# https://medium.com/@datascribeHR/creating-etl-data-pipelines-using-apache-airflow-b5038a740b00

# Creating ETL Data Pipelines using Apache Airflow

# Scenario

You are a data engineer at a data analytics consulting company. You have been assigned to a project that aims to de-congest the national highways by analyzing the road traffic data from different toll plazas. Each highway is operated by a different toll operator with a different IT setup that uses different file formats. Your job is to collect data available in different formats and consolidate it into a single file.

# Objectives

Create a shell script using bash commands to:

* Extract data from a csv file
* Extract data from a tsv file
* Extract data from a fixed width file
* Transform the data
* Load the transformed data into a new csv file

You will then create a DAG to call the shell script.

# Prepare the environment

1. Start Apache Airflow.

Open Apache Airflow in IDE

start\_airflow

2. Open a terminal and create a directory structure for staging area as follows:  
/home/project/airflow/dags/finalassignment/staging.

sudo mkdir -p /home/project/airflow/dags/finalassignment/staging

3. Download the dataset from the source to the destination mentioned below using **wget** command.

**Note: While downloading the file in the terminal use the sudo command before the command used to download the file.**

**Source :** <https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/tolldata.tgz>

**Destination :** /home/project/airflow/dags/finalassignment

Note: If you face any permission issues in writing to the directories, please execute the following commannd

sudo chown -R 100999 /home/project/airflow/dags/finalassignment  
sudo chmod -R g+rw /home/project/airflow/dags/finalassignment  
sudo chown -R 100999 /home/project/airflow/dags/finalassignment/staging  
sudo chmod -R g+rw /home/project/airflow/dags/finalassignment/staging

4. Change to the staging directory.

cd /home/project/airflow/dags/finalassignment/staging

# Create a DAG

**import the libraries**

# import the libraries  
from datetime import timedelta  
# The DAG object; we'll need this to instantiate a DAG  
from airflow.models import DAG  
# Operators; you need this to write tasks!  
from airflow.operators.bash\_operator import BashOperator  
# This makes scheduling easy  
from airflow.utils.dates import days\_ago

## Step 1: Define DAG arguments

#defining DAG arguments  
default\_args = {  
 'owner': 'human',  
 'start\_date': days\_ago(0),  
 'email': ['hello90@gmail.com'],  
 'email\_on\_failure': True,  
 'email\_on\_retry': True,  
 'retries': 1,  
 'retry\_delay': timedelta(minutes=2),  
}

DAG arguments are like settings for the DAG.

The above settings mention

* the owner name,
* when this DAG should run from: days\_age(0) means today,
* the email address where the alerts are sent to,
* the number of retries in case of failure, and
* the time delay between retries.

## Step 2: Define the DAG

Here you are creating a variable named dag by instantiating the DAG class with the following parameters.

ETL\_toll\_data is the ID of the DAG. This is what you see on the web console.

you are passing the dictionary default\_args, in which all the defaults are defined.

description helps us in understanding what this DAG does.

schedule\_interval tells us how frequently this DAG runs. In this case every day. (days=1).

# defining the DAG  
dag = DAG(  
 dag\_id = 'ETL\_toll\_data',  
 default\_args=default\_args,  
 description='Apache Airflow Project',  
 schedule\_interval=timedelta(days=1),  
)

## Step 3: Define the tasks

A task is defined using:

* A task\_id which is a string and helps in identifying the task.
* What bash command it represents.
* Which dag this task belongs to.

**# Task 1- to unzip the data**

# define the tasks  
# Task to unzip data  
  
unzip\_data = BashOperator(  
 task\_id= 'unzip\_data',  
 bash\_command= 'tar -xzvf /home/project/airflow/dags/tolldata.tgz -C /home/project/airflow/dags/finalassignment',  
 dag= dag,  
)

**# Task 2- to extract data from CSV**

This task should extract the fields Rowid, Timestamp, Anonymized Vehicle number, and Vehicle type from the vehicle-data.csv file and save them into a file named csv\_data.csv.

# Task to extract data from CSV  
extract\_data\_from\_csv = BashOperator(  
 task\_id= 'extract\_data\_from\_csv',  
 bash\_command= 'cut -d"," -f1-4 < /home/project/airflow/dags/finalassignment/vehicle-data.csv > /home/project/airflow/dags/finalassignment/csv\_data.csv',  
 dag= dag,  
)

cut: This is the command used to extract sections from each line of files.

<: This is a shell input redirection operator. It tells the shell to read input from the file specified.

>: This is a shell output redirection operator. It tells the shell to write the output of the command to the specified file.

**# Task 3- to extract data from TSV**

This task should extract the fields Number of axles, Tollplaza id, and Tollplaza code from the tollplaza-data.tsv file and save it into a file named tsv\_data.csv.

# Task to extract data from TSV  
  
extract\_data\_from\_tsv = BashOperator(  
 task\_id= 'extract\_data\_from\_tsv',  
 bash\_command= 'cut -f5-7 < /home/project/airflow/dags/finalassignment/tollplaza-data.tsv > /home/project/airflow/dags/finalassignment/tsv\_data.csv',  
 dag= dag,  
)

**# Task 4- to extract data from fixed-width file**

This task should extract the fields Type of Payment code, and Vehicle Code from the fixed width file payment-data.txt and save it into a file named fixed\_width\_data.csv.

# Task to extract data from fixed-width file  
  
extract\_data\_from\_fixed\_file = BashOperator(  
 task\_id= 'extract\_data\_from\_fixed\_file',  
 bash\_command= 'cut -c 59-68 < /home/project/airflow/dags/finalassignment/payment-data.txt > /home/project/airflow/dags/finalassignment/fixed\_width\_data.csv',  
 dag= dag,  
)

-c 59-68: This option specifies that characters from position 59 to 68 (inclusive) should be extracted from each line of the input file. Character positions are 1-based, meaning the first character of each line is position 1.

**# Task to consolidate data into a single CSV file**

This task should create a single csv file named extracted\_data.csv by combining data from the following files:

* csv\_data.csv
* tsv\_data.csv
* fixed\_width\_data.csv

The final csv file should use the fields in the order given below:

Rowid, Timestamp, Anonymized Vehicle number, Vehicle type, Number of axles, Tollplaza id, Tollplaza code, Type of Payment code, and Vehicle Code

# Task to consolidate data into a single CSV file  
  
consolidate\_data = BashOperator(  
 task\_id= 'consolidate\_data',  
 bash\_command= 'paste /home/project/airflow/dags/finalassignment/csv\_data.csv /home/project/airflow/dags/finalassignment/tsv\_data.csv /home/project/airflow/dags/finalassignment/fixed\_width\_data.csv > /home/project/airflow/dags/finalassignment/extracted\_data.csv',  
 dag= dag,  
)

**# Task to transform data**

This task should transform the vehicle\_type field in extracted\_data.csv into capital letters and save it into a file named transformed\_data.csv in the staging directory.

# Task to transform data  
  
Transform\_data = BashOperator(  
 task\_id= 'Transform\_data',  
 #bash\_command= f"awk 'BEGIN{FS=OFS=","} {split($4, a, "\t"); a[1]=toupper(a[1]); $4=a[1]; for(i=2; i<=length(a); i++) $4=$4"\t"a[i]; print $0}' /home/project/airflow/dags/finalassignment/extracted\_data.csv > /home/project/airflow/dags/finalassignment/staging/transformed-data.csv",  
 bash\_command = (  
 "awk 'BEGIN{FS=OFS=\",\"} {split($4, a, \"\\t\"); a[1]=toupper(a[1]); $4=a[1]; "  
 "for(i=2; i<=length(a); i++) $4=$4\"\\t\"a[i]; print $0}' "  
 "/home/project/airflow/dags/finalassignment/extracted\_data.csv > "  
 "/home/project/airflow/dags/finalassignment/staging/transformed-data.csv"  
),  
 dag= dag,  
)

**awk 'BEGIN{FS=OFS=","}**:

* FS=OFS=",": This sets the input and output field separators to commas, which is typical for CSV files.

**{split($4, a, "\t"); a[1]=toupper(a[1]); $4=a[1]; for(i=2; i<=length(a); i++) $4=$4"\t"a[i]; print $0}**:

* split($4, a, "\t"): This splits the 4th column using a tab character as the delimiter and stores the parts in array a.
* a[1]=toupper(a[1]): This converts the first part of the split 4th column to uppercase.
* $4=a[1]; for(i=2; i<=length(a); i++) $4=$4"\t"a[i]: This reconstructs the 4th column by concatenating the uppercased first part and the remaining parts separated by tabs.
* print $0: This prints the modified line.

**# Define the task pipeline**

Task pipeline helps us to organize the order of tasks.

# Set task pipeline  
unzip\_data >> extract\_data\_from\_csv >> extract\_data\_from\_tsv >> extract\_data\_from\_fixed\_file >> consolidate\_data >> Transform\_data

Save the DAG you defined into a file named ETL\_toll\_data.py.

**File Name or DAG ID**: Ensure that the file name (ETL\_toll\_data.py) matches the dag\_id specified in the DAG ('ETL\_toll\_data').

# Submit a DAG

Submitting a DAG is as simple as copying the DAG python file into dags folder in the AIRFLOW\_HOME directory.

Airflow searches for Python source files within the specified DAGS\_FOLDER. The location of DAGS\_FOLDER can be located in the airflow.cfg file, where it has been configured as /home/project/airflow/dags.

Airflow will load the Python source files from this designated location. It will process each file, execute its contents, and subsequently load any DAG objects present in the file.

Therefore, when submitting a DAG, it is essential to position it within this directory structure. Alternatively, the AIRFLOW\_HOME directory, representing the structure /home/project/airflow, can also be utilized for DAG submission.

Open a terminal and run the command below to submit the DAG that was created in the previous exercise.

cp my\_first\_dag.py $AIRFLOW\_HOME/dags

Verify that your DAG actually got submitted.

Run the command below to list out all the existing DAGs.

airflow dags list

Verify that ETL\_toll\_data is a part of the output.

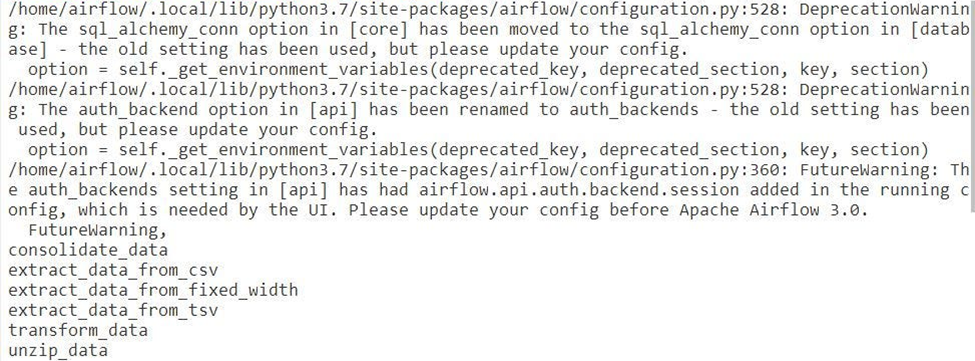
airflow dags list|grep "ETL\_toll\_data"

You should see your DAG name in the output.



Run the command below to list out all the tasks in ETL\_toll\_data.

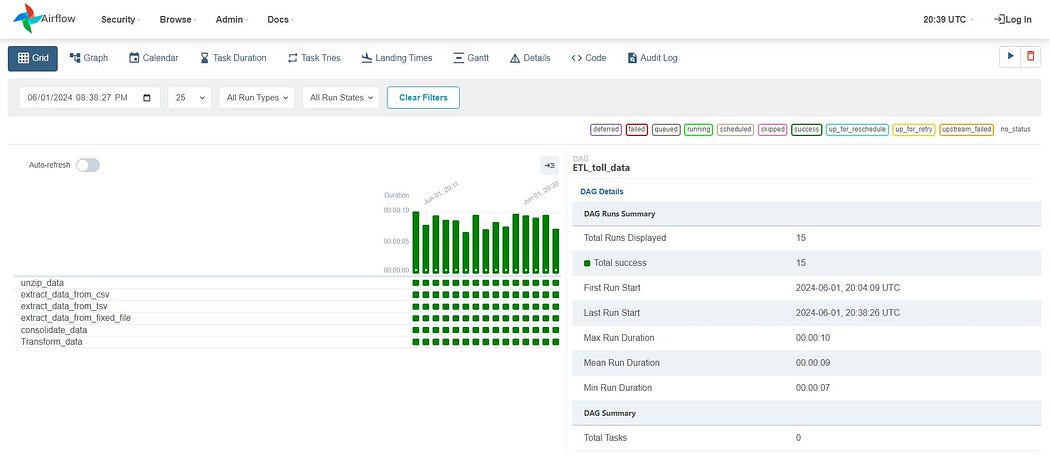
airflow tasks list ETL\_toll\_data

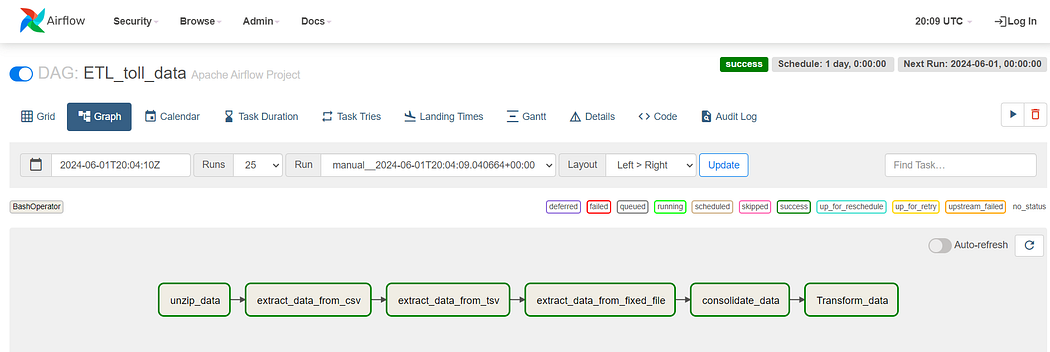


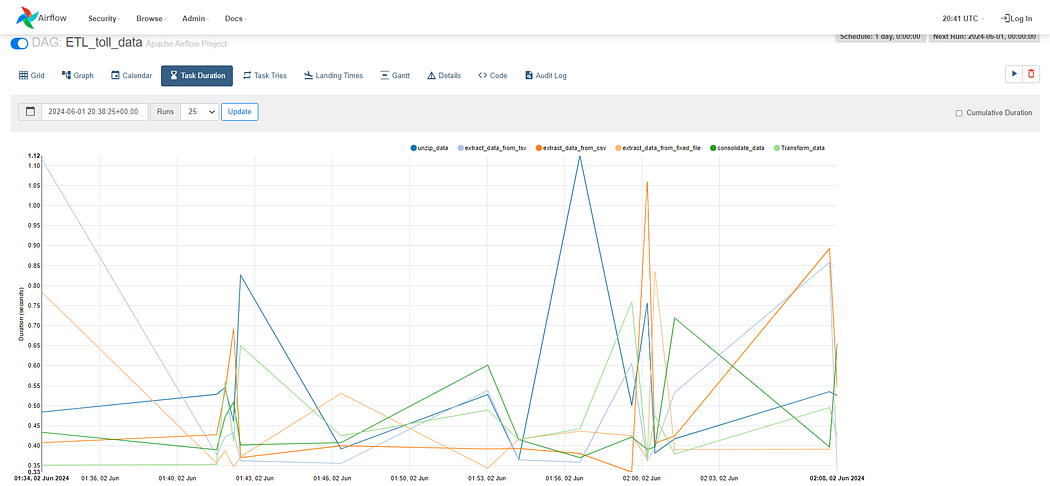
# Unpause/Pause the DAG

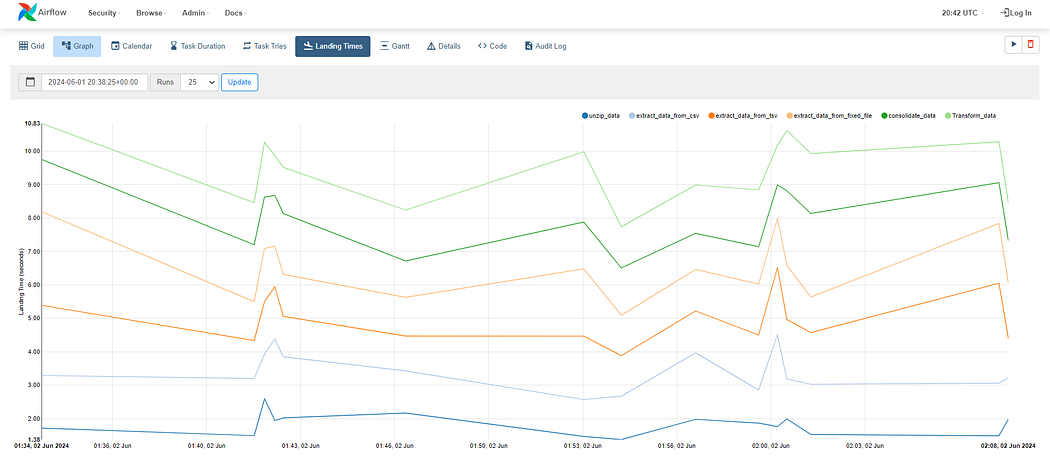
# Pause a DAG  
airflow dags pause ETL\_toll\_data  
  
# Unpause a DAG  
airflow dags unpause ETL\_toll\_data

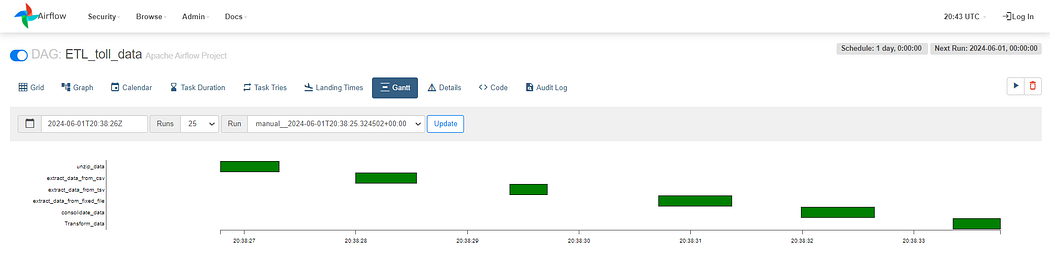
# Monitor the DAG











# Check Errors

If you face issues while submitting the DAG, you can check for errors using the following command in terminal:

/home/project$ airflow dags list-import-errors

# See the Code in Action

Interested in the code behind this article? Check out the implementation on my GitHub repository. Feel free to fork the repo and contribute!