

# Final Assignment (Part 1) - Creating ETL Data Pipelines using Apache Airflow



Estimated time needed: **90** minutes.

## About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and Apache Airflow and MySQL database running in a Docker container. You will also need an instance of DB2 running in IBM Cloud.

## 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.

## 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

In this assignment you will author 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

## Note - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. These screenshots will need to be uploaded for peer review in the next section of the course. You can use various free screengrabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots.

## Exercise 1 - Prepare the lab environment

Before you start the assignment:

- Start Apache Airflow.
- Download the dataset from the source to the destination mentioned below.

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

- Create a directory for staging area.

```
mkdir /home/project/airflow/dags/finalassignment/staging
```

## Exercise 2 - Create a DAG

### Task 1.1 - Define DAG arguments

Define the DAG arguments as per the following details:

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

Take a screenshot of the task code.

Name the screenshot `dag_args.jpg`. (Images can be saved with either the .jpg or .png extension.)

### Task 1.2 - Define the DAG

Create a DAG as per the following details.

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

Take a screenshot of the command you used and the output.

Name the screenshot `dag_definition.jpg`. (Images can be saved with either the .jpg or .png extension.)

### Task 1.3 - Create a task to unzip data

Create a task named `unzip_data`.

Use the downloaded data from the url given in the first part of this assignment in exercise 1 and uncompress it into the destination directory.

Take a screenshot of the task code.

Name the screenshot `unzip_data.jpg`. (Images can be saved with either the .jpg or .png extension.)

Read through the file `fileformats.txt` to understand the column details.

## Task 1.4 - Create a task to extract data from csv file

Create a task named `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`.

Take a screenshot of the task code.

Name the screenshot `extract_data_from_csv.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.5 - Create a task to extract data from tsv file

Create a task named `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`.

Take a screenshot of the task code.

Name the screenshot `extract_data_from_tsv.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.6 - Create a task to extract data from fixed width file

Create a task named `extract_data_from_fixed_width`.

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`.

Take a screenshot of the task code.

Name the screenshot `extract_data_from_fixed_width.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.7 - Create a task to consolidate data extracted from previous tasks

Create a task named `consolidate_data`.

This task should create a single csv file named `extracted_data.csv` by combining data from

- `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`

Hint: Use the bash `paste` command.

`paste` command merges lines of files.

Example : `paste file1 file2 > newfile`

The above command merges the columns of the files file1 and file2 and sends the output to newfile.

You can use the command `man paste` to explore more.

Take a screenshot of the command you used and the output.

Name the screenshot `consolidate_data.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.8 - Transform and load the data

Create a task named `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.

Take a screenshot of the command you used and the output.

Name the screenshot `transform.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.9 - Define the task pipeline

Define the task pipeline as per the details given below:

Task	Functionality
First task	<code>unzip_data</code>
Second task	<code>extract_data_from_csv</code>
Third task	<code>extract_data_from_tsv</code>
Fourth task	<code>extract_data_from_fixed_width</code>
Fifth task	<code>consolidate_data</code>
Sixth task	<code>transform_data</code>

Take a screenshot of the task pipeline section of the DAG.

Name the screenshot `task_pipeline.jpg`. (Images can be saved with either the .jpg or .png extension.)

# Exercise 3 - Getting the DAG operational.

Save the DAG you defined into a file named `ETL_toll_data.py`.

## Task 1.10 - Submit the DAG

Take a screenshot of the command you used and the output.

Name the screenshot `submit_dag.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.11 - Unpause the DAG

Take a screenshot of the command you used and the output.

Name the screenshot `unpause_dag.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 1.12 - Monitor the DAG

Take a screenshot of the DAG runs for the Airflow console.

Name the screenshot `dag_runs.jpg`. (Images can be saved with either the .jpg or .png extension.)

This concludes the assignment.

## Authors

Ramesh Sannareddy

## Other Contributors

Rav Ahuja

# Final Assignment (Part 2) - Creating Streaming Data Pipelines using Kafka



Estimated time needed: **45** minutes.

## About This SN Labs Cloud IDE

This Skills Network Labs Cloud IDE provides a hands-on environment for course and project related labs. It utilizes Theia, an open-source IDE (Integrated Development Environment) platform, that can be run on desktop or on the cloud. To complete this lab, we will be using the Cloud IDE based on Theia and Kafka and MySQL database running in a Docker container. You will also need an instance of DB2 running in IBM Cloud.

## 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 the earlier session will get lost. Please plan to complete these labs in a single session to avoid losing your data.

## 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. As a vehicle passes a toll plaza, the vehicle's data like `vehicle_id,vehicle_type,toll_plaza_id` and timestamp are streamed to Kafka. Your job is to create a data pipe line that collects the streaming data and loads it into a database.

## Objectives

In this assignment you will create a streaming data pipe by performing these steps:

- Start a MySQL Database server.
- Create a table to hold the toll data.
- Start the Kafka server.
- Install the Kafka python driver.
- Install the MySQL python driver.
- Create a topic named toll in kafka.
- Download streaming data generator program.
- Customize the generator program to steam to toll topic.
- Download and customise streaming data consumer.
- Customize the consumer program to write into a MySQL database table.
- Verify that streamed data is being collected in the database table.

## Note - Screenshots

Throughout this lab you will be prompted to take screenshots and save them on your own device. These screenshots will need to be uploaded for peer review in the next section of the course. You can use various free screengrabbing tools or your operating system's shortcut keys (Alt + PrintScreen in Windows, for example) to capture the required screenshots.

## Exercise 1 - Prepare the lab environment

Before you start the assignment, complete the following steps to set up the lab:

- Step 1: Download Kafka.

```
wget https://archive.apache.org/dist/kafka/2.8.0/kafka_2.12-2.8.0.tgz
```

- Step 2: Extract Kafka.

```
tar -xzf kafka_2.12-2.8.0.tgz
```

- Step 3: Start MySQL server.

```
start_mysql
```

- Step 4: Connect to the mysql server, using the command below. Make sure you use the password given to you when the MySQL server starts. Please make a note or record of the password because you will need it later.

```
mysql --host=127.0.0.1 --port=3306 --user=root --password=Mjk0NDQtcnNhbm5h
```

- Step 5: Create a database named `tolldata`.

At the 'mysql>' prompt, run the command below to create the database.

```
create database tolldata;
```

- Step 6: Create a table named `livetolldata` with the schema to store the data generated by the traffic simulator.

Run the following command to create the table:

```
use tolldata;

create table livetolldata(timestamp datetime,vehicle_id int,vehicle_type char(15),toll_plaza_id smallint);
```

This is the table where you would store all the streamed data that comes from kafka. Each row is a record of when a vehicle has passed through a certain toll plaza along with its type and anonymized id.

- Step 7: Disconnect from MySQL server.

```
exit
```

- Step 8: Install the python module `kafka-python` using the pip3 command.

```
pip3 install kafka-python
```

This python module will help you to communicate with kafka server. It can used to send and receive messages from kafka.

- Step 8: Install the python module `mysql-connector-python` using the pip3 command.

```
pip3 install mysql-connector-python
```

This python module will help you to interact with mysql server.

## Exercise 2 - Start Kafka

### Task 2.1 - Start Zookeeper

Start zookeeper server.

Take a screenshot of the command you run.

Name the screenshot `start_zookeeper.jpg`. (Images can be saved with either the .jpg or .png extension.)

### Task 2.2 - Start Kafka server

Start Kafka server

Take a screenshot of the command you run.

Name the screenshot `start_kafka.jpg`. (Images can be saved with either the .jpg or .png extension.)



## Task 2.3 - Create a topic named `toll`

Create a Kafka topic named `toll`

Take a screenshot of the command you run.

Name the screenshot `create_toll_topic.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 2.4 - Download the Toll Traffic Simulator

Download the `toll_traffic_generator.py` from the url given below using 'wget'.

`https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/toll_traffic_generator.py`

Open the code using the theia editor using the "Menu --> File --> Open" option.

Take a screenshot of the task code.

Name the screenshot `download_simulator.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 2.5 - Configure the Toll Traffic Simulator

Open the `toll_traffic_generator.py` and set the topic to `toll`.

Take a screenshot of the task code with the topic clearly visible.

Name the screenshot `configure_simulator.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 2.6 - Run the Toll Traffic Simulator

Run the `toll_traffic_generator.py`.

Hint : `python3 <pythonfilename>` runs a python program on the theia lab.

Take a screenshot of the output of the simulator.

Name the screenshot `simulator_output.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 2.7 - Configure `streaming_data_reader.py`

Download the `streaming_data_reader.py` from the url below using 'wget'.

`https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/streaming_data_reader.py`

Open the `streaming_data_reader.py` and modify the following details so that the program can connect to your mysql server.

`TOPIC`

`DATABASE`

`USERNAME`

`PASSWORD`

Take a screenshot of the code you modified.

Name the screenshot `streaming_reader_code.jpg`. (Images can be saved with either the .jpg or .png extension.)

## Task 2.8 - Run `streaming_data_reader.py`

Run the `streaming_data_reader.py`

Take a screenshot of the output of the `streaming_data_reader.py`.

Name the screenshot `data_reader_output.jpg`. (Images can be saved with either the .jpg or .png extension.)

```
python3 streaming_data_reader.py
```

## Task 2.9 - Health check of the streaming data pipeline.

If you have done all the steps till here correctly, the streaming toll data would get stored in the table `livetolldata`.

List the top 10 rows in the table `livetolldata`.

Take a screenshot of the command and the output.

Name the screenshot `output_rows.jpg`. (Images can be saved with either the .jpg or .png extension.)

This concludes the assignment.

## Authors

Ramesh Sannareddy

## Other Contributors

Rav Ahuja

## Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-08-16	0.1	Ramesh Sannareddy	Created initial version

2021-12-01	0.2	Jeff Grossman	Added copy code blocks
------------	-----	---------------	------------------------

Copyright (c) 2021 IBM Corporation. All rights reserved.



## Task 1.1 - Define DAG arguments

```
1  default_args = {
2      'owner': 'Cristiano Ronaldo',
3      'start_date': days_ago(0),
4      'email': ['cr7@somemail.com'],
5      'email_on_failure': True,
6      'email_on_retry': True,
7      'retries': 1,
8      'retry_delay': timedelta(minutes=5),
9  }
```

## Task 1.2 - Define the DAG

```
# defining the DAG

dag = DAG(
    dag_id='ETL_toll_data',
    default_args=default_args,
    description='Apache Airflow Final Assignment',
    schedule_interval=timedelta(days=1),
)
```

## Task 1.3 - Create a task to unzip data

```
# task 1 unzipdata
unzip_data = BashOperator(
    task_id='unzip_data',
    bash_command=f"tar -xvzf /home/project/airflow/dags/tolldata.tgz -C /home/project/airflow/dags",
    dag=dag,
)
```

## Task 1.4 - Create a task to extract data from csv file

```
# task 2 extract data from csv

extract_data_from_csv = BashOperator(
    task_id='extract_data_from_csv',
    bash_command='cut -d"," -f1,2,3,4 /home/project/airflow/dags vehicle-data.csv > csv_data.csv',
    dag=dag,
)
```

## Task 1.5 - Create a task to extract data from tsv file

```
# task 3 extract data from TSV

extract_data_from_tsv = BashOperator(
    task_id='extract_data_from_tsv',
    bash_command='cut -d" " -f5,6,7 /home/project/airflow/dags tollplaza-data.tsv > tsv_data.csv',
    dag=dag,
)
```

## Task 1.6 - Create a task to extract data from fixed width file

```
# task 4 extract from fixed_width

extract_data_from_fixed_width = BashOperator(
    task_id='extract_data_from_fixed_width',
    bash_command='cut -b 59-62,63-68 /home/project/airflow/dags payment-data.txt > fixed_width_data.csv',
    dag=dag,
)
```

## Task 1.7 - Create a task to consolidate data extracted from previous tasks

```
# task 5 consolidate

consolidate_data = BashOperator(
    task_id='consolidate_data',
    bash_command=f"paste path/csv_data.csv path/tsv_data.csv path/fixed_width_data.csv >path/extracted_data.csv",
    dag=dag,
```

## Task 1.8 - Transform and load the data

```
# define the task 'transform_data'
transform_data = BashOperator(
    task_id='transform_data',
    bash_command='tr "[a-z]" "[A-Z]" < extracted_data_csv > transformed_data.csv',
    dag=dag,
```

## Task 1.9 - Define the task pipeline

```
# task pipeline
unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_from_fixed_width >> consolidate_data >> transform_data
```

## Task 1.10 - Submit the DAG

```
theia@theiadocker-joaocosta1:/home/project$ cp ETL_toll_data.py $AIRFLOW_HOME/dags
theia@theiadocker-joaocosta1:/home/project$
```

## Task 1.11 - Unpause the DAG

```
theia@theiadocker-joaocosta1:/home/project$ airflow dags unpause ETL_toll_data
Dag: ETL_toll_data, paused: False
```

## Task 2.1 - Start Zookeeper

```
theia@theiadocker-joacosta1:/home/project$ cd kafka_2.12-2.8.0
theia@theiadocker-joacosta1:/home/project/kafka_2.12-2.8.0$ bin/zookeeper-server-start.sh config/zoo
keeper.properties
```

## Task 2.2 - Start Kafka server

```
theia@theiadocker-joacosta1:/home/project$ cd kafka_2.12-2.8.0
theia@theiadocker-joacosta1:/home/project/kafka_2.12-2.8.0$ bin/kafka-server-start.sh config/server.
properties
```


## Task 2.3 - Create a topic named toll

```
theia@theiadocker-joacosta1:/home/project$ cd kafka_2.12-2.8.0
theia@theiadocker-joacosta1:/home/project/kafka_2.12-2.8.0$ bin/kafka-topics.sh --create --topic toll
--bootstrap-server localhost:9092
Created topic toll.
theia@theiadocker-joacosta1:/home/project/kafka_2.12-2.8.0$
```

## Task 2.4 - Download the Toll Traffic Simulator

```
toll_traffic_generator.py ×
kafka_2.12-2.8.0 > toll_traffic_generator.py > ...
1  """
2  Top Traffic Simulator
3  """
4  from time import sleep, time, ctime
5  from random import random, randint, choice
6  from kafka import KafkaProducer
7  producer = KafkaProducer(bootstrap_servers='localhost:9092')
8
9  TOPIC = 'set your topic here'
10
11  VEHICLE_TYPES = ("car", "car", "car", "car", "car", "car", "car", "car",
12                  "car", "car", "car", "truck", "truck", "truck",
13                  "truck", "van", "van")
14  for _ in range(100000):
15      vehicle_id = randint(10000, 10000000)
16      vehicle_type = choice(VEHICLE_TYPES)
17      now = ctime(time())
18      plaza_id = randint(4000, 4010)
19      message = f"{now},{vehicle_id},{vehicle_type},{plaza_id}"
20      message = bytearray(message.encode("utf-8"))
21      print(f"A {vehicle_type} has passed by the toll plaza {plaza_id} at {now}.")
theia@theiadocker-joacosta1: /home/project/kafka_2.12-2.8.0
theia@theiadocker-joacosta1: /home/project/kafka_2.12-2.8.0 ×
```

## Task 2.5 - Configure the Toll Traffic Simulator

```
kafka_2.12-2.8.0 >  toll_traffic_generator.py > ...
1  """
2  Top Traffic Simulator
3  """
4  from time import sleep, time, ctime
5  from random import random, randint, choice
6  from kafka import KafkaProducer
7  producer = KafkaProducer(bootstrap_servers='localhost:9092')
8
9  TOPIC = 'toll'
10
11  VEHICLE_TYPES = ("car", "car", "car", "car", "car", "car", "car", "car",
12                  "car", "car", "car", "truck", "truck", "truck",
13                  "truck", "van", "van")
14  for _ in range(100000):
15      vehicle_id = randint(10000, 10000000)
16      vehicle_type = choice(VEHICLE_TYPES)
17      now = ctime(time())
18      plaza_id = randint(4000, 4010)
19      message = f"{now},{vehicle_id},{vehicle_type},{plaza_id}"
20      message = bytearray(message.encode("utf-8"))
```

## Task 2.6 - Run the Toll Traffic Simulator

```
_2.12-2.8.0/toll_traffic_generator.pyobject$ /usr/bin/python3 /home/project/kafka_
A car has passed by the toll plaza 4008 at Tue Feb 15 13:54:08 2022.
A van has passed by the toll plaza 4003 at Tue Feb 15 13:54:10 2022.
A car has passed by the toll plaza 4006 at Tue Feb 15 13:54:11 2022.
A car has passed by the toll plaza 4001 at Tue Feb 15 13:54:13 2022.
A truck has passed by the toll plaza 4006 at Tue Feb 15 13:54:15 2022.
A van has passed by the toll plaza 4008 at Tue Feb 15 13:54:16 2022.
A van has passed by the toll plaza 4008 at Tue Feb 15 13:54:18 2022.
A car has passed by the toll plaza 4007 at Tue Feb 15 13:54:19 2022.
A car has passed by the toll plaza 4002 at Tue Feb 15 13:54:20 2022.
A car has passed by the toll plaza 4002 at Tue Feb 15 13:54:21 2022.
A car has passed by the toll plaza 4006 at Tue Feb 15 13:54:22 2022.
A car has passed by the toll plaza 4002 at Tue Feb 15 13:54:23 2022.
A truck has passed by the toll plaza 4006 at Tue Feb 15 13:54:24 2022.
A car has passed by the toll plaza 4008 at Tue Feb 15 13:54:26 2022.
A car has passed by the toll plaza 4001 at Tue Feb 15 13:54:27 2022.
A car has passed by the toll plaza 4008 at Tue Feb 15 13:54:29 2022.
A car has passed by the toll plaza 4010 at Tue Feb 15 13:54:29 2022.
```

## Task 2.7 - Configure `streaming_data_reader.py`

```
ming_data_reader.pyoacosta1:/home/project$ /usr/bin/python3 /home/project/stream
Connecting to the database
Connected to database
Connecting to Kafka
Connected to Kafka
Reading messages from the topic toll
```

## Task 2.8 - Run `streaming_data_reader.py`

```
ming_data_reader.pyoacosta1:/home/project$ /usr/bin/python3 /home/project/stream
Connecting to the database
Connected to database
Connecting to Kafka
Connected to Kafka
Reading messages from the topic toll
```

## Task 2.9 - Health check of the streaming data pipeline.

```
Database changed
mysql> select * from livetolldata LIMIT 10;
```

timestamp	vehicle_id	vehicle_type	toll_plaza_id
2022-02-15 14:12:57	8558911	car	4000
2022-02-15 14:12:57	1551082	car	4001
2022-02-15 14:12:57	9688853	car	4009
2022-02-15 14:12:58	6083014	car	4010
2022-02-15 14:13:00	4699061	car	4001
2022-02-15 14:13:01	7026362	car	4010
2022-02-15 14:13:02	429999	car	4010
2022-02-15 14:13:04	6188869	car	4008
2022-02-15 14:13:06	5955480	truck	4009
2022-02-15 14:13:07	5387282	car	4007

```
10 rows in set (0.00 sec)
```