azure.com/#home with "admin Sachin" and "Developer Sachin" both, it will ask to create new password.

Step 8: to create group, click on "Manage Account", from right top, this Databricks portal is created neither by Workspace admin nor developer, in order to add user, click on "User Management" from left pane-> "Groups" -> "Add Group", we are going to create two groups, first group is "Workspace Admins"-> "Add Members" from admin only and second group is "Developer team"-> "Add Members" of developer only.

Step 9: It's time to give permission, in Databricks portal, click on "Manage Account", from right top, this Databricks portal is created neither by Workspace admin nor developer, in order to give permission, click on "Workspaces", click on respective "Workspace" -> inside it "permissions"-> "Add Permissions"-> we need to add groups which we created in Step 8, to admin group assign "Permission" as "Admin" and to developer group assign "Permission" as "User".

Step 10: Now login from https://portal.azure.com/signin/index/ using username and password, need to paste databricks workspace go to Databricks portal, click on "Manage Account", from right top, this Databricks portal is created neither by Workspace admin nor developer, go to Azure portal from main where we created first Databricks workspace, copy "Workspace URL" ending with xxx.azuredatabricks.net.



Sign in with admin credentials, in similar way, go to Azure portal from main where we created first Databricks workspace, copy "Workspace URL" ending with xxx.azuredatabricks.net and sign in with user credentials.

- **Just check that in developer portal, we do not have "Manage Account" setting, also in developer portal cannot see any compute resources in "Compute" tab.**

Step 11: Create Cluster Policies: login with admin and move to databricks portal from this "sachin admin" login, go to "admin setting" from right top.

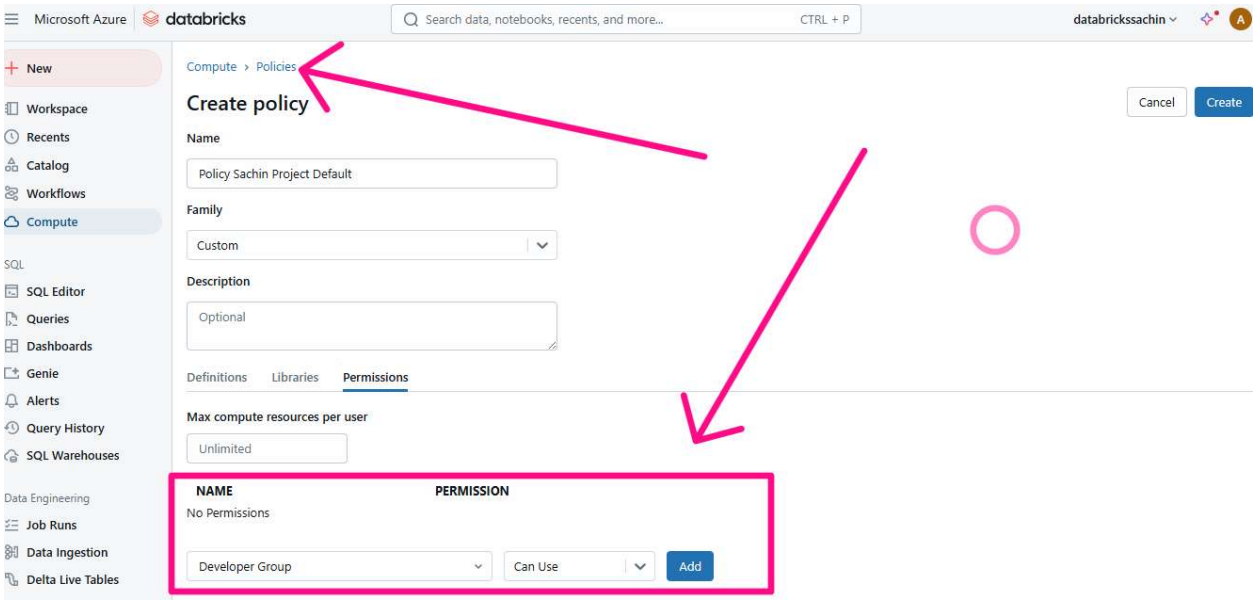
- **Click on "Identity and access" from second left pane. Click on "Manage" from "Management and Permissions" in "Users". Click on "Kumar Developer" right three dots, click on "Entitlements", check on "Unrestricted cluster creation", "Confirm" it.**
- **Now check "Compute" tab of "Kumar Developer" in databricks portal that, this "create compute " resource is now enabled.**

- **We do not want to give all kind of "Compute" resources to "Kumar Developer", so we can restrict by using create policies, otherwise it is going to shortwave bill.**
- **(This step is for disable Compute Resource in developer portal)To create policies, click on "Kumar Developer" right three dots, click on "Entitlements", check off "Unrestricted cluster creation", "Confirm" it. Now check "Compute" tab of "Kumar Developer" in databricks portal that, this "create compute" resource is now disabled again.**
- **Compute policy reference: https://learn.microsoft.com/en-us/azure/databricks/admin/clusters/policy-definition**
- **Jump to ""Sachin Admin" databricks portal, click on "compute", click on "Policies": click on "create policy"-> give "Name" as "Sachin Project Default Policy" , select "Family" as "custom"->**

Change the following code:

```
{
 "node_type_id": {
  "type": "allowlist",
  "values": [
   "Standard_DS3_v2"
  ]
 },
 "spark_version": {
  "type": "fixed",
  "value": "13.3.x-scala2.12"
 },
 "runtime_engine": {
  "type": "fixed",
  "value": "STANDARD",
  "hidden": true
 },
 "num_workers": {
  "type": "fixed",
  "value": 0,
  "hidden": true
 },
 "data_security_mode": {
  "type": "fixed",
  "value": "SINGLE_USER"
 },
 "cluster_type": {
  "type": "fixed",
  "value": "all-purpose"
 },
 "instance_pool_id": {
  "type": "forbidden",
  "hidden": true
 },
 "azure_attributes.availability": {
  "type": "fixed",
  "value": "ON_DEMAND_AZURE",
  "hidden": true
 },
 "spark_conf.spark.databricks.cluster.profile": {
  "type": "fixed",
  "value": "singleNode",
  "hidden": true
 },
 "autotermination_minutes": {
  "type": "fixed",
  "value": 20
 }
}
```

- **Click on “Permission” in ”Sachin Admin” databricks portal, click on “compute”, click on “Policies”: click on “create policy”-> give “Name” as “Sachin Project Default Policy”, in “Name”, select “Developers” -> “can use”, click on “Create” from right top.**



- **Now, if you jump to as “Sachin Developer” databricks portal, “Compute” tab, one “Sachin Project Default Policy” will appear at right top.**

Step 12: **Cluster Pools in Databricks:**

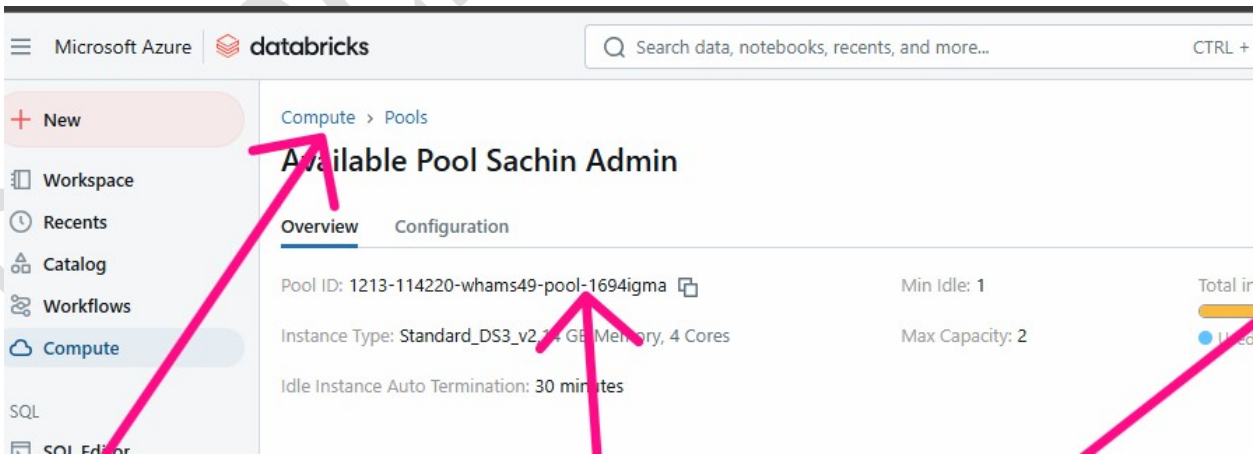
- **Purpose of Cluster Pools**:
  - Reduce cluster creation time for workflows or notebooks.
  - Enable ready-to-use compute resources in mission-critical scenarios.
- **Cluster Creation Process**:
  - Compute resources are required to run workflows or notebooks.
  - Creating a cluster involves requesting virtual machines (VMs) from a cloud service provider.
  - VM initialization takes ~3-5 minutes, which may not be ideal for real-time tasks.
- **Cluster Pool Functionality**:
  - Cluster pools pre-request VMs from the cloud provider based on configurations.
  - Keeps some VMs ready in a running or idle state.
  - Enables faster cluster creation by utilizing these pre-initialized VMs.
- **Performance and Cost Trade-Off**:
  - Performance improves as clusters are ready in reduced time (~half the usual time).
  - Databricks does not charge for idle instances not in use, but cloud provider infrastructure costs still apply.
- **Use Case**:
  - Ideal for workflows or notebooks needing rapid cluster creation.
  - Balances between cost and efficiency by keeping resources ready.
- **Key Takeaway**:

- Cluster pools enhance performance by maintaining idle VMs for quick allocation, albeit with associated cloud costs.

- **Cluster Pools in Databricks Hands-on: Jump to "Sachin Admin" databricks portal, click on compute, go to "Pool", click on "create pool", name as "Available Pool Sachin Admin", pool will already keep you instances in ready and running state so that we can use them while creating the cluster. And these will access the resources which are readily available.**
- **Also keep "Min Idle" as 1 and "Max Capacity" as 2. Now let me make the minimum idle instance to one and maximum two. This means all the time this one instance will be in ready and in running state. And in case if this one instance is used by any cluster, another will be in the Idle state because minimum one will be idle all the time, irrespective of the one is attached or not. So in maximum of two will be created. So one can be used by cluster and if that is already been occupied, another one will be in the idle state.**
- **Change "terminate instances above minimum tier" to 30 minutes of idle time.**
- **Change "Instance Type" to "Standard_DS3_v2"**
- **Change "On-demand/spot" to "All On-demand" radio button, bcs sometimes Spot instances are not available.**
- **Create it. It will take much time. Copy Pool ID from here.**
- **Now go to Edit Policies under "Policy tab" which was done in Step 11, make changes in:**

```
},
"instance_pool_id": {
  "type": "forbidden",
  "hidden": true
},
```

To Get Pool ID here:




```
"instance_pool_id": {
   "type": "fixed",
   "value": "1211-104550-gybed90-pool-ns7wqs2q<Your Pool ID>"
```

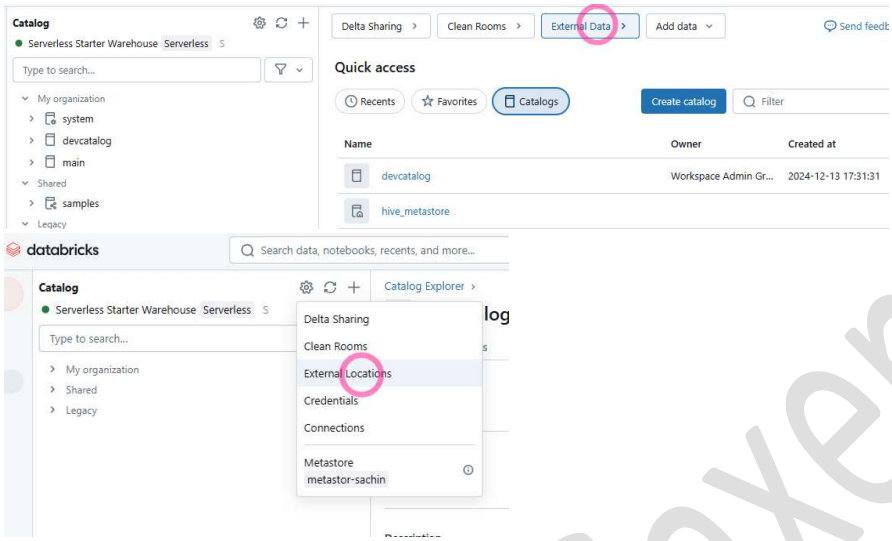
- **Now go to Compute Tab in "Sachin Admin" databricks, click on "Pool"->select given "availavle Pool Sachin Admin" -> click on Permission-> select "Developers group" (not individual developer) to "Can Attach to"-> "+Add", "Save" it.**

Step 13: Creating a Dev Catalogs: go to "Sachin Admin" databricks portal, go to "Catalog" tab, but "Create Catalog" is disabled now because we haven't define this permission, in order to give permission, go to "Main Datbricks" portal (neither Sachin Workspace admin nor Kumar developer), go to databricks portal, go to "Catalog" tab, "Catalog Explror" -> click on "Create Catalog" from right, name "Catalog name" as "DevCatalog", type as "Standard", skip "Storage location". Click on "Create".

Step 14: Unity Catalog Privileges:

- **Go to "Sachin Admin" databricks portal, but still can't see "Dev Catalog" because Sachin Admin and Kumar developer both are not having the required privileges or permission to use.**
- **Go to "Main Databricks" portal who is account admin (neither Sachin Workspace admin nor Kumar developer) go to "Catalog", Click on "Dev Catalog", then "Permissions", then "Grant", this screen is Unity catalog UI to grant privileges to "Sachin Admin", then click on "Grant", select group name "WorkSpace admins" checkbox on "create table", "USE SCHEMA", "Use Catalog" and "Select" in "Privileges presets", do not check anything here. Click on "Grant". Now, go to "Sachin Admin" databricks portal, "Dev Catalog" is showing here.**
- **To transfer ownership of "Dev Catalog", go to "Main Databricks" portal who is account admin (neither Sachin Workspace admin nor Kumar developer), go to "Catalog", Click on "Dev Catalog", click on pencil icon from mid top near Owner: sacinsax@gmail.com, "Set Owner for Dev Catalog", change to "Workspace admins" not to specific user, bcs if one user leave the organization then it creates havoc situations.**
- **Now, go to "Sachin Admin" databricks portal, "Dev Catalog" is showing here.**
- **Now, go to "Sachin Admin" databricks portal, create a "notebook" here, to run any cell in this notebook, we need "Compute", select "Create with Personal compute", "Project Defaults".**
- **Go to "Sachin Admin Databricks" portal go to "Catalog", Click on "Dev Catalog", then "Permissions", then "Grant", this screen is Unity catalog UI to grant privileges to "Sachin Admin", then click on "Grant", select group name "WorkSpace admins" checkbox on "Use Catalog", "USE SCHEMA", "Create Table" and "Select" in "Privileges presets", do not check anything here.**
- **Now Run SQL command, file is saved in "Day 9" folder with name "Unity Catalog Privileges.sql". in code GRANT USE_CATALOG ON CATALOG `devcatalog` TO `Developer Group`**

Step 15: Creating and accessing External location and storage credentials:

- **Step A: Go to “Sachin Admin Databricks” portal go to “Catalog”, we do not find any external data here, to find “External Data”, go to “Main Databricks” portal who is account admin (neither Sachin Workspace admin nor Kumar developer) go to “Catalog”, in “Catalog Explorer”, there is “External Data” below, click on “Storage Credentials”.**
- 

- **Step B: Now in ADLS Gen2, “deltadbstg”-> “test”-> “files”->”SchemaManagementDelta.csv”.**
- **Now give role assignment “Storage blob Data Contributor” to “db-access-connector” from IAM role in Azure Portal of Main admin.**
- **Step C: Now go to Step A Databrick portal, click on “create credential” under “External Data” below, click on “Storage Credentials”, “Storage Credentials Name” as “Deltastorage”, to get “Access connector Id”, go to “db-access-connector” from Azure Portal, will find “Resource ID”, copy this and paste to “Access connector Id”, click on “create”.**
- **Step D: Go to “Main Databricks” portal who is account admin (neither Sachin Workspace admin nor Kumar developer) go to “Catalog”, in “Catalog Explorer”, there is “External Data” below, click on “External Data”, click on ”Create external location” -> “Create a new external location”click on “External location name”: “DeltaStorageLocation”, in “Storage credential”, select “Deltastorage” which we created in Step C.**
- **To find URL: abfss://test@deltadbstg.dfs.core.windows.net/files (go to ADLS Gen2 “deltadbstg”-> “EndPoints”-> “Data Lake Storage”), click on “create”.**
- **Click on “Test Connection”.**
- **Step E: create a notebook in “Main Databricks” portal who is account admin (neither Sachin Workspace admin nor Kumar developer), create a compute, create with “Unrestricted”, “Multi node”, create a Access mode “Shared” , uncheck “Use Photon Acceleration”, Min workers: 1, Max workers: 2.**
- **Run the following code in notebook:**
- **Df=(spark.read.format(‘csv’).option(‘header’,’true’).load(‘abfss://test@deltadbstg.dfs.core.windows.net/files / ’))**
- **Display(Df)**

Step 16: Managed and External Tables in Unity Catalog: Do hands on also.

Question: Which of the following is primary needed to create an external table in an Unity Catalog Enabled workspace?

Answer: You need an external location created primarily pointing out to that location , So you can get access to the path to create external table.

Question: Can managed table use Delta, CSV, JSON, avro format?

Answer: No, Managed table can use only Delta format.

Day 10: Spark Structured Streaming – basics

Reference: To be published

Details: to be added

Day 11: Autoloader – Intro, Autoloader - Schema inference: Hands-on

Reference: To be published

Details: to be added

Day 12: Project overview: Creating all schemas dynamically

Reference: To be published

Details: to be added

Day 13: Ingestion to Bronze: raw_roads data to bronze Table

Reference: To be published

Details: to be added

Day 14: Silver Layer Transformations: Transforming Silver Traffic data

Reference: To be published

Details: to be added

Day 15: Golder Layer: Getting data to Gold Layer

Reference: To be published

Details: to be added

Day 16: Orchestrating with WorkFlows: Adding run for common notebook in all notebooks

Reference: To be published

Details: to be added

Day 17: Reporting with PowerBI

Reference: To be published

Details: to be added

Day 18: Delta Live Tables: End to end DLT Pipeline

Reference: To be published

Details: to be added

Day 19: Capstone Project I

Reference: To be published

Details: to be added

Day 20: Capstone Project II

Reference: To be published

Details: to be added