Disclaimer: **These slides are copyrighted and strictly for personal use only**

- This document is reserved for people enrolled into the [Ultimate Big Data Masters Program (Cloud Focused) by Sumit Sir]

- **Please do not share this document,** it is intended for personal use and exam preparation only, thank you.

- If you've obtained these slides for free on a website that is not the course's website, please reach out to legal@trendytech.in . Thanks!

- All the Best and Happy Learning!

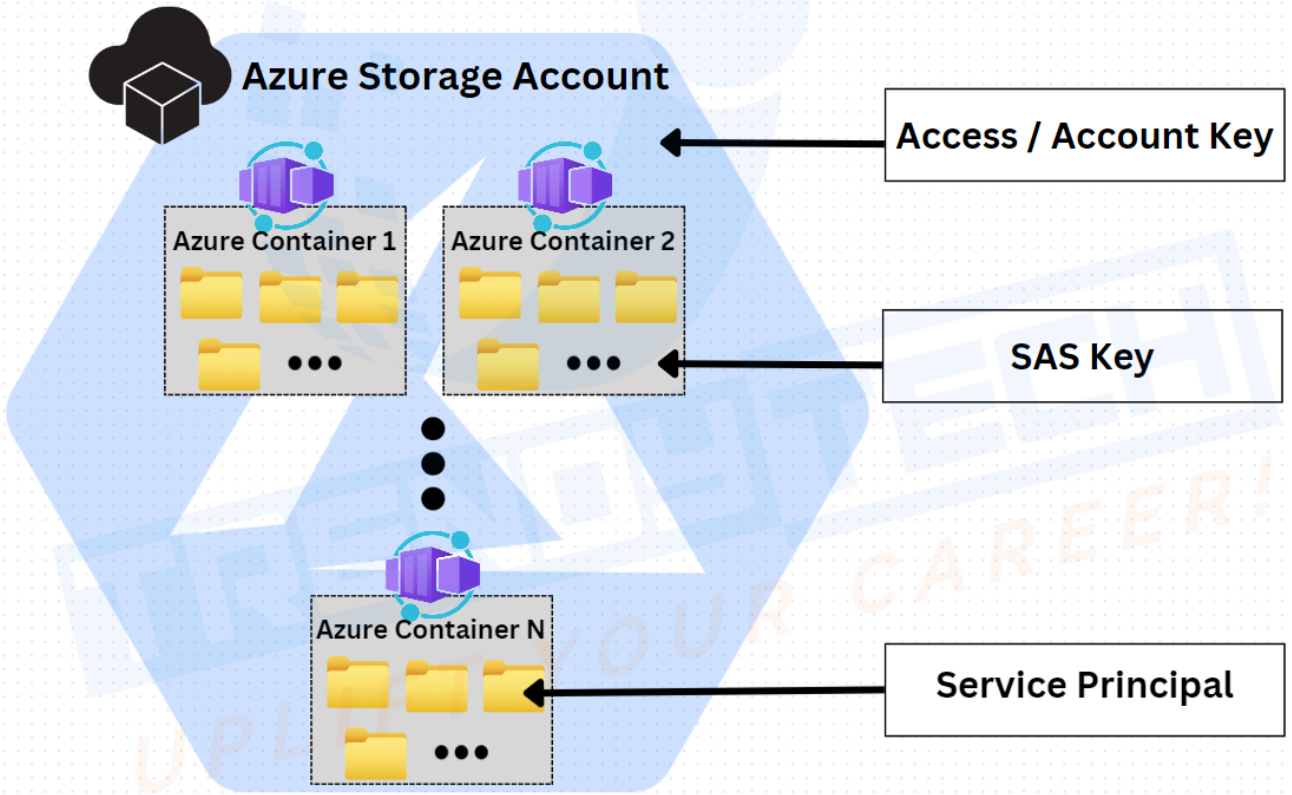
NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

©2023 All Rights Reserved Worldwide

3 ways of accessing the Storage Account

3 ways of creating mount points and accessing the files in the storage account

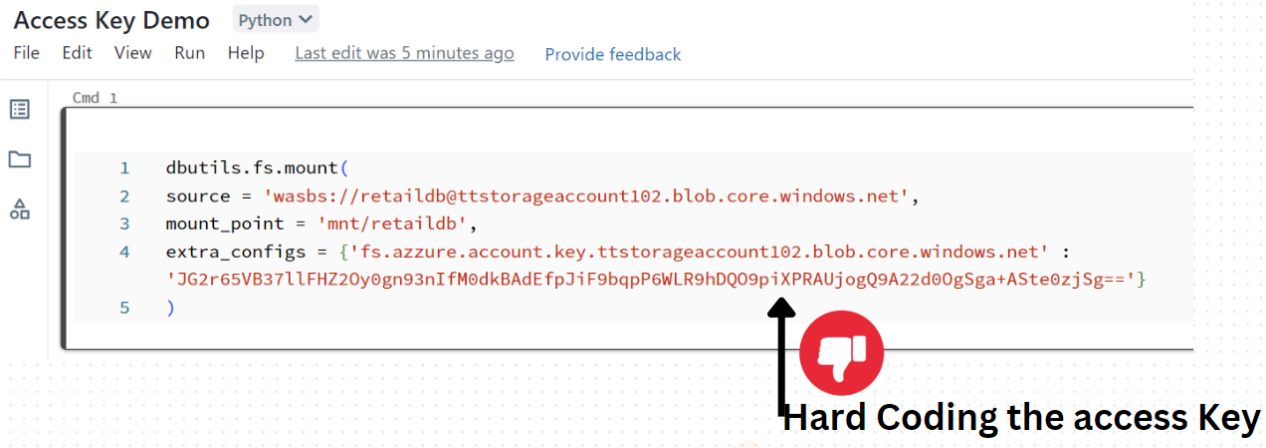
1. **Access Key / Account Key** - Access permissions are set at the storage account level. There is no possibility of restricting the access for a specific container / folder.
2. **SAS Key** - Shared Access Signature - Access permissions are set at container level. Better control on security as compared to Access Key
3. **Service Principal** - Fine grain control on the access permissions as it is set at the folder level and therefore provides the best security.



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Secret Scope



While using the Access/ Account Key approach to access the storage account, the key is hardcoded and it should be avoided.

Therefore, the Secret Scope concept came into picture. There are 2 options in Azure Secret Scope

1. **Azure Key Vault Backed Secret Scope**

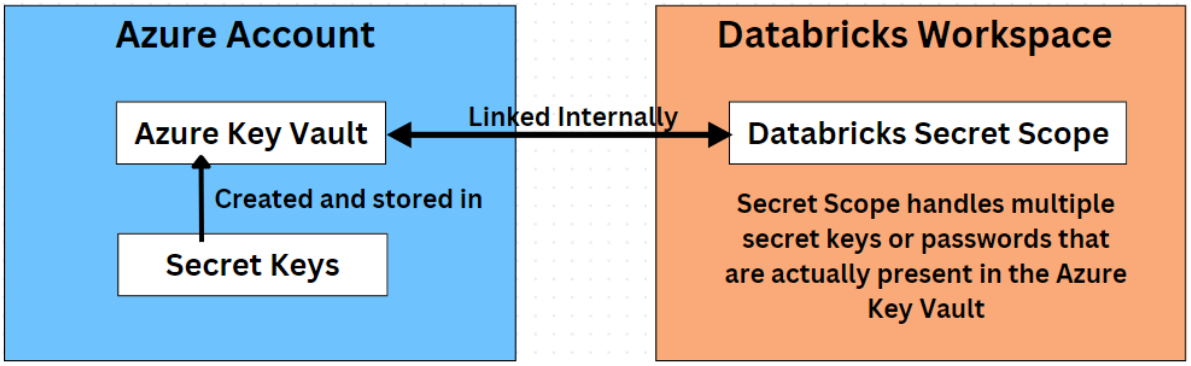
   The keys are stored in the Azure Key Vault. It is a preferred option as the other Azure services can use the keys.

Steps for creating an Azure Key Vault and linking that to the Databricks Secret Scope :

- Create an Azure Key Vault resource
- Create a secret key and store it in the Key Vault
- Launch the Databricks Workspace and append /#secrets/createScope at the end of the url to create a secret scope.
- Fill in a scope name and link it to the Azure Key Vault by providing the Vault URL and Resource ID (details will be present under properties section in the azure key vault)

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide



2. **Databricks Backed Secret Scope**

The keys are stored in an encrypted Databricks database. The Databricks backed secret scope can be created only through a CLI or an API but not through a UI.

Steps for creating a Databricks Backed Secret Scope :

- Execute the following command on your terminal to create a secret scope

  databricks secrets create-scope --scope <scope-name> --initial-manage-principal users

- Command to list all the secret scopes

  databricks secrets list-scopes

- Adding values to the secret scope

  databricks secrets put --scope <scope-name> --key <key-name>

  On executing the above command an editor will be launched for entering the secret key value.

- Command to list the keys in the scope

  databricks secrets list --scope <scope-name>

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

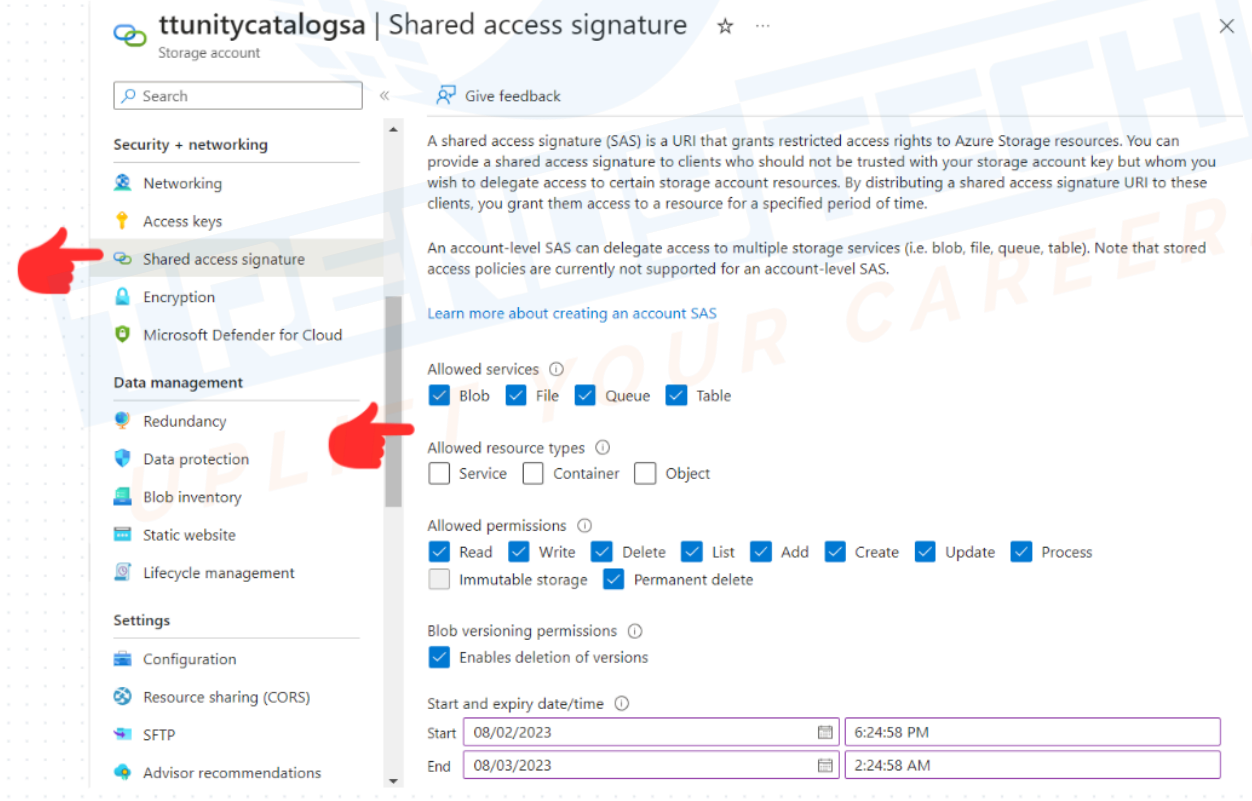
legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

- Creating a mount point to access the storage account using the Databricks backed secret scope

```
dbutils.fs.mount(

source = 'wasbs://retaildb@<storage-account-name>.blob.core.windows.net',

mount_point = 'mnt/retaildb',

extra_configs = {'fs.azzure.account.key.<storage-account-name>.blob.core.windows.net' : dbutils.secrets.get(‘<scope-name>’, ‘<key-name>’)}

)
```

Note: Make sure to replace the respective values in place of variables mentioned in angle brackets

SAS - Shared Access Signature



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

SAS token has to be generated and it would be required to connect to the storage account to access the required files. Now, this token can be added in the spark configurations using spark.conf.set to connect to the storage account.

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

spark.conf.set("fs.azure.sas.token.provider.type.<storage-account-name>.dfs.core.windows.net", "org.apache.hadoop.fs.azurebfs.sas.FixedSASTokenProvider")

spark.conf.set("fs.azure.sas.fixed.token.<storage-account-name>.dfs.core.windows.net", "<SAS-Token>")

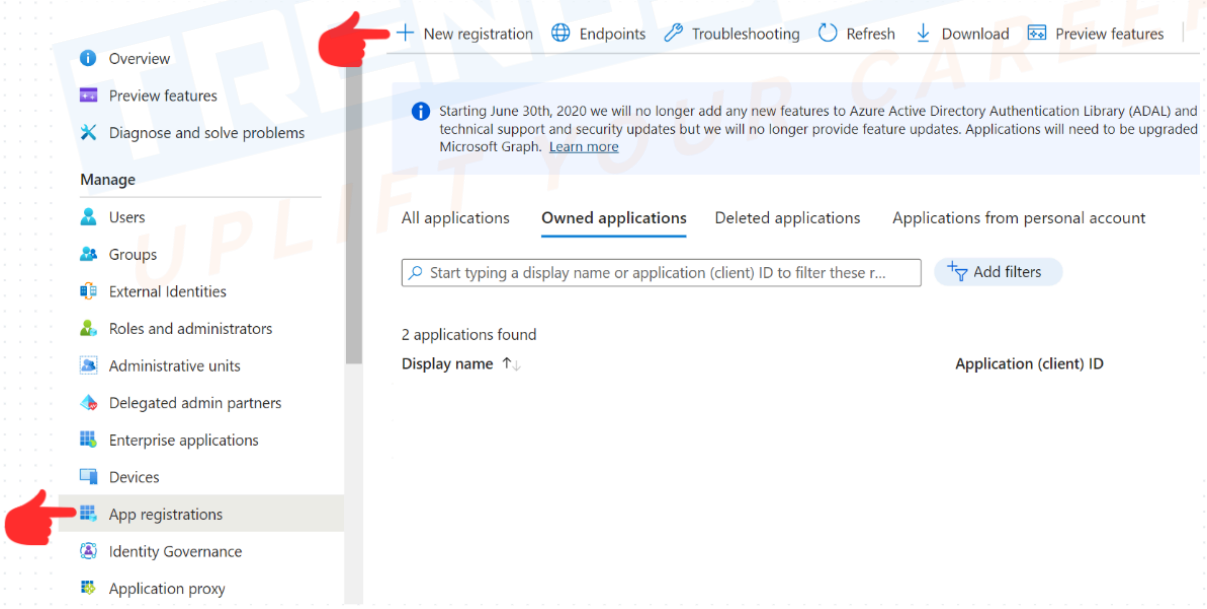
Note: Make sure to replace the respective values in place of variables mentioned in angle brackets

The SAS token can be added as a secret key to the azure key vault to access in a more secure manner.

Service Principal

To create a service principal, go through the following steps

- Navigate to the Azure Active Directory
- Create a new registration to register an application under App Registrations

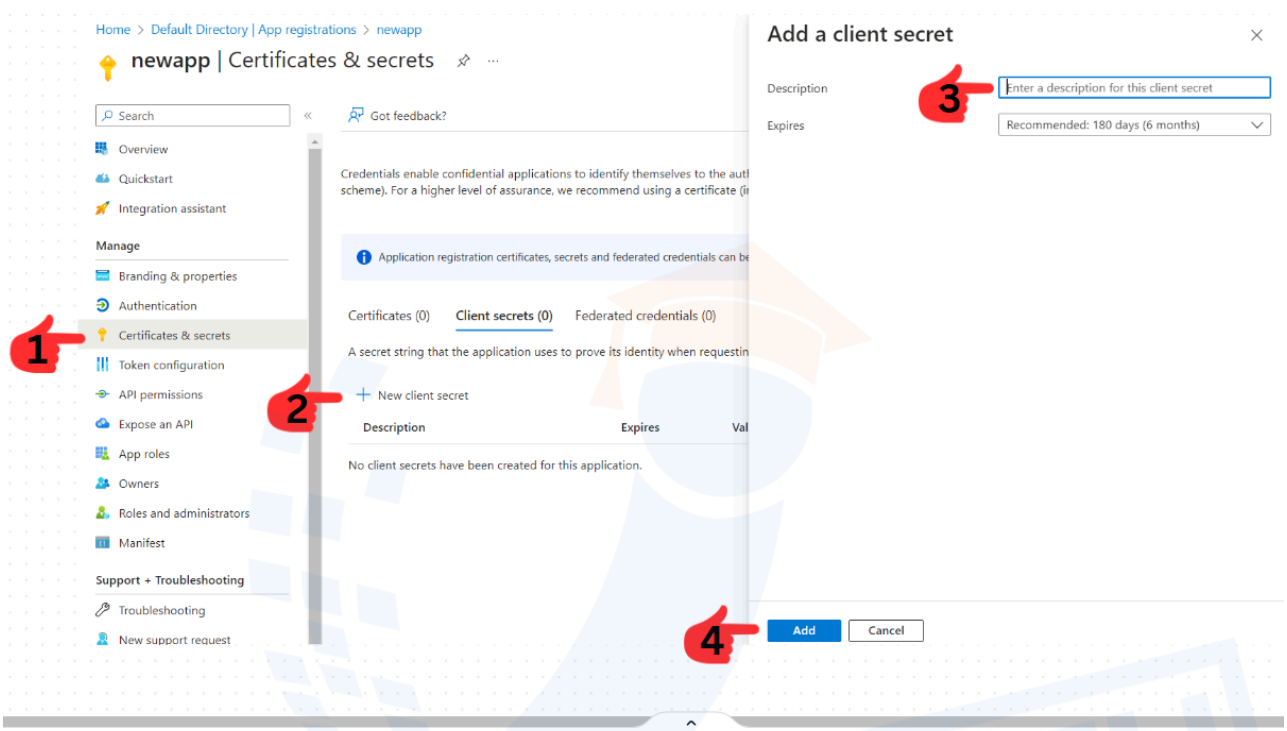


NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

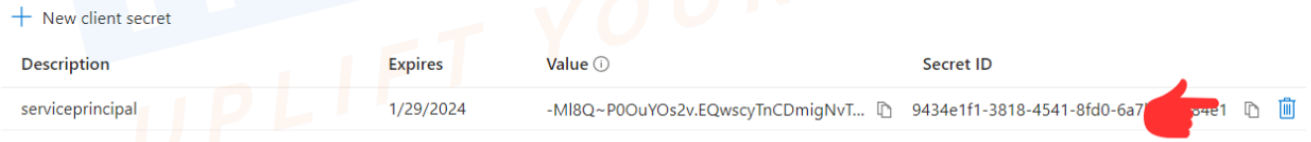
legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

- Creating a Secret or Password for the application

  Click on the New Client Secret under the “Certificates and Secrets section” to create a secret.



After the client secret is created, the password can be copied and used to establish a connection with the storage account



- The access control permissions can be provided even at directory/ folder level using the service principal approach.
- The spark configurations with spark.conf.set to connect to the storage account.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

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

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

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

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

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

- **Note: Make sure to replace the respective values in place of variables mentioned in angle brackets**
- **The service principal secret key can be added to the azure key vault to access in a more secure manner.**

Data Lake

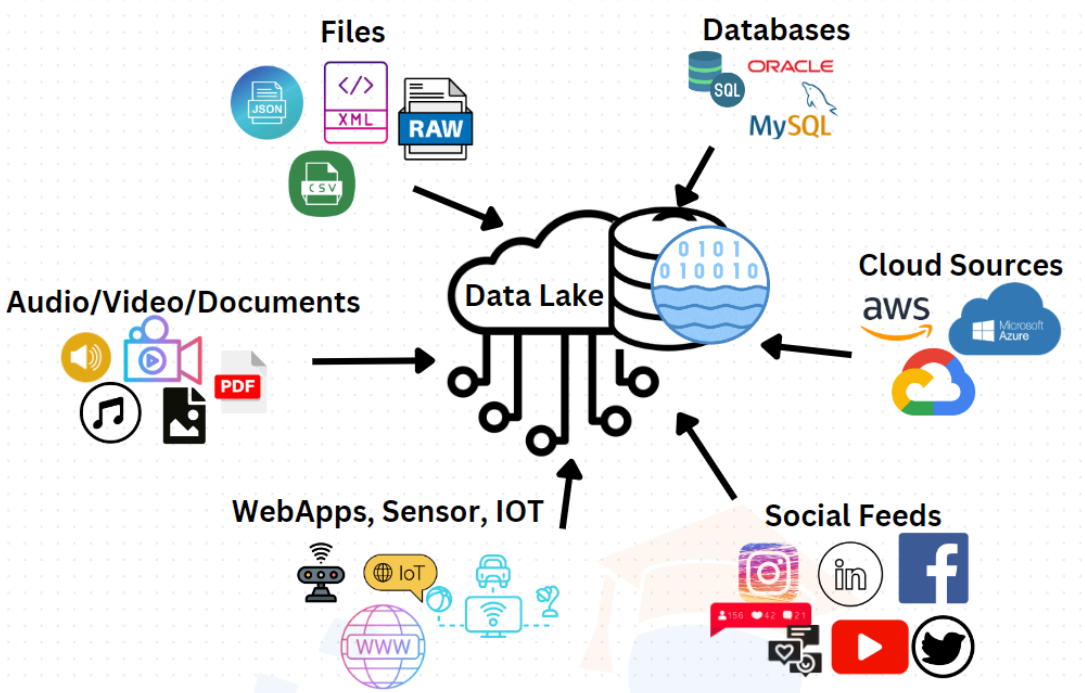
A Data Lake is a storage repository that holds vast amounts of raw data in its native format. Data lake stores all the data in the form of files.

It is very scalable and can store any kind of Data (mp3, mp4, CSV, JSON, XML, …)

Examples : Amazon S3, ADLS Gen2, Google Cloud Storage, etc.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide



Advantages of Data Lake

- Cost effective
- Scalable
- Any kind of data can be stored (Structured, Semi-Structured, Un-Structured)

Challenges of Data Lake

- ACID Guarantees are not provided

Any Database will provide ACID guarantees, this is required to maintain Data Consistency during transactions.

ACID Properties

Atomicity : Data should remain atomic implies that either all the transactions should be performed completely or should not be executed at all. A Rollback would take place in case of failed transactions.

Consistency : Data should always be correct and consistent before and after a transaction has taken place.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Isolation : Data reads should be allowed only on completion of a transaction. Data reads while the transaction is ongoing can lead to incomplete / inconsistent data giving incorrect results.

Durability : Ensures data permanency, that is the data after successful transaction becomes permanent in the database. Even in case of system failures, data still survives and doesn't hamper the results.

Scenarios where ACID guarantee is not achievable

1. **Job failing while Appending the data**

   Data Atomicity and Consistency is affected in this case.

2. **Job failing while Overwriting the data**

   Data Atomicity, Consistency & Durability is affected in this case.

3. **Simultaneous Reads & Writes**

   Data Isolation is affected in this case.

4. **Appending Data with a different Schema**

In the above mentioned scenarios, there are several challenges faced like

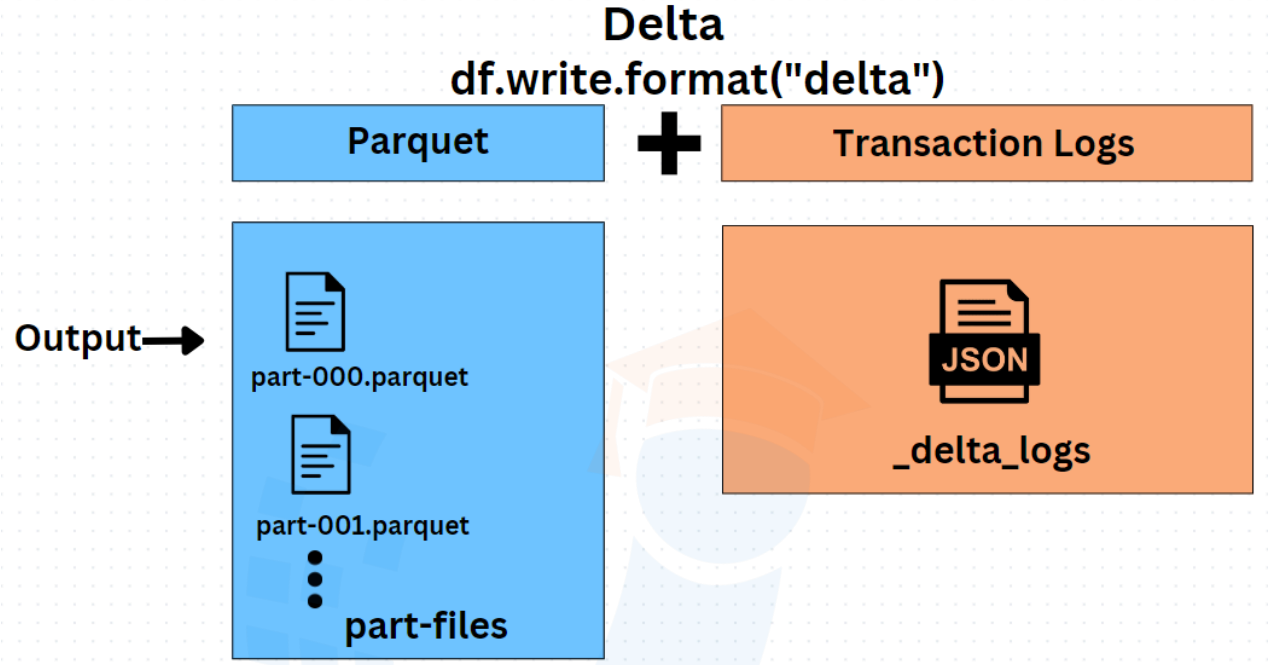
- Data Reliability Issues
- No Data Validation
- Data Corruption in case of failures
- DML operations cannot be performed as the ACID guarantee is not provided.
- Data Quality Issues
- No Data Versioning possible for historical data.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

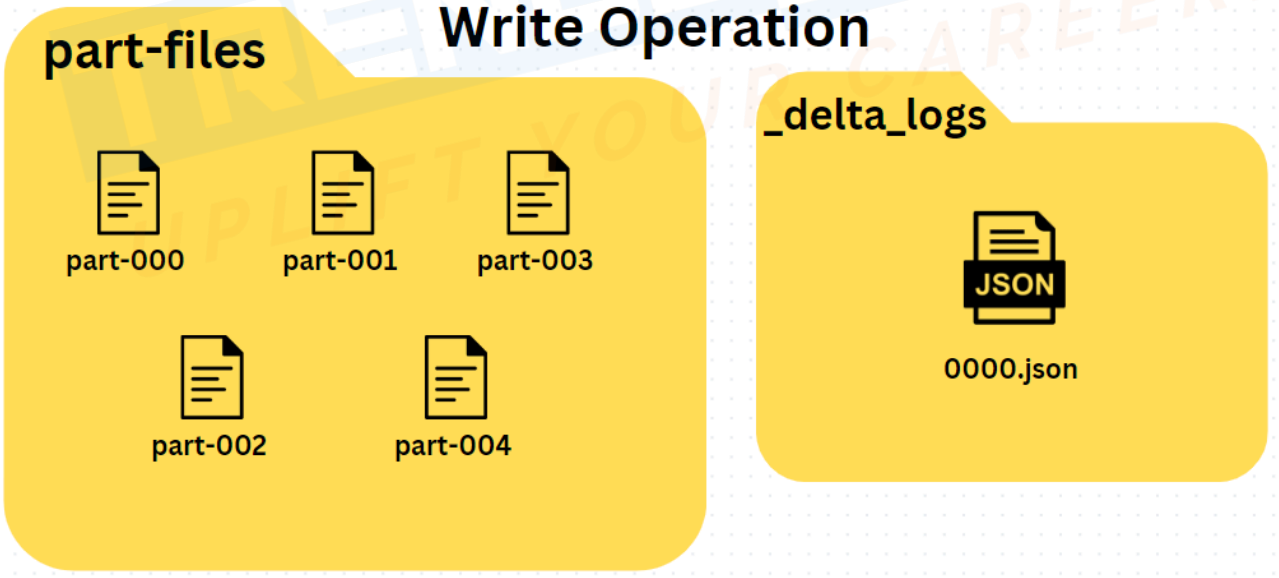
Delta Lake

Is an open source storage layer that acts like a small utility installed on Spark Cluster.



-Transaction log files will be generated only after all the part files are created and the entire job is executed completely.

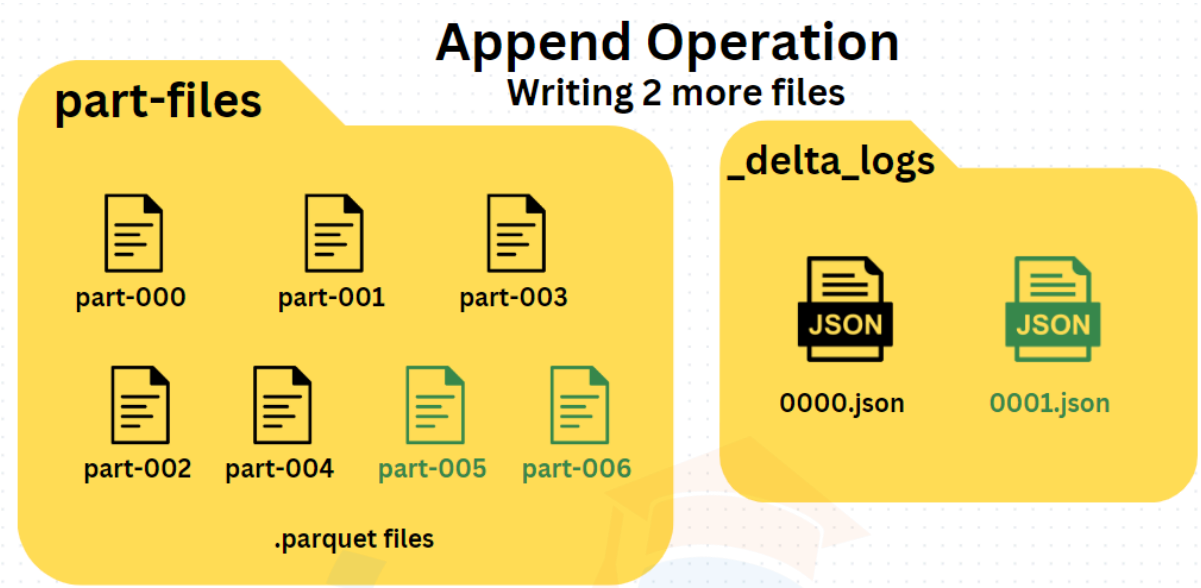
WRITE Operation



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

APPEND Operation



Process for a Write and Read Operation

For Write Operation	For Read Operation
• All the part files are written first. • Then a transaction log file is added to _delta_log folder in a JSON format	• First the transaction log files are read. • Then based on the logs, the required part files are read.

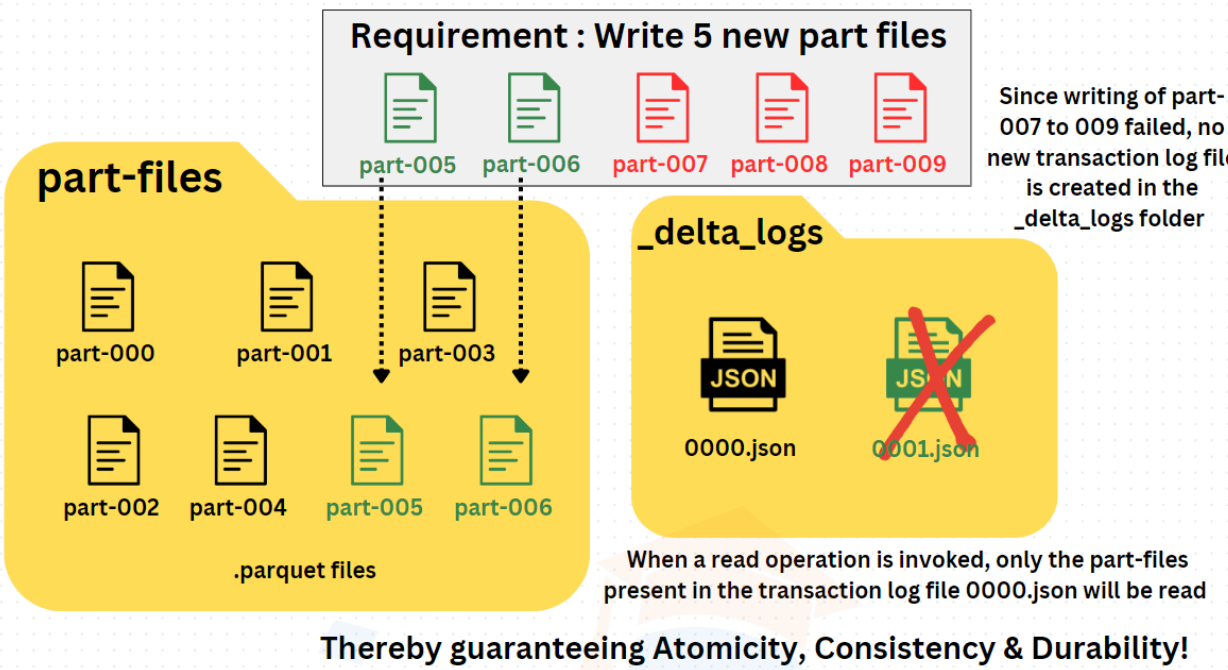
1. **Consider the Scenario - "Job failing while appending the data"**

In the case of Data Lake, this scenario led to issues that didn't guarantee data atomicity & consistency.

However, in the case of Delta Lake, with transaction logs coming into picture, this challenge was overcome.

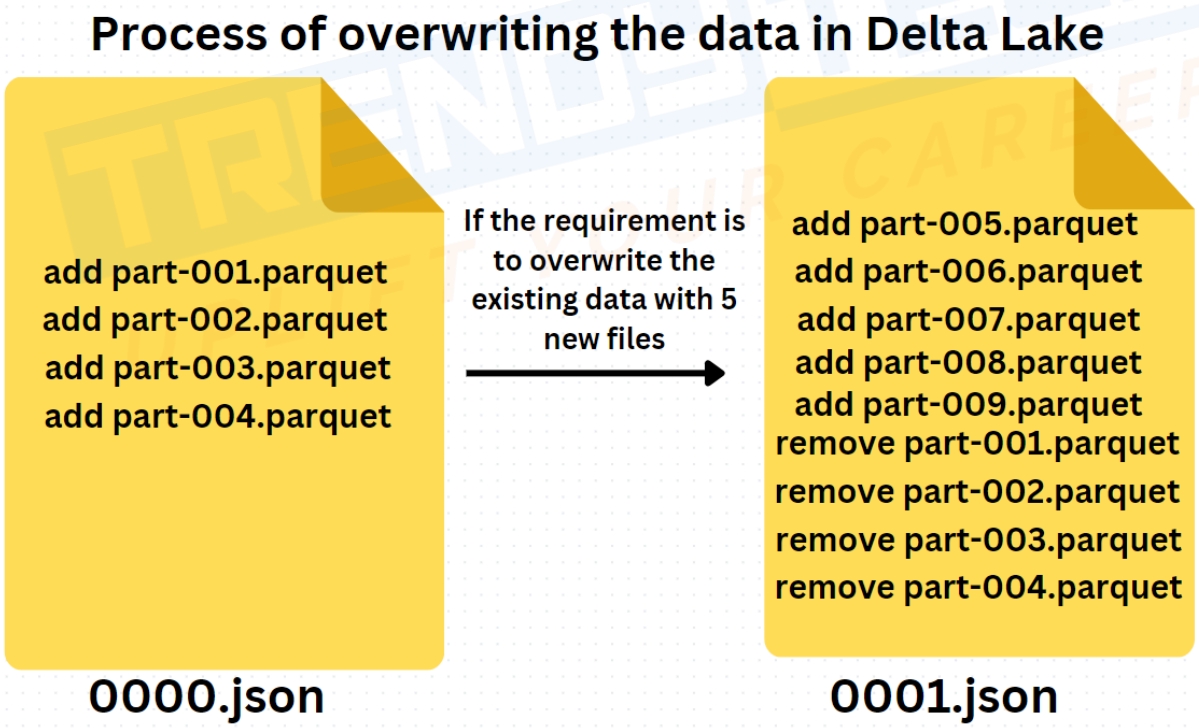
NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide



2. **Consider the Scenario - "Job failing while Overwriting the data"**

- In the case of Delta Lake, when an overwrite operation is triggered, all the existing files are not deleted initially.
- But rather all the new files are added and once the files are successfully written, a new transaction log file is generated which includes operations to remove the existing files after adding the new files.



NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

- If the job fails while adding the part-files and only partial data is written, then the new transaction log file will not be generated and the reader reads the previous files. This ensures atomicity, consistency & durability.

3. **Consider the Scenario - "Simultaneous Reads and Writes"**

- In the case of delta lake, transaction log files are generated only after all the part files are added successfully.
- This ensures that when a reader reads, no partial data will be read.
- Either the reader reads the previous part files (when the part files of new write operation are still under process)
- Or the reader reads all the newly added part files of the latest successful write operation.

Updates & Deletes in Delta Lake

- Lets see how delta lake supports DML operations like update and delete without compromising on ACID properties

When an update has to be made to a existing record present in a part file

1. A new part file will be created and the record to be updated will be added to the new part file with the required changes made.
2. The records which were present in the old part files that needed no changes will be copied directly to the new part file generated.
3. Then a transaction log file will be generated which has transactions indicating **add new part file** & **remove old part file**.

When a record has to be deleted from an existing part file

1. A new part file will be created and the record to be deleted will not be added to the new part file.
2. Rest of the records which were present in the old part files apart from the records to be deleted will be copied directly to the new part file generated.
3. Then a transaction log file will be generated which has transactions indicating **add new part file** & **remove old part file**.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

4. Consider the Scenario - “Appending Data with a different Schema”

- Schema changes is taken care effectively in case of delta lake
- Version history can be maintained efficiently in the case of delta lake with the use of **Time Travel** feature.
- Data quality checks can be applied with the use of commit logs.
- Performance improvement can be achieved with the help of commit logs that provide data statistics.

Delta Lake Practicals

Setup Steps:

1. Create a Resource Group
2. Creating a Databricks workspace in the Resource group and launching the workspace.
3. Create a Cluster in the Databricks workspace.
4. Create a Storage Account.
5. Create a new Container in the storage account.
6. Upload files in the container.
7. After the cluster is deployed, create a Notebook to execute the spark code to access the files in the storage account using mount.

   Creating a mount point /mnt/retaildb to access the files in the storage account

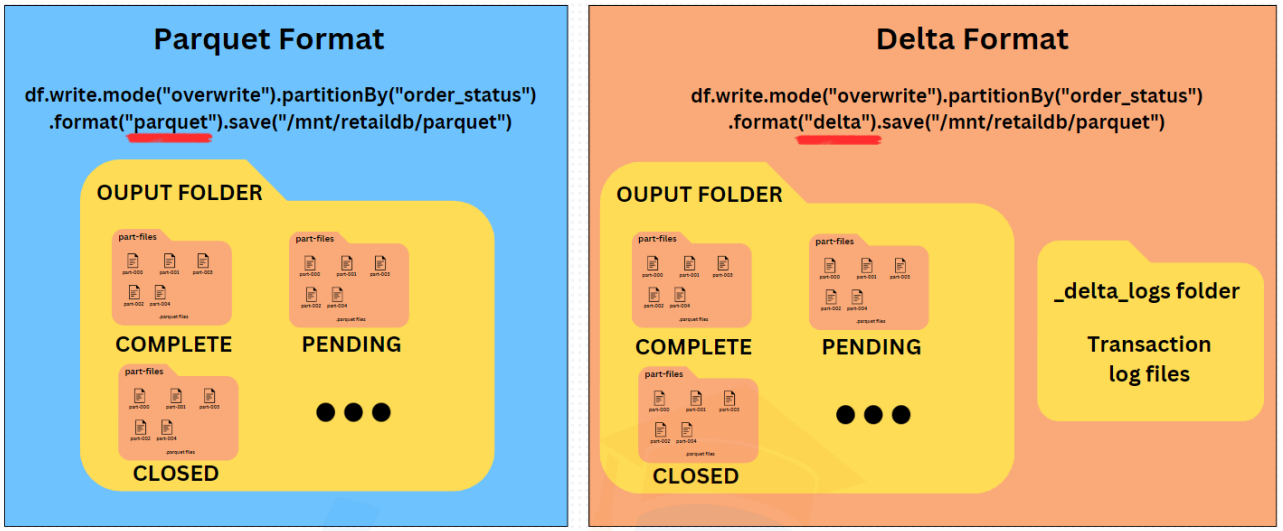
   Ex:

```
dbutils.fs.mount(

source = 'wasbs://input@week13demosa.blob.core.windows.net',

mount-point = '/mnt/retaildb',

extra_configs =
{'fs.azure.account.key.week13demosa.blob.core.windows.net' :
'vrCSZYdbaQp2Mcws3JIRqZlwfwmzjGaqBdE0nVQF/zeCwsJ0ud
/y6HCz71Aeb8R9QAm1MPjNmJ3r+AStlHYdeg=='}

)
```

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

8. Create a Dataframe to perform the needed transformations on the data.



9. Creating **Database** and **Delta Tables** on the data stored in the delta lake

Parquet Format

```
%sql
create database if not exists retaildb
```

```
%sql
create table retaildb.ordersparquet using parquet location
"/mnt/retaildb/parquet/orders.parquet/*"
```

parquet contains the schema of the table there schema enforcement required

Delta Format

```
%sql

create table retaildb.ordersdelta using delta location
"/mnt/retaildb/delta/orders.delta/*"
```

```
%sql
select * from retaildb.ordersdelta
```

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

Key points :

- History is available only for delta file formats as it provides ACID guarantee. (describe history <table-name>)
- If the table is dropped, the data is not deleted as it is an external table.
- So far, a **2 step process was used to create a Delta table**
    a. Adding the files to storage account
    b. Created an External table on top of the data present in the storage account
- However, it is possible to **create a table while writing the data to the dataframe in a single step**, which is as follows :

```
df.
write.
mode("overwrite").
partitionBy("order_status").
format("delta").
option("path", "/mnt/retaildb/delta/orders.delta").
saveAsTable("retaildb.ordersdeltatable")
```

10. Inserting data into Delta Table

There are 3 ways of inserting the data into the delta table

a. Insert Command

insert into retaildb,ordersdelta values ('111111111', '2023-07-25 00:00:00.0' , '222222222' , 'CLOSED')

b. Append Command

dfnew = spark.read.csv("/mnt/retaildb/ordersappend.csv" , header = True)

dfnew.write.mode("append").partitionBy("order_status").format("delta").save("/mnt/retaildb/delta/orders.delta")

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

c. **Copy Command**

A new file is added to the storage account and then copied using the following command

Example

%sql

copy into retaildb.ordersdelta from "/mnt/retaildb/orders1.csv" fileformat = csv format_options('header' = 'true')

11. Performing Update and Delete

- **Schema mismatch will be identified in case of Delta format** and an error occurs if data with a different schema is inserted, thereby ensuring data quality.
- **Delta format supports Schema Evolution.**
  dfnew.write.mode("append").partitionBy("order_status").format("delta").option("mergeSchema","true").save("/mnt/retaildb/delta/orders.delta")
- **Updating a record**
  Example
  %sql
  update retaildb.ordersdelta set order_status='CLOSED' where order_id = 5
  [
  On executing the above update, Say COMPLETE has 50 records, then 2 new files will be created
  a. COMPLETE will be created with 49 records
  b. CLOSED will be created with 1 record.

  Then a transaction log file will be created with respective entries of adding and removing the files.
  - remove complete with 50 records
  - add complete with 49 records
  - add closed with 1 record

  ]
- **Deleting a record**
  Example
  %sql
  delete from retaildb.ordersdelta where order_id = 5
  [
  On executing the above update, Say COMPLETE has 50 records, then 2 new files will be created
  a. COMPLETE will be created with 49 records

  Then a transaction log file will be created with respective entries of adding and removing the files.

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide

- remove complete with 50 records
- add complete with 49 records

]

12. Applying Constraints and Access History using **Time Travel**

Applying Constraints

NOT NULL
Example
%sql
alter table retaildb.ordersdelta alter column customer_id SET not NULL

CHECK
Example
%sql
alter table retaildb.ordersdelta add constraint status_ck check (order_status in ('ON_HOLD','PAYMENT_REVIEW','CLOSED','COMPLETE','CANCELED','SUSPECTED_FRAUD', 'PROCESSING','PENDING','PENDING_PAYMENT'))

Accessing History using Time Travel
Example
%sql
select * from retaildb.ordersdelta where order_id = 6

(doesn't display any results as in this latest version, this record is not present)

If we change the version to previous versions then the results will be displayed.
%sql
select * from retaildb.ordersdelta version as of 2 where order_id = 6

OR

%sql
select * from retaildb.ordersdelta timestamp as of '2022-09-23T14:01:01.000+0000'

%sql
restore table retaildb.ordersdelta to a version as of 0
(this will restore the current table to the first version)

NOT FOR DISTRIBUTION @Sumit Mittal www.trendytech.in

legal@trendytech.in
www.trendytech.in
©2023 All Rights Reserved Worldwide