Database & Python ETL

Provision one database of your choosing (SQL, NoSQL, Graph). Write a python ETL that ingests the provided data, transforms it in some way, and loads it into the database. This should be reproducible code with documentation. (Terraform / CloudFormation / Ansible, docker-compose etc.).

Introduction:

I have chosen the **Databricks Platform** for building the ETL, as The Databricks Lakehouse Platform combines the best elements of **data lakes** and **data warehouses.** It empowers data scientists, data engineers and data analysts with a simple collaborative environment to run interactive and scheduled data analysis workloads.

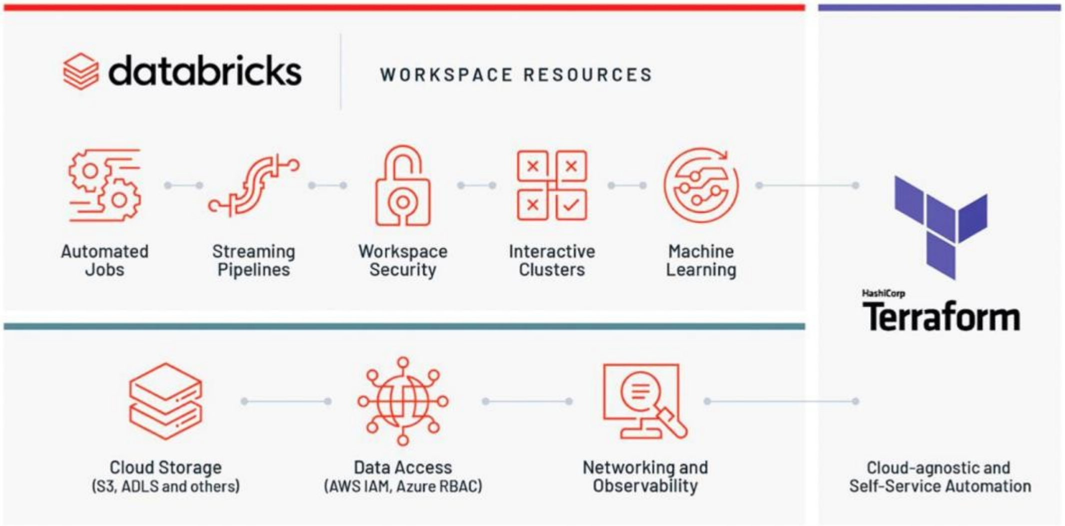
Databricks Terraform provider is to support all Databricks REST APIs, supporting automation of the most complicated aspects of deploying and managing your data platforms. We can deploy and manage clusters and jobs and to configure data access.

Objective:

Goal is to ingest the data, transform it and load it into database tables. Automate complicated aspects of deploying and managing your data platforms using Terraform provider.

Architecture:

Build ETL in databricks notebook, created database in databricks lake house. Used Terraform to automate deployment


databricks
WORKSPACE RESOURCES
Automated Jobs
Streaming Pipelines
Workspace Security
Interactive Clusters
Machine Learning
HashiCorp Terraform
Cloud Storage (S3, ADLS and others)
Data Access (AWS IAM, Azure RBAC)
Networking and Observability
Cloud-agnostic and Self-Service Automation

Terraform supports management of all Databricks resources and underlying cloud infrastructure

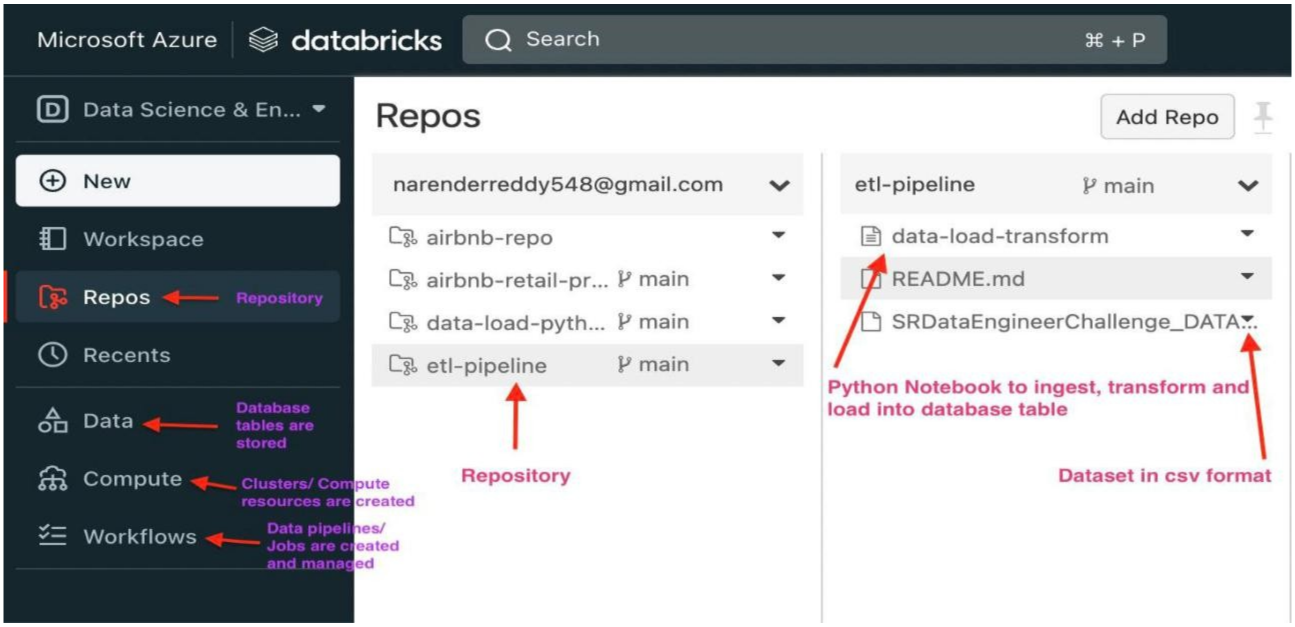
Implementation of the use case

Step 1: Create Azure Databricks Workspace
Step 2: Integrate with GitHub
Step 3: Add repository in Databricks
Step 4: Upload csv file into Databricks Repo
Step 5: Create Python notebook
Step 6: Python code for the below tasks

- **Data Extraction**
  - Define the widgets to enter the values of the parameters
  - Assign parameter values to variables
  - Read the csv file and create pandas dataframe
- **Data Transformation**
  - Convert column datatype to int
  - Convert pandas dataframe to spark dataframe
- **Data Loading**
  - Define source, database name and table name
  - Database Creation
  - Write/Load the data from dataframe to table
- **Query the data from the table**

Step 7: Create Databricks Job to automate the task
Step 8: Create Job
Step 9: Terraform configuration

Structure of the project



Repository

Database tables are stored

Clusters/ Compute resources are created

Data pipelines/ Jobs are created and managed

Repository

Python Notebook to ingest, transform and load into database table

Dataset in csv format

Step 1: Create Azure Databricks Workspace

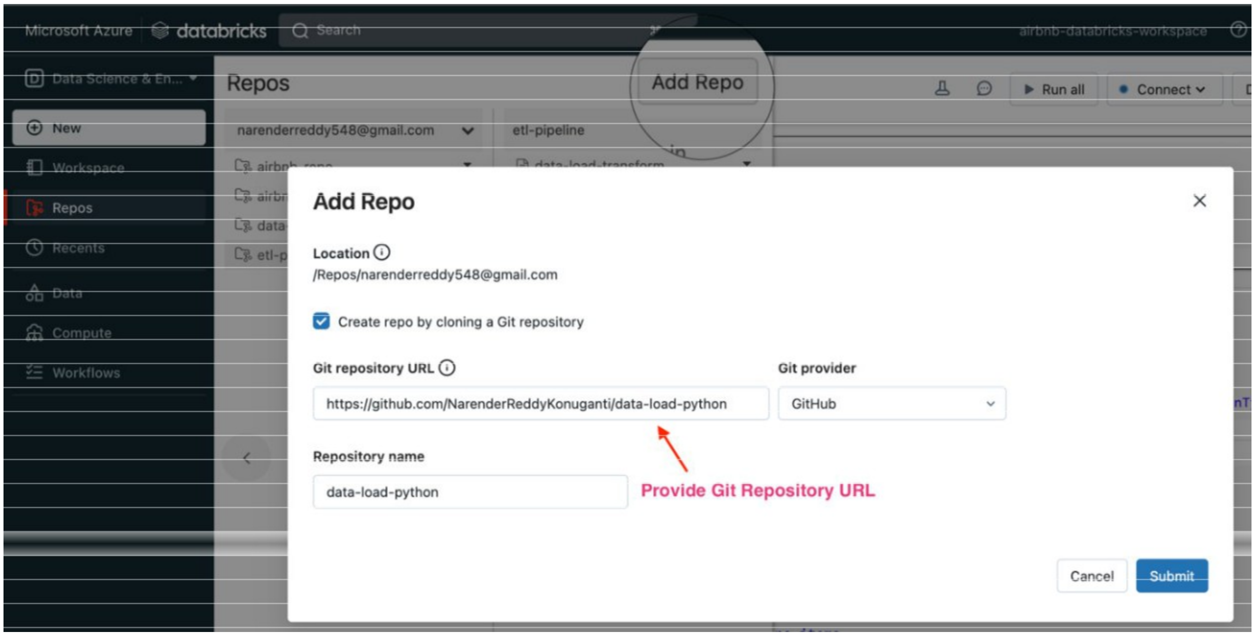
Follow the steps in https://learn.microsoft.com/en-us/azure/databricks/getting-started/ to create Azure Workspace

Step 2: Integrate with GitHub

Integrate GIT with Databricks using access token, click here for steps Git-Integration

Step 3: Add repository in Databricks

Navigate to workspace ‡ Repos ‡ Add Repo ‡ Provide Git repository URL



Add Repo

Location
/Repos/narenderreddy548@gmail.com

☑ Create repo by cloning a Git repository

Git repository URL: https://github.com/NarenderReddyKonuganti/data-load-python

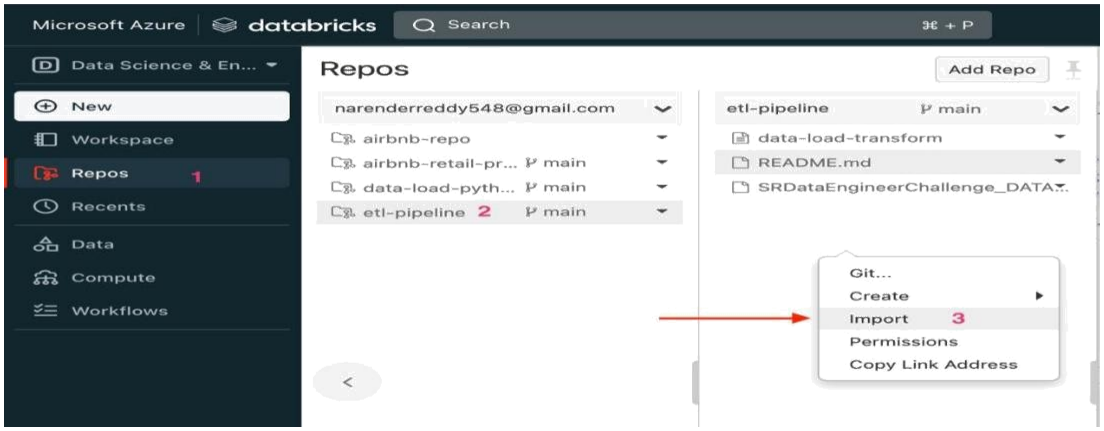
Git provider: GitHub

Repository name: data-load-python

Provide Git Repository URL

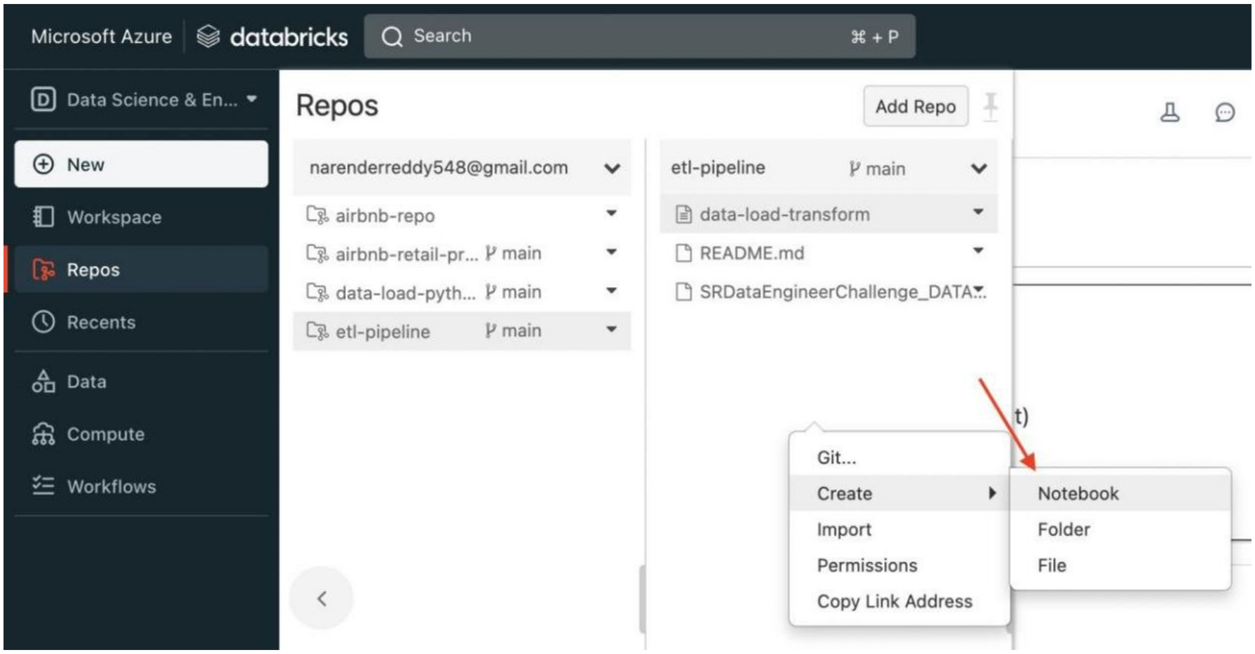
Step 4: Upload csv file into Databricks Repo

Navigate to the etl-pipeline repo and right click ‡ import ‡ upload csv file



Step 5: Create Python notebook

Navigate to the etl-pipeline repo and right click ‡ create ‡ notebook



<u>Create a Cluster:</u>

To do exploratory data analysis and data engineering, create a cluster to provide the compute resources needed to execute commands.

∑ Click **Compute** in the sidebar.

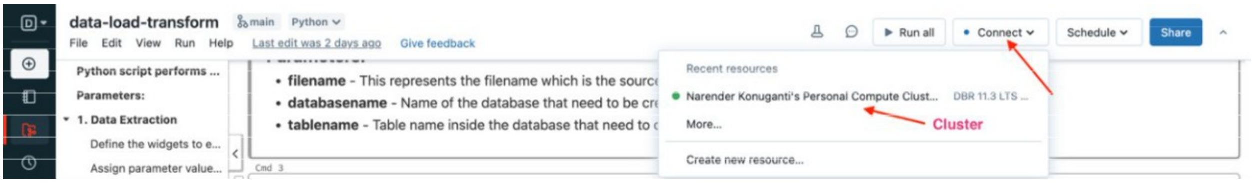
∑ On the Compute page, click **Create Cluster**. This opens the New Cluster page.

∑ Specify a unique name for the cluster, leave the remaining values in their default state, and click **Create Cluster**.

<u>Attach notebook to cluster:</u>

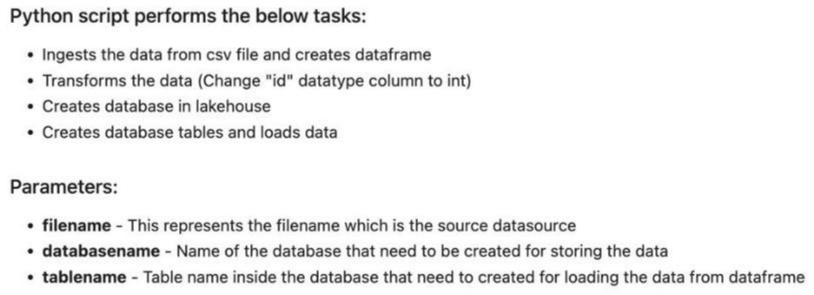
Navigate to the etl-pipeline repo and click on notebook “data-load-transform”

Click on connect and click cluster



- **filename** - This represents the filename which is the sourc
- **databasename** - Name of the database that need to be cr
- **tablename** - Table name inside the database that need to c

Step 6: Python code for the below tasks



Python script performs the below tasks:

- Ingests the data from csv file and creates dataframe
- Transforms the data (Change "id" datatype column to int)
- Creates database in lakehouse
- Creates database tables and loads data

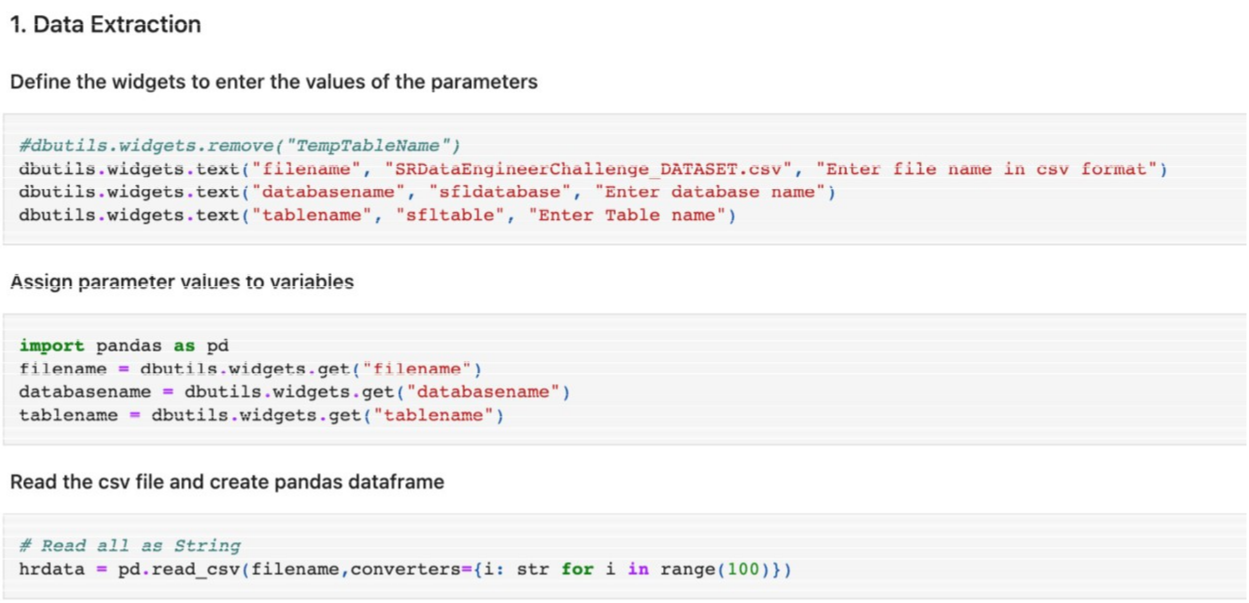
Parameters:

- **filename** - This represents the filename which is the source datasource
- **databasename** - Name of the database that need to be created for storing the data
- **tablename** - Table name inside the database that need to created for loading the data from dataframe

Data Extraction

- Define the widgets to enter the values of the parameters
    - **Widgets -** Main purpose of widgets is to **parameterize** notebooks. These are used to define the parameters for the notebook.
    - When we **parameterize**, code can be directly deployed in production and run with updated parameter values as per PROD.
    - There are 3 parameters used which is described in above screenshot.

- Assigned parameter values to variables
- Read the csv file and created pandas dataframe



1. Data Extraction

Define the widgets to enter the values of the parameters

```
#dbutils.widgets.remove("TempTableName")
dbutils.widgets.text("filename", "SRDataEngineerChallenge_DATASET.csv", "Enter file name in csv format")
dbutils.widgets.text("databasename", "sfldatabase", "Enter database name")
dbutils.widgets.text("tablename", "sfltable", "Enter Table name")
```

Assign parameter values to variables

```
import pandas as pd
filename = dbutils.widgets.get("filename")
databasename = dbutils.widgets.get("databasename")
tablename = dbutils.widgets.get("tablename")
```

Read the csv file and create pandas dataframe

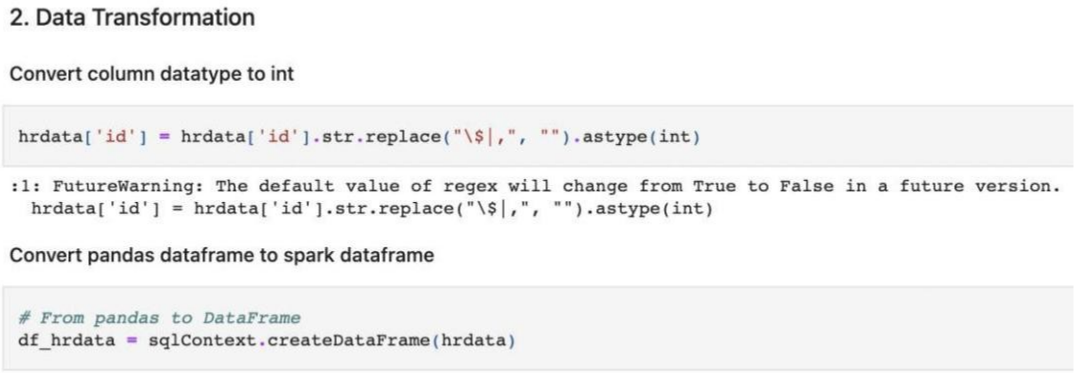
```
# Read all as String
hrdata = pd.read_csv(filename,converters={i: str for i in range(100)})
```

Data Transformation

∑ Converted column datatype to int

○ <u>Transformation</u> - Changed data type of column “id” from **string** to **int**.

∑ Converted pandas dataframe to spark dataframe



2. Data Transformation

Convert column datatype to int

```
hrdata['id'] = hrdata['id'].str.replace("\$|,", "").astype(int)
```

```
:1: FutureWarning: The default value of regex will change from True to False in a future version.
  hrdata['id'] = hrdata['id'].str.replace("\$|,", "").astype(int)
```

Convert pandas dataframe to spark dataframe

```
# From pandas to DataFrame
df_hrdata = sqlContext.createDataFrame(hrdata)
```

Data Loading

∑ Define source, database name and table name

∑ Database Creation

∑ Write/Load the data from dataframe to table

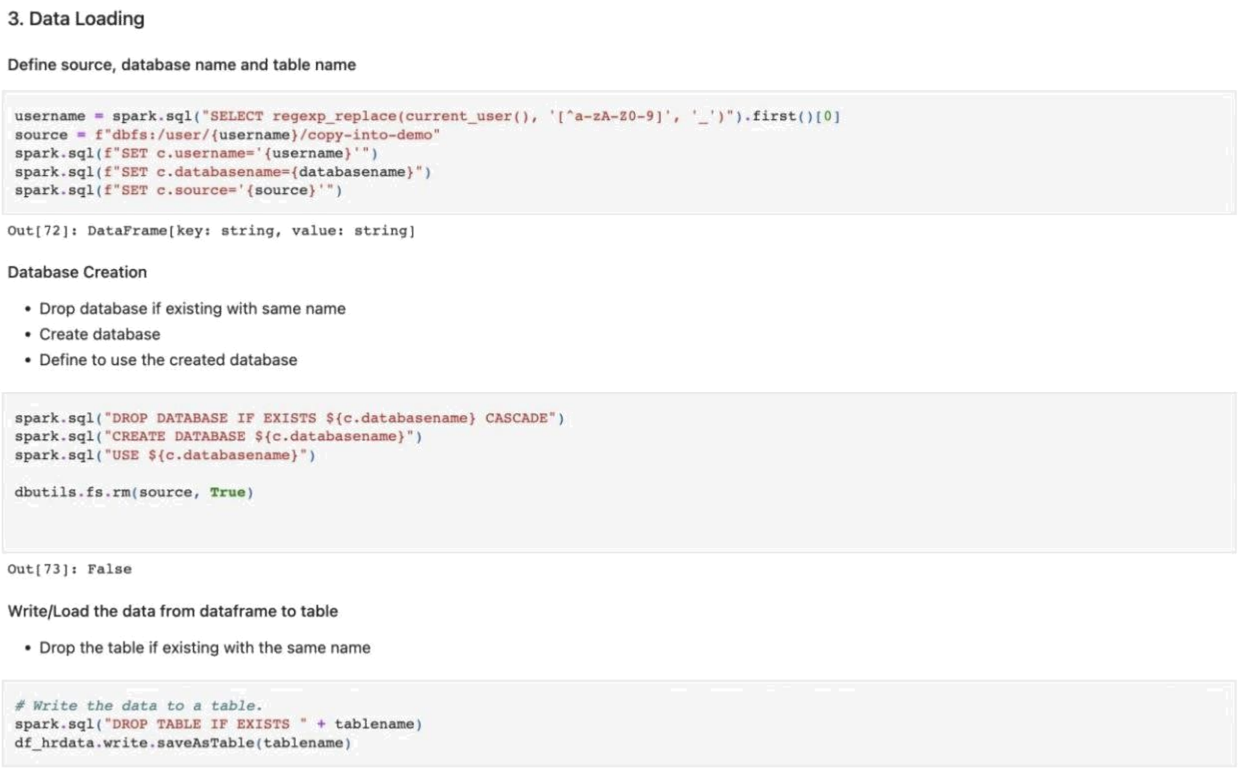
❖ In the first step, we have defined the source where the database tables are to the created

❖ Specified **database** name and **table name**

❖ Created database in databricks lake house

❖ Created table inside the database

❖ Load the transformed data from dataframe to **database table**



3. Data Loading

Define source, database name and table name

```
username = spark.sql("SELECT regexp_replace(current_user(), '[^a-zA-Z0-9]', '_')").first()[0]
source = f"dbfs:/user/{username}/copy-into-demo"
spark.sql(f"SET c.username='{username}'")
spark.sql(f"SET c.databasename={databasename}")
spark.sql(f"SET c.source='{source}'")
```

```
Out[72]: DataFrame[key: string, value: string]
```

Database Creation

- Drop database if existing with same name
- Create database
- Define to use the created database

```
spark.sql("DROP DATABASE IF EXISTS ${c.databasename} CASCADE")
spark.sql("CREATE DATABASE ${c.databasename}")
spark.sql("USE ${c.databasename}")

dbutils.fs.rm(source, True)
```

```
Out[73]: False
```

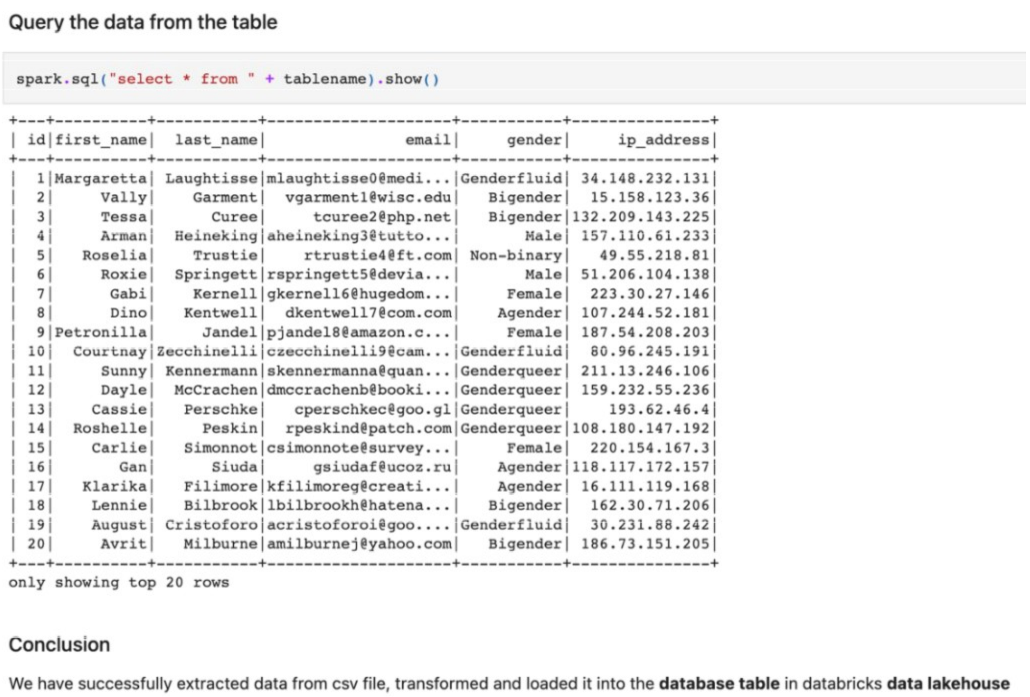
Write/Load the data from dataframe to table

- Drop the table if existing with the same name

```
# Write the data to a table.
spark.sql("DROP TABLE IF EXISTS " + tablename)
df_hrdata.write.saveAsTable(tablename)
```

Query the data from the table

Below is result of the data retrieved from table.



Query the data from the table

```
spark.sql("select * from " + tablename).show()
```

```
+---+----------+-----------+--------------------+-----------+---------------+
| id|first_name| last_name|               email|     gender|     ip_address|
+---+----------+-----------+--------------------+-----------+---------------+
|  1|Margaretta| Laughtisse|mlaughtisse0@medi...|Genderfluid| 34.148.232.131|
|  2|     Vally|    Garment|  vgarment1@wisc.edu|   Bigender|  15.158.123.36|
|  3|     Tessa|      Curee|     tcuree2@php.net|   Bigender|132.209.143.225|
|  4|     Arman|  Heineking|aheineking3@tutto...|       Male| 157.110.61.233|
|  5|   Roselia|    Trustie|    rtrustie4@ft.com| Non-binary|   49.55.218.81|
|  6|     Roxie|  Springett|rspringett5@devia...|       Male| 51.206.104.138|
|  7|      Gabi|    Kernell|gkernell6@hugedom...|     Female|  223.30.27.146|
|  8|      Dino|   Kentwell|  dkentwell7@com.com|    Agender| 107.244.52.181|
|  9|Petronilla|     Jandel|pjandel8@amazon.c...|     Female| 187.54.208.203|
| 10|  Courtnay|Zecchinelli|czecchinelli9@cam...|Genderfluid|  80.96.245.191|
| 11|     Sunny| Kennermann|skennermanna@quan...|Genderqueer| 211.13.246.106|
| 12|     Dayle|  McCrachen|dmccrachenb@booki...|Genderqueer| 159.232.55.236|
| 13|    Cassie|   Perschke|   cperschkec@goo.gl|Genderqueer|    193.62.46.4|
| 14|  Roshelle|     Peskin|  rpeskind@patch.com|Genderqueer|108.180.147.192|
| 15|    Carlie|   Simonnot|csimonnote@survey...|     Female|  220.154.167.3|
| 16|       Gan|      Siuda|     gsiudaf@ucoz.ru|    Agender|118.117.172.157|
| 17|   Klarika|   Filimore|kfilimoreg@creati...|    Agender| 16.111.119.168|
| 18|    Lennie|   Bilbrook|lbilbrookh@hatena...|   Bigender|  162.30.71.206|
| 19|    August| Cristoforo|acristoforoi@goo....|Genderfluid|  30.231.88.242|
| 20|     Avrit|   Milburne|amilburnej@yahoo.com|   Bigender| 186.73.151.205|
+---+----------+-----------+--------------------+-----------+---------------+
only showing top 20 rows
```

Conclusion

We have successfully extracted data from csv file, transformed and loaded it into the **database table** in databricks **data lakehouse**

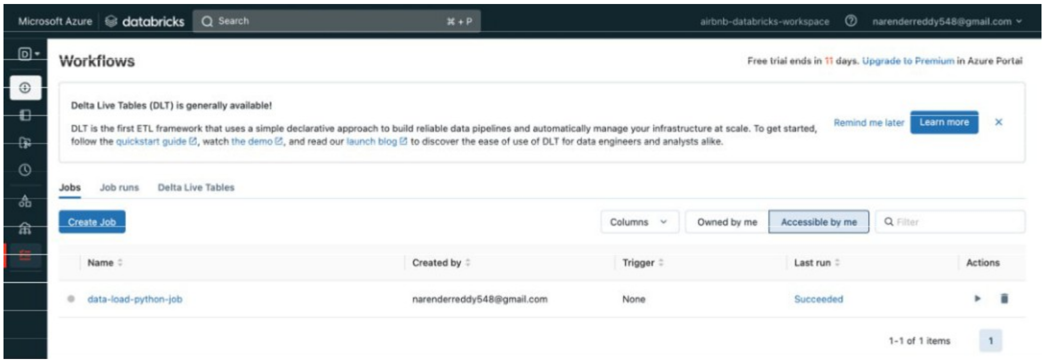
We can utilize the above data for running visualization and running ML models in Databricks.

Step 8: Create Job

∑ Click on **Workflows** in the sidebar.

∑ On the Workflows page, click Create Job. This opens the New Job page.

∑ Specify a job name, git location, notebook name, Cluster name and parameters.



Microsoft Azure | databricks | Search ⌘ + P | airbnb-databricks-workspace | narenderreddy548@gmail.com

Workflows

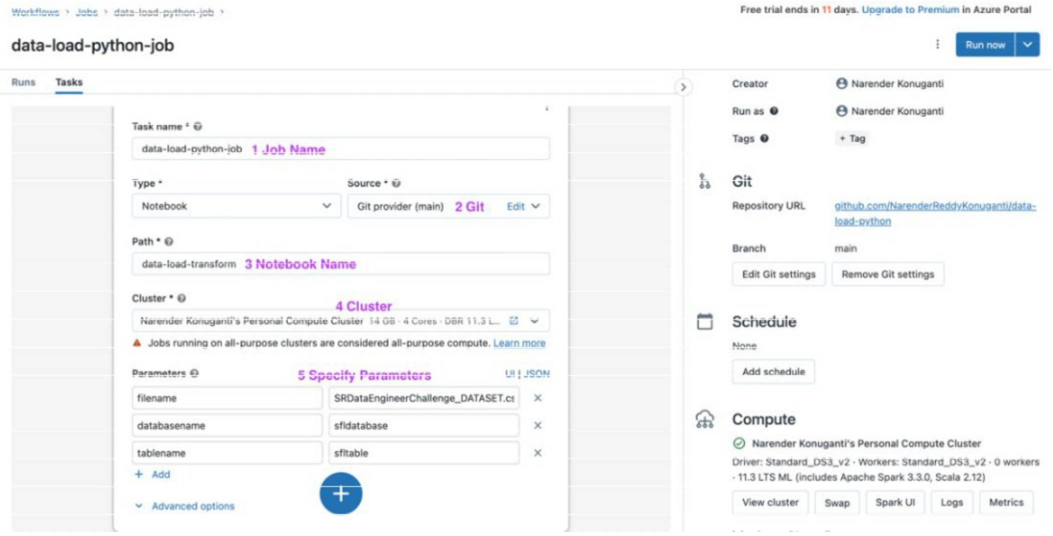
Free trial ends in 11 days. Upgrade to Premium in Azure Portal

Delta Live Tables (DLT) is generally available!

DLT is the first ETL framework that uses a simple declarative approach to build reliable data pipelines and automatically manage your infrastructure at scale. To get started, follow the quickstart guide, watch the demo, and read our launch blog to discover the ease of use of DLT for data engineers and analysts alike.

Name	Created by	Trigger	Last run	Actions
data-load-python-job	narenderreddy548@gmail.com	None	Succeeded	

1-1 of 1 items 1



Workflows › Jobs › data-load-python-job ›

data-load-python-job

Free trial ends in 11 days. Upgrade to Premium in Azure Portal

Task name *: data-load-python-job 1 Job Name

Type *: Notebook

Source *: Git provider (main) 2 Git Edit

Path *: data-load-transform 3 Notebook Name

Cluster *: Narender Konuganti's Personal Compute Cluster 14 GB · 4 Cores · DBR 11.3 L... 4 Cluster

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

Parameters 5 Specify Parameters UI | JSON

filename	SRDataEngineerChallenge_DATASET.cs
databasename	sfldatabase
tablename	sfltable

Creator: Narender Konuganti

Run as: Narender Konuganti

Tags: + Tag

Git

Repository URL: github.com/NarenderReddyKonuganti/data-load-python

Branch: main

Schedule

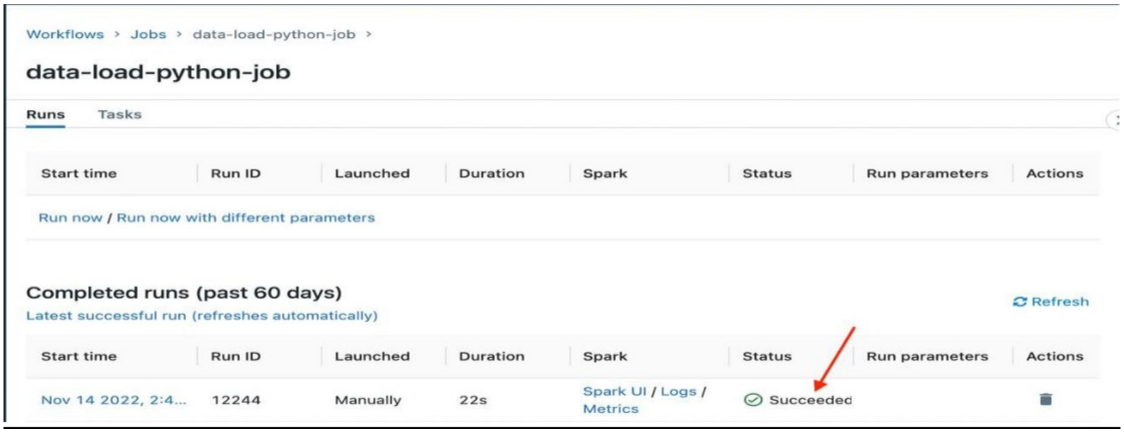
None

Compute

Narender Konuganti's Personal Compute Cluster

Driver: Standard_DS3_v2 · Workers: Standard_DS3_v2 · 0 workers · 11.3 LTS ML (includes Apache Spark 3.3.0, Scala 2.12)

<u>Job Run is successful</u>



Workflows › Jobs › data-load-python-job ›

data-load-python-job

Start time	Run ID	Launched	Duration	Spark	Status	Run parameters	Actions

Run now / Run now with different parameters

Completed runs (past 60 days)

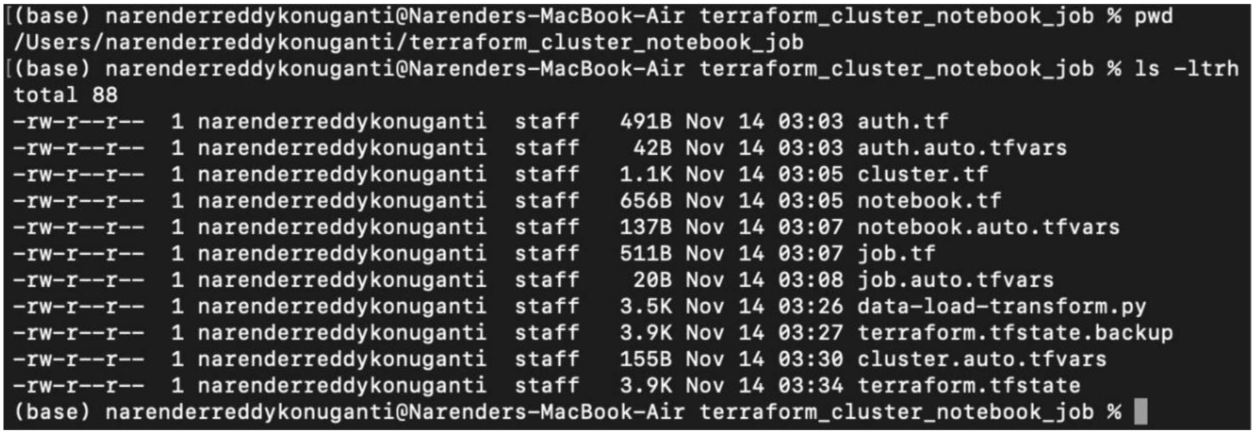
Latest successful run (refreshes automatically)

Start time	Run ID	Launched	Duration	Spark	Status	Run parameters	Actions
Nov 14 2022, 2:4...	12244	Manually	22s	Spark UI / Logs / Metrics	Succeeded		

Step 9: Terraform configuration

Please follow the steps from Configure Terraform Authentication to *install requirements* and *Configure Terraform authentication* to authenticate the **Databricks Terraform provider** with your **Azure Databricks account** and your Azure Databricks workspace,

Follow sample configuration to provision a Databricks notebook, cluster, and a job to run the notebook on the cluster, in an existing Databricks workspace.



```
[(base) narenderreddykonuganti@Narenders-MacBook-Air terraform_cluster_notebook_job % pwd
 /Users/narenderreddykonuganti/terraform_cluster_notebook_job
[(base) narenderreddykonuganti@Narenders-MacBook-Air terraform_cluster_notebook_job % ls -ltrh
 total 88
 -rw-r--r--  1 narenderreddykonuganti  staff   491B Nov 14 03:03 auth.tf
 -rw-r--r--  1 narenderreddykonuganti  staff    42B Nov 14 03:03 auth.auto.tfvars
 -rw-r--r--  1 narenderreddykonuganti  staff   1.1K Nov 14 03:05 cluster.tf
 -rw-r--r--  1 narenderreddykonuganti  staff   656B Nov 14 03:05 notebook.tf
 -rw-r--r--  1 narenderreddykonuganti  staff   137B Nov 14 03:07 notebook.auto.tfvars
 -rw-r--r--  1 narenderreddykonuganti  staff   511B Nov 14 03:07 job.tf
 -rw-r--r--  1 narenderreddykonuganti  staff    20B Nov 14 03:08 job.auto.tfvars
 -rw-r--r--  1 narenderreddykonuganti  staff   3.5K Nov 14 03:26 data-load-transform.py
 -rw-r--r--  1 narenderreddykonuganti  staff   3.9K Nov 14 03:27 terraform.tfstate.backup
 -rw-r--r--  1 narenderreddykonuganti  staff   155B Nov 14 03:30 cluster.auto.tfvars
 -rw-r--r--  1 narenderreddykonuganti  staff   3.9K Nov 14 03:34 terraform.tfstate
 (base) narenderreddykonuganti@Narenders-MacBook-Air terraform_cluster_notebook_job % 
```

me.tf --> This file gets information about the current databricks user
notebook.tf -->This file represents the notebook
notebook.auto.tfvars --> This file specifies the notebook's properties like notebook filename and language
data-load-transform.py --> This file represents the notebook's contents.
cluster.tf --> This file represents the cluster
cluster.auto.tfvars --> This file specifies the cluster's properties like cluster name and number of workers
job.tf --> This file represents the job that runs the notebook on the cluster.
job.auto.tfvars --> This file specifies the job's properties.

- Run terraform plan. If there are any errors, fix them, and then run the command again.

- **Run terraform apply**

When the above command is invoked, below are sequence of steps followed:

- Reads file **me.tf** and authenticates the user
- Reads **notebook.tf**, **notebook.auto.tfvars** and create notebook with content taken from **data-load-transform.py**
- Reads **cluster.tf**, **cluster.auto.tfvars** and creates cluster with specified name and no of workers
- Reads **job.tf**, **job.auto.tfvars** and creates job, assigns above created cluster to the job.

Verify that the notebook, cluster, and job were created: in the output of the terraform apply command, find the URLs for notebook_url, cluster_url, and job_url, and go to them.

Given the context of the use case, most important file is **data-load-transform.py** where the code with parameterization (*Filename*, *Database name*, *Table name*)

When you invoke the above created job with parameters as per environment, code will be executed.

Conclusion: Terraform code is reproducible to create **notebooks**, **clusters** and **jobs** which can be run with parameters based on environment

GitHub Repository Link:

https://github.com/spappala1/ETLPipeline