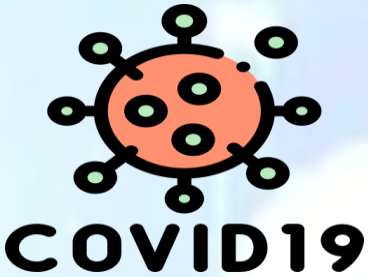


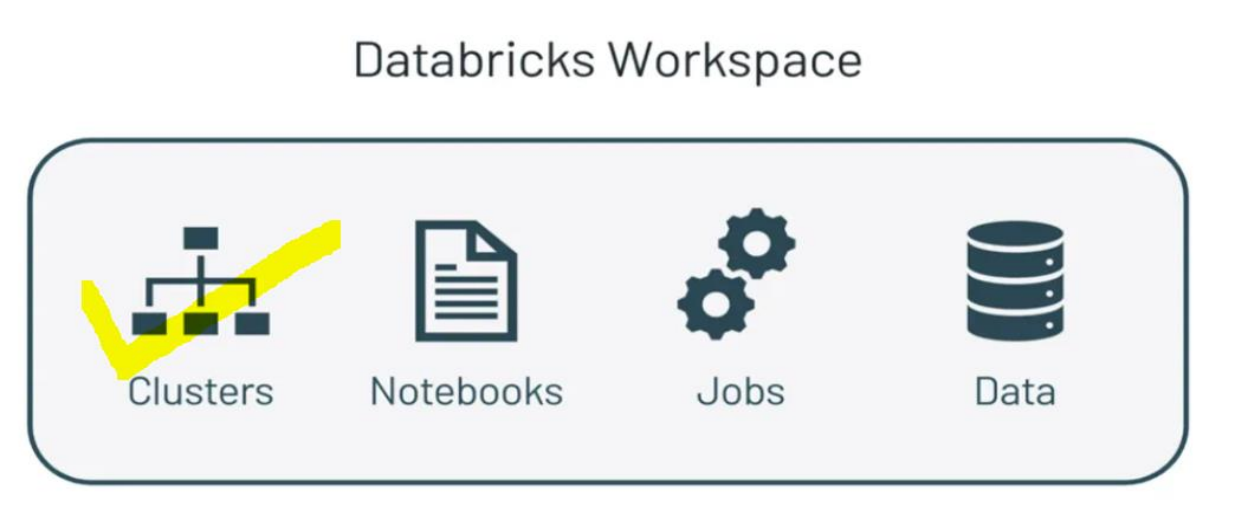
Business Use Case:

- **We have real-time COVID-19 data stored in Amazon S3. Our goal is to connect Databricks to this S3 bucket and perform analysis on the data.**
- **The data is in CSV format. We need to convert it into tables and conduct analysis based on specific metrics, focusing on COVID-19 trends and insights.**

Step-by-Step Guide for End-to-End AWS Databricks Project

Step 1: Set Up the Databricks Cluster

1. **Open Databricks:**
   - Start by logging into your Databricks workspace.
   - Navigate to the "Clusters" tab.
2. **Create a Cluster:**
   - Click on "Create Cluster."
   - Name the cluster, e.g., "Testing1."
   - Use the default settings, which typically include 15 GB memory.
   - Click "Create" and wait for the cluster to initialize.



Step 2: Prepare Data in Amazon S3

1. **Access Amazon S3:**
   - Log in to your AWS Management Console.
   - Search for "S3" and open it.
2. **Create an S3 Bucket:**
   - Click "Create Bucket."
   - Name the bucket, e.g., "testing45678."
   - Choose the region (e.g., North Virginia).
   - Use default settings and click "Create Bucket."
3. **Upload Data to S3:**
   - Click on the newly created bucket.
   - Click "Upload" and then "Add File."
   - Select your CSV files, e.g., "employee_covid_vaccine_statewise.csv."
   - Click "Upload" to upload the files to the S3 bucket.

Step 3: Connect Databricks to Amazon S3

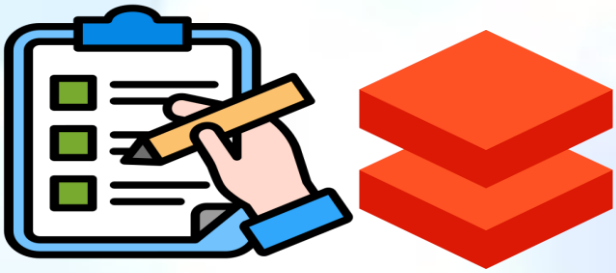
1. **Obtain AWS Credentials:**
    - In the AWS Management Console, go to the "IAM" service.
    - Navigate to "Users" and click "Add User."
    - Create a username (e.g., "employee").
    - Select "Programmatic access."
    - Assign the "AdministratorAccess" policy.
    - Review and create the user.
    - Download the CSV file containing the Access Key ID and Secret Access Key.
2. **Mount S3 Bucket to Databricks:**
    - In Databricks, go to your workspace.
    - Open a new notebook and run the following script:

Code Explanation:

1. **Listing Files in the Directory:**

```
dbutils.fs.ls("/Filestore/tables")
```

Purpose: This command lists all the files in the "/Filestore/tables" directory within the Databricks environment, ensuring that the files you need are present.



2. Importing Required Libraries:

```
from pyspark.sql.functions import *
import urllib
```

Purpose:

- pyspark.sql.functions: This library contains useful functions for DataFrame operations.
- urllib: This library is used to encode the AWS secret key for secure URL handling

3. **Defining File Type and Reading Data:**

```
file_type = "csv"
first_row_is_header = "true"
delimiter = ","

aws_keys_df = spark.read.format(file_type) \
    .option("header", first_row_is_header) \
    .option("sep", delimiter) \
    .load("/Filestore/tables/new_user_credentials-1.csv")
```

Purpose:

- file_type: Defines that the file format is CSV.
- first_row_is_header: Indicates that the first row of the CSV file contains headers.
- delimiter: Sets the delimiter as a comma.
- **Action:** Reads the CSV file containing AWS credentials into a Spark
- DataFrame (aws_keys_df).

4. **Extracting AWS Access and Secret Keys:**

```
ACCESS_KEY = aws_keys_df.where(col('User name') ==
'data').select('Access key ID').collect()[0]['Access key ID']
SECRET_KEY = aws_keys_df.where(col('User name') ==
'data').select('Secret access key').collect()[0]['Secret access key']
```

Purpose: Extracts the Access key ID and Secret access key from the DataFrame, filtering by 'User name' equal to 'data'.

5. **Encoding the Secret Key:**

```
ENCODED_SECRET_KEY = urllib.parse.quote(string=SECRET_KEY,
safe="")
```

Purpose: Encodes the secret key using urllib.parse.quote() to ensure it's safely included in the URL.

6. Mounting the AWS S3 Bucket:

```
AWS_S3_BUCKET = "hellowdmdkdk"
MOUNT_NAME = "/mnt/hellowdmdkdk"
SOURCE_URL = "s3n://{}:{}@{}".format(ACCESS_KEY,
ENCODED_SECRET_KEY, AWS_S3_BUCKET)

dbutils.fs.mount(SOURCE_URL, MOUNT_NAME)
```

Purpose:

- AWS_S3_BUCKET: Name of the S3 bucket.
- MOUNT_NAME: Local mount point for accessing the S3 bucket.
- SOURCE_URL: Constructs the S3 URL using access and secret keys.
- **Action:** Mounts the S3 bucket to the specified mount point in Databricks.

7. **Listing Files in the Mounted Directory:**

```
%fs ls "/mnt/hellowdmdkdk/"
```

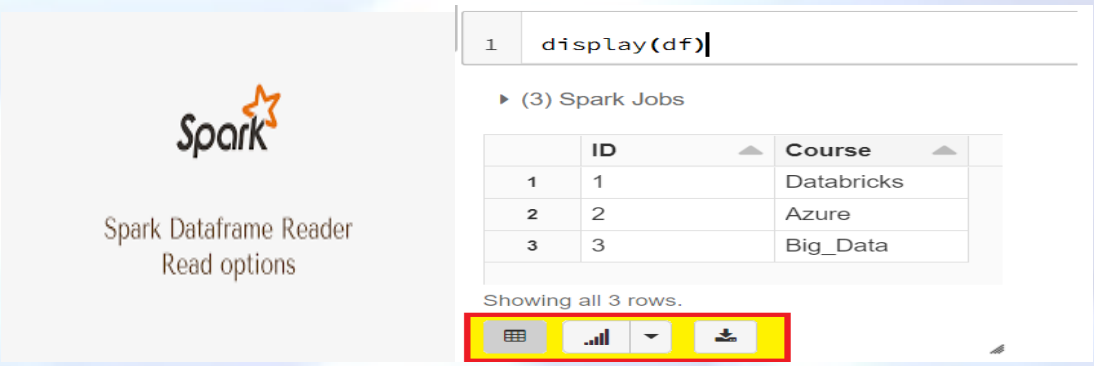
Purpose: Lists the files in the mounted S3 directory to verify that the mount was successful.

8. Reading and Displaying Data from the Mounted Directory:

```
file_location = "/mnt/hellowdmdkdk/employee.csv"
file_type = "csv"

infer_schema = "true"
first_row_is_header = "true"
delimiter = ","

df = spark.read.format(file_type) \
    .option("inferSchema", infer_schema) \
    .option("header", first_row_is_header) \
    .option("sep", delimiter) \
    .load(file_location)

display(df)
```

Purpose:

- file_location: Specifies the file path within the mounted S3 directory.
- **Action:** Reads the CSV file into a Spark DataFrame and displays it.



9. Saving the DataFrame as a Table:

```
df.write.mode("overwrite").saveAsTable("employee")
```

Purpose: Saves the DataFrame as a table named "employee" in the Databricks environment. The overwrite mode ensures any existing table with the same name is replaced.