Build ETL Data Pipelines with PythonOperator using Apache Airflow

Estimated time needed: **90** minutes.

Project Scenario

You are a data engineer at a data analytics consulting company. You have been assigned a project 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

In this assignment, you will develop an Apache Airflow DAG that will:

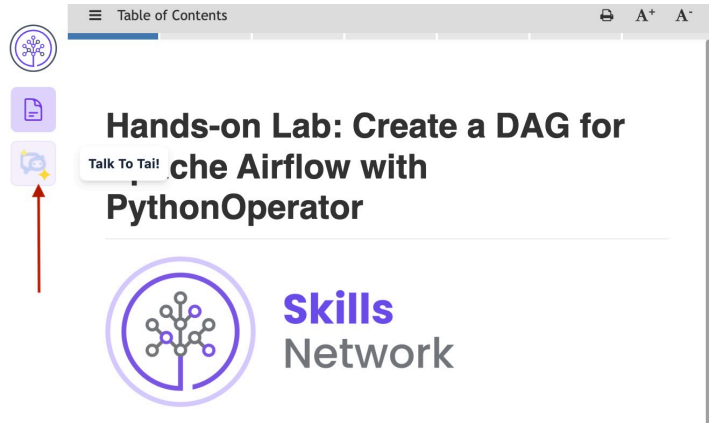
- Extract data from a csv file
- Extract data from a tsv file
- Extract data from a fixed-width file
- Transform the data
- Load the transformed data into the staging area

About Skills Network Cloud IDE

Skills Network Cloud IDE (based on Theia and Docker) provides an environment for hands-on labs for course and project-related labs. Theia is an open-source IDE (Integrated Development Environment) that can be run on a desktop or on the cloud. To complete this lab, you will be using the Cloud IDE based on Theia, running in a Docker container.

Important notice about this lab environment

Please be aware that sessions for this lab environment are not persistent. A new environment is created for you every time you connect to this lab. Any data you may have saved in an earlier session will get lost. To avoid losing your data, please plan to complete these labs in a single session. You can use the `Tai` AI assistant to complete this task.



Exercise 1: Prepare the lab environment

1. Start Apache Airflow.

Open Apache Airflow in IDE

   Please wait until Airflow starts up fully and is active before you proceed further. If there is an error starting Airflow, please restart it.

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

```
1
sudo mkdir -p /home/project/airflow/dags/python_etl/staging
```

Copied! Executed!

3. Execute the following commands to avoid any permission issues in writing to the directories.

```
1
sudo chmod -R 777 /home/project/airflow/dags/python_etl
```

Copied! Executed!

Exercise 2: Add imports, define DAG arguments, and define DAG

1. Create a file named `ETL_toll_data.py` in `/home/project` directory and add the necessary imports and DAG arguments to it.

Parameter	Value
owner	<You may use any dummy name>
start_date	today
email	<You may use any dummy email>
retries	1
retry_delay	5 minutes

2. Create a DAG as per the following details.

Parameter	Value
DAG id	`ETL_toll_data`
Schedule	Daily once
default_args	as you have defined in the previous step
description	Apache Airflow Final Assignment

Exercise 3: Create Python functions

1. Create a Python function named `download_dataset` to download the data set from the source to the destination. You will call this function from the task.

   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/python_etl/staging`

2. Create a Python function named `untar_dataset` to untar the downloaded data set.

3. Create a function named `extract_data_from_csv` to 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`.

4. Create a function named `extract_data_from_tsv` to 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`.

5. Create a function named `extract_data_from_fixed_width`to 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`.

6. Create a function named `consolidate_data` to 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`

7. Create a function named `transform_data` to 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.

Exercise 4: Create a tasks using PythonOperators and define pipeline

1. Create 7 tasks using Python operators that does the following using the Python functions created in Task 2.

   1. download_dataset
   2. untar_dataset
   3. extract_data_from_csv
   4. extract_data_from_tsv
   5. extract_data_from_fixed_width
   6. consolidate_data
   7. transform_data

2. Define the task pipeline based on the details given below:

Task	Functionality
First task	`download_data`
Second task	`unzip_data`
Third task	`extract_data_from_csv`

Task	Functionality
Fourth task	`extract_data_from_tsv`
Fifth task	`extract_data_from_fixed_width`
Sixth task	`consolidate_data`
Seventh task	`transform_data`

Exercise 5: Save, submit, and run DAG

1. Save the DAG you defined.
2. Submit the DAG by copying it into `$AIRFLOW_HOME/dags` directory.

▶ Click here if your DAG does not get submitted properly.

3. Use CLI or Web UI to unpause the task.
4. Observe the outcome of the tasks in DAG on the Airflow console.

Solution

▼ Click here for the solution

```
 1. 1
 2. 2
 3. 3
 4. 4
 5. 5
 6. 6
 7. 7
 8. 8
 9. 9
10. 10
11. 11
12. 12
13. 13
14. 14
15. 15
16. 16
17. 17
18. 18
19. 19
20. 20
21. 21
22. 22
23. 23
24. 24
25. 25
26. 26
27. 27
28. 28
29. 29
30. 30
31. 31
32. 32
33. 33
34. 34
35. 35
36. 36
37. 37
38. 38
39. 39
40. 40
41. 41
42. 42
43. 43
44. 44
45. 45
46. 46
47. 47
48. 48
49. 49
50. 50
51. 51
52. 52
53. 53
54. 54
55. 55
56. 56
57. 57
58. 58
59. 59
60. 60
61. 61
62. 62
63. 63
64. 64
65. 65
66. 66
67. 67
68. 68
69. 69
70. 70
```

```
71. 71
72. 72
73. 73
74. 74
75. 75
76. 76
77. 77
78. 78
79. 79
80. 80
81. 81
82. 82
83. 83
84. 84
85. 85
86. 86
87. 87
88. 88
89. 89
90. 90
91. 91
92. 92
93. 93
94. 94
95. 95
96. 96
97. 97
98. 98
99. 99
100. 100
101. 101
102. 102
103. 103
104. 104
105. 105
106. 106
107. 107
108. 108
109. 109
110. 110
111. 111
112. 112
113. 113
114. 114
115. 115
116. 116
117. 117
118. 118
119. 119
120. 120
121. 121
122. 122
123. 123
124. 124
125. 125
126. 126
127. 127
128. 128
129. 129
130. 130
131. 131
132. 132
133. 133
134. 134
135. 135
136. 136
137. 137
138. 138
139. 139
140. 140
141. 141
142. 142
143. 143
144. 144
145. 145
146. 146
147. 147
148. 148
149. 149
150. 150
151. 151
152. 152
153. 153
154. 154
155. 155
156. 156
157. 157
```

```
1. from datetime import timedelta
2. from airflow.models import DAG
3. from airflow.operators.python import PythonOperator
4. from airflow.utils.dates import days_ago
5. import requests
6. import tarfile
7. import csv
8. import shutil
9.
10. # Define the path for the input and output files
11. source_url = 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/
12. destination_path = '/home/project/airflow/dags/python_etl/staging'
13.
14. # Function to download the dataset
15. def download_dataset():
16.     response = requests.get(source_url, stream=True)
```

```
    if response.status_code == 200:
        with open(f"{destination_path}/tolldata.tgz", 'wb') as f:
            f.write(response.raw.read())
    else:
        print("Failed to download the file")

# Function to untar the dataset
def untar_dataset():
    with tarfile.open(f"{destination_path}/tolldata.tgz", "r:gz") as tar:
        tar.extractall(path=destination_path)

# Function to extract data from CSV
def extract_data_from_csv():
    input_file = f"{destination_path}/vehicle-data.csv"
    output_file = f"{destination_path}/csv_data.csv"
    with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:

        writer = csv.writer(outfile)
        writer.writerow(['Rowid', 'Timestamp', 'Anonymized Vehicle number', 'Vehicle type'])
        for line in infile:
            row = line.split(',')
            writer.writerow([row[0], row[1], row[2], row[3]])

# Function to extract data from TSV
def extract_data_from_tsv():
    input_file = f"{destination_path}/tollplaza-data.tsv"
    output_file = f"{destination_path}/tsv_data.csv"
    with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
        writer = csv.writer(outfile)
        writer.writerow(['Number of axles', 'Tollplaza id', 'Tollplaza code'])
        for line in infile:
            row = line.split('\t')
            writer.writerow([row[0], row[1], row[2]])

# Function to extract data from fixed width file
def extract_data_from_fixed_width():
    input_file = f"{destination_path}/payment-data.txt"
    output_file = f"{destination_path}/fixed_width_data.csv"
    with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
        writer = csv.writer(outfile)
        writer.writerow(['Type of Payment code', 'Vehicle Code'])
        for line in infile:
            writer.writerow([line[0:6].strip(), line[6:12].strip()])

# Function to consolidate data
def consolidate_data():
    csv_file = f"{destination_path}/csv_data.csv"
    tsv_file = f"{destination_path}/tsv_data.csv"
    fixed_width_file = f"{destination_path}/fixed_width_data.csv"
    output_file = f"{destination_path}/extracted_data.csv"

    with open(csv_file, 'r') as csv_in, open(tsv_file, 'r') as tsv_in, open(fixed_width_file, 'r') as fixed_in, open(output_file, 'w
        csv_reader = csv.reader(csv_in)
        tsv_reader = csv.reader(tsv_in)
        fixed_reader = csv.reader(fixed_in)
        writer = csv.writer(out_file)

        writer.writerow(['Rowid', 'Timestamp', 'Anonymized Vehicle number', 'Vehicle type', 'Number of axles', 'Tollplaza id', 'Toll
        next(csv_reader)
        next(tsv_reader)
        next(fixed_reader)

        for csv_row, tsv_row, fixed_row in zip(csv_reader, tsv_reader, fixed_reader):
            writer.writerow(csv_row + tsv_row + fixed_row)

# Function to transform data
def transform_data():
    input_file = f"{destination_path}/extracted_data.csv"
    output_file = f"{destination_path}/transformed_data.csv"

    with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
        reader = csv.DictReader(infile)
        writer = csv.DictWriter(outfile, fieldnames=reader.fieldnames)
        writer.writeheader()

        for row in reader:
            row['Vehicle type'] = row['Vehicle type'].upper()
            writer.writerow(row)

# Default arguments for the DAG
default_args = {
    'owner': 'Your name',
    'start_date': days_ago(0),
    'email': ['your email'],
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

# Define the DAG
dag = DAG(
    'ETL_toll_data',
    default_args=default_args,
    description='Apache Airflow Final Assignment',
    schedule_interval=timedelta(days=1),
)

# Define the tasks
download_task = PythonOperator(
    task_id='download_dataset',
    python_callable=download_dataset,
    dag=dag,
)

untar_task = PythonOperator(
```

```
    task_id='untar_dataset',
    python_callable=untar_dataset,
    dag=dag,
)

extract_csv_task = PythonOperator(
    task_id='extract_data_from_csv',
    python_callable=extract_data_from_csv,
    dag=dag,
)

extract_tsv_task = PythonOperator(
    task_id='extract_data_from_tsv',
    python_callable=extract_data_from_tsv,
    dag=dag,
)

extract_fixed_width_task = PythonOperator(
    task_id='extract_data_from_fixed_width',
    python_callable=extract_data_from_fixed_width,
    dag=dag,
)

consolidate_task = PythonOperator(
    task_id='consolidate_data',
    python_callable=consolidate_data,
    dag=dag,
)

transform_task = PythonOperator(
    task_id='transform_data',
    python_callable=transform_data,
    dag=dag,
)

# Set the task dependencies
download_task >> untar_task >> [extract_csv_task, extract_tsv_task, extract_fixed_width_task] >> consolidate_task >> transform_task
```

Copied!

Authors

Lavanya T S
Ramesh Sannareddy

Other Contributors

Rav Ahuja