

Apache Airflow Final Project Report — ETL Toll Data Pipeline

1. Introduction

In today's data-driven world, efficient data extraction, transformation, and loading (ETL) processes are crucial for analytics and decision-making. This project demonstrates the use of **Apache Airflow** — a powerful workflow orchestration tool — to automate and schedule ETL tasks.

The objective is to process toll data from various file formats (CSV, TSV, and fixed-width text) and consolidate it into a single transformed dataset that can be used for traffic pattern analysis.

By completing this project, I gained **hands-on experience in workflow automation, task dependencies, data transformation, and Airflow DAG creation.**

2. Skills Learned

Through this project, I developed the following skills:

- **Apache Airflow DAG Design:** Creating and managing Directed Acyclic Graphs for task automation
- **Bash Scripting in Airflow:** Using BashOperator to execute Linux commands for ETL steps
- **ETL Workflow Automation:** Automating extraction, transformation, and loading processes
- **Data Manipulation in Linux:** Handling CSV, TSV, and fixed-width text files using shell commands
- **Dependency Management:** Establishing relationships between sequential Airflow tasks
- **Error Handling and Retries:** Configuring retries and alerts for robust pipeline execution
- **Airflow UI Monitoring:** Monitoring DAG runs, logs, and task progress visually

3. Objectives

This project aims to develop 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

Task 1: Set up the lab environment

- Start Apache Airflow.
- Open a terminal and create a directory structure for the staging area as follows:

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

- Execute the following commands to give appropriate permission to the directories.

```
sudo chmod -R 777 /home/project/airflow/dags/finalassignment
```

- Download the data set from the source to the following destination using the `curl` command.

```
sudo curl https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DB0250EN-SkillsNetwork/labs/Final%20Assignment/tolldata.tgz -o /home/project/airflow/dags/finalassignment/tolldata.tgz
```

Task 2: Create imports, DAG argument, and definition

- Create a new file named ETL_toll_data.py in /home/project directory and open it in the file editor.
- Import all the necessary packages to build the DAG.
- Define the DAG arguments as per the following details in the ETL_toll_data.py file:

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

The screenshot displays the Skills Network Labs interface. On the left, a sidebar contains navigation icons for Lab guide, Ask Tai, Inbox, What's new, Support, and Reset lab. The main content area shows the task instructions for Step 3 and Step 4. Step 3 instructs to define DAG arguments in the ETL_toll_data.py file, with a table listing the parameters and their values. Step 4 instructs to define the DAG in the same file. Below the instructions, a terminal window shows the execution of commands to create the file and list its contents. The code editor on the right shows the Python code for ETL_toll_data.py, which imports datetime and timedelta from the datetime module, imports DAG and BashOperator from the airflow module, and defines the default_args dictionary with the same values as in the table.

3. Define the DAG arguments as per the following details in the `ETL_toll_data.py` file:

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`.

4. Define the DAG in the `ETL_toll_data.py` file using 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

```
1 from datetime import datetime, timedelta
2 from airflow import DAG
3 from airflow.operators.bash import BashOperator
4
5 default_args = {
6     'owner': 'sangeeta K',
7     'start_date': datetime.now(),
8     'email': ['srk12@gmail.com'],
9     'email_on_failure': True,
10    'email_on_retry': True,
11    'retries': 1,
12    'retry_delay': timedelta(minutes=5),
13 }
14
```

theia@theiadocker-sangeetaramd:/home/project X

```
.tgz
theia@theiadocker-sangeetaramd:/home/projects$ sudo touch ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/projects$ ls -l /home/project/ETL_toll_data.py
-rw-r--r-- 1 theia users 708 Oct 13 23:19 /home/project/ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/projects$
```

- Define the DAG in the `ETL_toll_data.py` file using 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

The screenshot displays a Skills Network Labs environment. On the left, a sidebar contains navigation icons for Lab guide, Ask, Inbox, What's new, Support, and Reset lab. The main content area shows a task code editor with a table of parameters and a code editor for `ETL_toll_data.py`.

Parameter Table:

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

Code Editor (ETL_toll_data.py):

```

1 from datetime import datetime, timedelta
2 from airflow import DAG
3 from airflow.operators.bash import BashOperator
4
5
6 default_args = {
7     'owner': 'sangeeta K',
8     'start_date': datetime.now(),
9     'email': ['srk12@gmail.com'],
10    'email_on_failure': True,
11    'email_on_retry': True,
12    'retries': 1,
13    'retry_delay': timedelta(minutes=5),
14}
15
16 with DAG(
17     dag_id='ETL_toll_data',
18     default_args=default_args,
19     description='Apache Airflow Final Assignment',
20     schedule_interval='@daily',
21 ) as dag:
22     pass
  
```

Terminal Window:

```

theia@theiadocker-sangeetaramd:/home/project $
theia@theiadocker-sangeetaramd:/home/project $ sudo touch ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/project $ ls -l /home/project/ETL_toll_data.py
-rw-r--r-- 1 theia users 788 Oct 13 23:16 /home/project/ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/project $
  
```

Instructions:

- Take a screenshot of the task code. Name the screenshot `dag_args.jpg`.
- Define the DAG in the `ETL_toll_data.py` file using the following details.
- Take a screenshot of the command and output you used. Name the screenshot `dag_definition.jpg`.

Final Screenshots:

At the end of this exercise, you should have the following screenshots with `.jpg` or `.png` extension:

- `dag_args.jpg`
- `dag_definition.jpg`

Task 3: Create the tasks using BashOperator

- Create a task named `unzip_data` to unzip data. Use the data downloaded in the first part of this assignment in Set up the lab environment and uncompress it into the destination directory using `tar`.

Exercise 3: Create the tasks using BashOperator

1. Create a task named `unzip_data` to unzip data. Use the data downloaded in the first part of this assignment in `Set up the lab environment` and uncompress it into the destination directory using `tar`.
Take a screenshot of the task code. Name the screenshot `unzip_data.jpg`.
You can locally `untar` and read through the file `fileformats.txt` to understand the column details.
2. Create a task 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`.
Take a screenshot of the task code. Name the screenshot `extract_data_from_csv.jpg`.
3. Create a task named `extract_data_from_tsv` to extract the fields `Number of axes`, `Tollplaza id`, and `Tollplaza code` from the `tollplaza-data.tsv` file and save it into a file

```
ETL_toll_data.py
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27

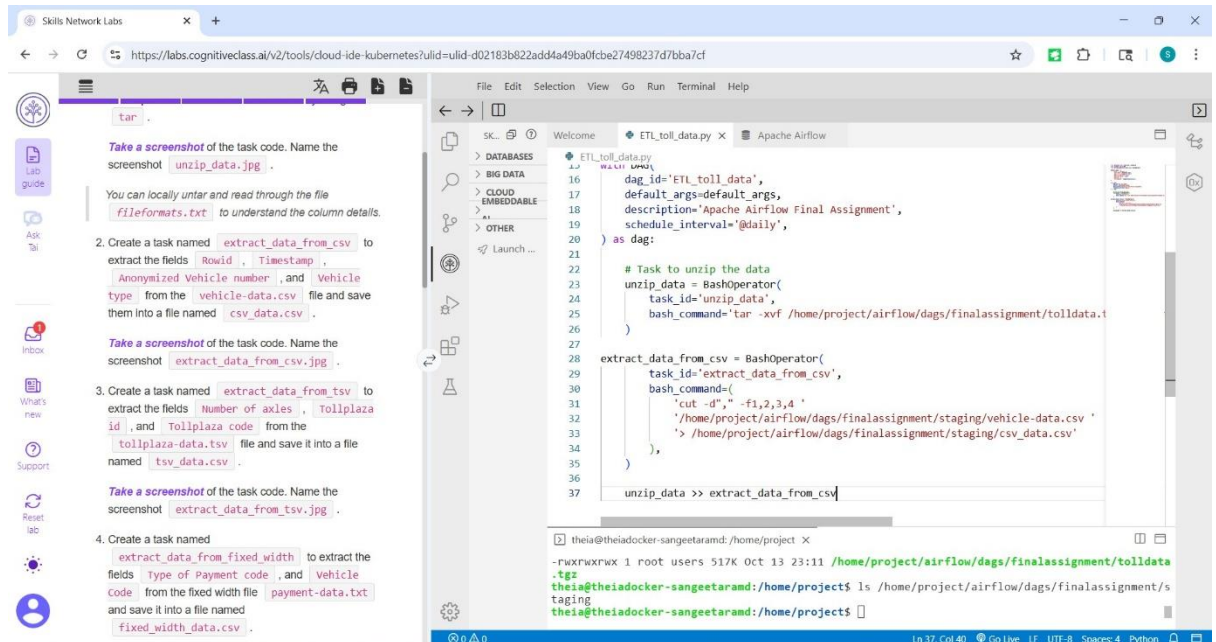
default_args = {
    'owner': 'sangeeta K',
    'start_date': datetime.now(),
    'email': ['srk12@gmail.com'],
    'email_on_failure': True,
    'email_on_retry': True,
    'retries': 1,
    'retry_delay': timedelta(minutes=5),
}

with DAG(
    dag_id='ETL_toll_data',
    default_args=default_args,
    description='Apache Airflow Final Assignment',
    schedule_interval='@daily',
) as dag:

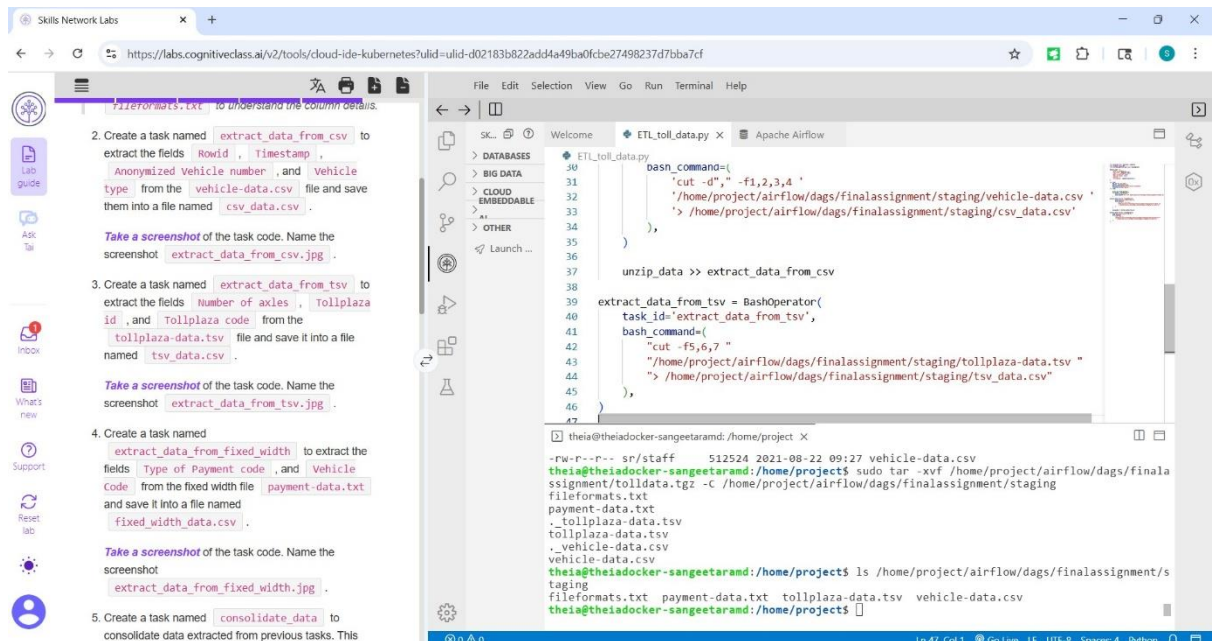
    # Task to unzip the data
    unzip_data = BashOperator(
        task_id='unzip_data',
        bash_command='tar -xvf /home/project/airflow/dags/finalassignment/tolldata.tgz'
    )

theia@theiadocker-sangeetaramd:/home/project X
.tgz
theia@theiadocker-sangeetaramd:/home/project$ sudo touch ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/project$ ls -l /home/project/ETL_toll_data.py
-rw-r--r-- 1 theia users 708 Oct 13 23:19 /home/project/ETL_toll_data.py
theia@theiadocker-sangeetaramd:/home/project$
```

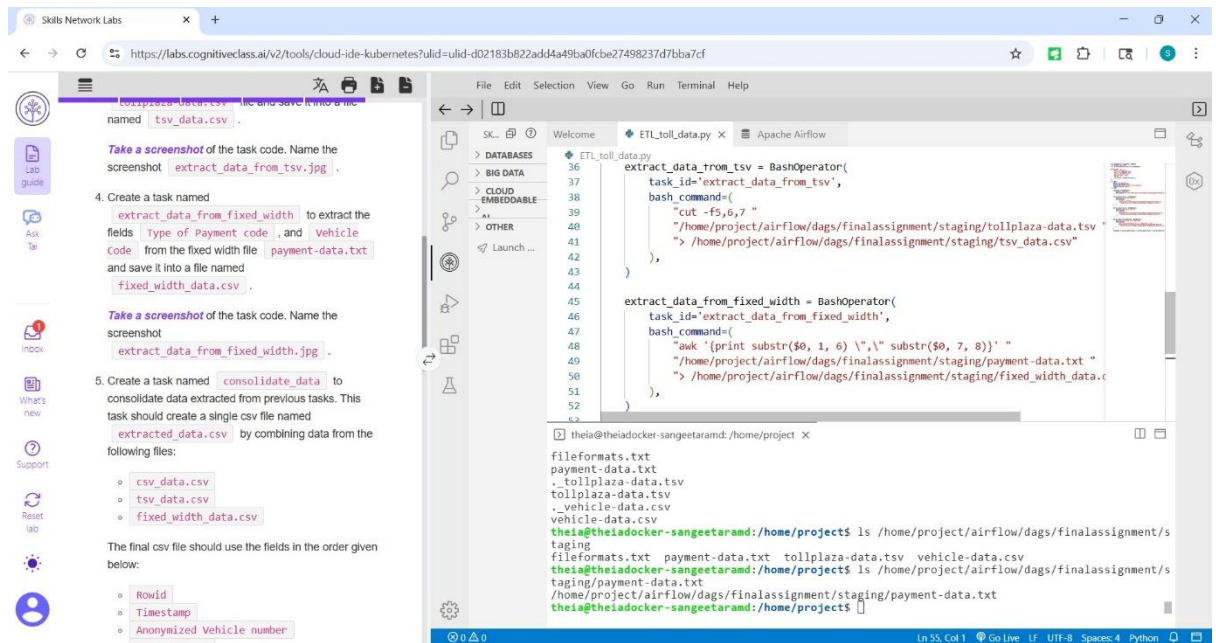
- Create a task named `extract_data_from_csv` to extract the fields `Rowid`, `Timestamp`, `Anonymized Vehicle number`, and `Vehicle type` from the `vehicle-data.csv` file and save them into a file named `csv_data.csv`.



- Create a task 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`.



- Create a task 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`.



- Create a task named `consolidate_data` to consolidate data extracted from previous tasks. 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
- Vehicle Code

Skills Network Labs

https://labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=ulid-d02183b822add4a49ba0fcb27498237d7bba7cf

extract_data_from_csv.jpg

4. Create a task 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`.

Take a screenshot of the task code. Name the screenshot `extract_data_from_fixed_width.jpg`.

5. Create a task named `consolidate_data` to consolidate data extracted from previous tasks. 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 axes`
- `Tollplaza id`
- `Tollplaza code`

File Edit Selection View Go Run Terminal Help

ETL_toll_data.py Apache Airflow

DATABASES

BIG DATA

CLOUD EMBEDDABLE

OTHER

Launch...

Debug

```
51 ),
52 )
53
54 unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_fro
55
56
57 consolidate_data = BashOperator(
58     task_id='consolidate_data',
59     bash_command=(
60         "paste -d, "
61         "/home/project/airflow/dags/finalassignment/staging/csv_data.csv "
62         "/home/project/airflow/dags/finalassignment/staging/tsv_data.csv "
63         "/home/project/airflow/dags/finalassignment/staging/fixed_width_data.csv
64         "> /home/project/airflow/dags/finalassignment/staging/extracted_data.csv
65     ),
66     unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_fro
```

theia@theiadocker-sangeetaramd:/home/project

fileformats.txt payment-data.txt tollplaza-data.tsv vehicle-data.csv

theia@theiadocker-sangeetaramd:/home/project\$ ls /home/project/airflow/dags/finalassignment/s

taging/payment-data.txt

theia@theiadocker-sangeetaramd:/home/project\$ ls /home/project/airflow/dags/finalassignment/s

taging

fileformats.txt payment-data.txt tollplaza-data.tsv vehicle-data.csv

theia@theiadocker-sangeetaramd:/home/project\$ /home/project/airflow/dags/finalassignment/stag

ing/extracted_data.csv

bash: /home/project/airflow/dags/finalassignment/staging/extracted_data.csv: No such file or

directory

theia@theiadocker-sangeetaramd:/home/project\$

Ln 66, Col 118 Go Live LF UTF-8 Spaces: 4 Python

- Create a task 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.

The screenshot shows a Skills Network Labs interface. On the left, a task guide lists steps for creating an Airflow DAG. Step 6 instructs to create a task 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. A hint suggests using the `tr` command within the BashOperator in Airflow. Step 7 defines the task pipeline.

On the right, the Apache Airflow DAG editor shows the code for the `transform_data` task. The code uses a BashOperator to run the `tr` command to convert the `vehicle_type` field to uppercase.

```

transform_data = BashOperator(
    task_id='transform_data',
    bash_command=(
        "awk -F',' ' {OFS=\",\"; $4=toupper($4); print $0} ' "
        "/home/project/airflow/dags/finalassignment/staging/extracted_data.csv "
        "> /home/project/airflow/dags/finalassignment/staging/transformed_data.csv"
    ),
)

```

The terminal output shows the command being executed and the resulting file listing.

```

theia@theiadocker-sangeetaramd:/home/project $ ls -lh /home/project/airflow/dags/finalassignment/staging/extracted_data.csv
-rw-r--r-- 1 theia users 745K Oct 13 23:58 /home/project/airflow/dags/finalassignment/staging/extracted_data.csv
theia@theiadocker-sangeetaramd:/home/project $

```

- Define the task pipeline as per the details given below:

Task	Functionality
First task	unzip_data
Second task	extract_data_from_csv
Third task	extract_data_from_tsv
Fourth task	extract_data_from_fixed_width
Fifth task	consolidate_data
Sixth task	transform_data

Skills Network Labs

https://labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=d02183b82add4a49ba0fcb27498237d7bba7cf

Task

Second task extr

Third task extr

Fourth task extr

Fifth task cons

Sixth task trar

Take a screenshot of the task pipeline section of the DAG. Name the screenshot task_pipeline.jpg.

At the end of this exercise, you should have the following screenshots with .jpg or .png extension:

1. unzip_data.jpg

2. extract_data_from_csv.jpg

3. extract_data_from_tsv.jpg

4. extract_data_from_fixed_width.jpg

File Edit Selection View Go Run Terminal Help

ETL_toll_data.py

Apache Airflow

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

transform_data = BashOperator(
task_id='transform_data',
bash_command=(
"awk -F',' '{OFS=\"\\n\\n\"; \$4=toupper(\$4); print \$0}' "
"/home/project/airflow/dags/finalassignment/staging/extracted_data.csv "
"> /home/project/airflow/dags/finalassignment/staging/transformed_data.csv"
),
)

unzip_data >> extract_data_from_csv >> extract_data_from_tsv >> extract_data_from_fixed_width >> consolidate_data >> transform_data

theia@theiadocker-sangeetaramd:/home/project

/home/project/airflow/dags/finalassignment/staging/tsv_data.csv \

/home/project/airflow/dags/finalassignment/staging/fixed_width_data.csv \

/home/project/airflow/dags/finalassignment/staging/extracted_data.csv

theia@theiadocker-sangeetaramd:/home/project\$ ls -lh /home/project/airflow/dags/finalassignment/staging/extracted_data.csv

-rw-r--r-- 1 theia users 745K Oct 13 23:58 /home/project/airflow/dags/finalassignment/staging/extracted_data.csv

theia@theiadocker-sangeetaramd:/home/project\$ awk -F',' '{OFS="\\n\\n"; \$4=toupper(\$4); print \$0}' /home/project/airflow/dags/finalassignment/staging/extracted_data.csv > /home/project/airflow/dags/finalassignment/staging/transformed_data.csv

theia@theiadocker-sangeetaramd:/home/project\$ ls /home/project/airflow/dags/finalassignment/staging/transformed_data.csv

/home/project/airflow/dags/finalassignment/staging/transformed_data.csv

theia@theiadocker-sangeetaramd:/home/project\$

Task 4: Getting the DAG operational

- Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.

The screenshot shows a Skills Network Labs exercise page on the left and an Apache Airflow web interface on the right.

Exercise 4: Getting the DAG operational

1. Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.
Take a screenshot showing that the DAG you created is in the list of DAGs. Name the screenshot `submit_dag.jpg`.
2. Unpause and trigger the DAG through CLI or Web UI.
3. *Take a screenshot* of DAG unpaused on CLI or the GUI. Name the screenshot `unpause_trigger_dag.jpg`.
4. *Take a screenshot* of the tasks in the DAG run through CLI or Web UI. Name the screenshot `dag_tasks.jpg`.
5. *Take a screenshot* the DAG runs for the Airflow console through CLI or Web UI. Name the screenshot `dag_runs.jpg`.

Screenshot checklist

You should have the following screenshots with `.jpg` or `.png` extension:

Apache Airflow Web Interface:

- File Edit Selection View Go Run Terminal Help
- ETL_toll_data.py (Deleted)
- transform_data = BashOperator(
task_id='transform_data',
bash_command='awk -F"," '{OFS=","; \$4=toupper(\$4); print
'> /home/project/airflow/dags/finalassignment/
'> /home/project/airflow/dags/finalassignment/
dag=dag
)
- # ---- Task Pipeline ----
unzip_data >> extract_data_from_csv >> extract_data_from_tsv
- thela@theladocker-sangeetaramd:/home/project
- 95 DeprecationWarning: The sql_alchemy_conn option in [core] has been moved to the sql_alchemy_conn option in [database] - the old setting has been used, but please update your config.
/home/airflow/.local/lib/python3.9/site-packages/airflow/models/base.py:72 DeprecationWarning: The sql_alchemy_conn option in [core] has been moved to the sql_alchemy_conn option in [database] - the old setting has been used, but please update your config.
- dag_id | fileloc | owners | is_paused
- ETL_toll_data | /home/project/airflow/dags/ETL_toll_data.py | Sangeeta K | None
- conditional_dataset_and_time_based_timetable | /home/airflow/.local/lib/python3.9/site-packages/airflow/example_dags/example_dataset.py | airflow | True
- consume_1_and_2_with_dataset_expressions | /home/airflow/.local/lib/python3.9/site-pa | airflow | True

➤ Unpause and trigger the DAG through CLI or Web UI.

operational

1. Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.
Take a screenshot showing that the DAG you created is in the list of DAGs. Name the screenshot `submit_dag.jpg`.
2. Unpause and trigger the DAG through CLI or Web UI.
Take a screenshot of DAG unpaused on CLI or the GUI. Name the screenshot `unpause_trigger_dag.jpg`.
3. Take a screenshot of the tasks in the DAG run through CLI or Web UI. Name the screenshot `dag_tasks.jpg`.
4. Take a screenshot of the DAG runs for the Airflow console through CLI or Web UI. Name the screenshot `dag_runs.jpg`.

Screenshot checklist

You should have the following screenshots with `.jpg` or `.png` extension:

1. `submit_dag.jpg`
2. `unpause_trigger_dag.jpg`
3. `dag_tasks.jpg`
4. `dag_runs.jpg`

Apache Airflow DAG Definition:

```
airflow > dags > ETL_toll_data.py
17 'retries': 1,
18 'retry_delay': timedelta(minutes=5),
19 }
20
21 # ---- DAG definition ----
22
```

dag_id	fileloc	owners	is_paused
ETL_toll_data	/home/project/airflow/dags/ETL_toll_data.py	Sangeeta K	False
conditional_dataset_and_time_based_timetable	/home/airflow/.local/lib/python3.9/site-packages/airflow/example_dags/example_datasets.py	airflow	True
consume_1_and_2_with_dataset_expressions	/home/airflow/.local/lib/python3.9/site-packages/airflow/example_dags/example_datasets.py	airflow	True

operational

1. Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.
Take a screenshot showing that the DAG you created is in the list of DAGs. Name the screenshot `submit_dag.jpg`.
2. Unpause and trigger the DAG through CLI or Web UI.
Take a screenshot of DAG unpaused on CLI or the GUI. Name the screenshot `unpause_trigger_dag.jpg`.
3. Take a screenshot of the tasks in the DAG run through CLI or Web UI. Name the screenshot `dag_tasks.jpg`.
4. Take a screenshot of the DAG runs for the Airflow console through CLI or Web UI. Name the screenshot `dag_runs.jpg`.

Screenshot checklist

You should have the following screenshots with `.jpg` or `.png` extension:

1. `submit_dag.jpg`
2. `unpause_trigger_dag.jpg`
3. `dag_tasks.jpg`
4. `dag_runs.jpg`

Apache Airflow DAG Definition:

```
airflow > dags > ETL_toll_data.py
17 'retries': 1,
18 'retry_delay': timedelta(minutes=5),
19 }
20
21 # ---- DAG definition ----
22
```

dag_id	fileloc	owners	is_paused
ETL_toll_data	/home/project/airflow/dags/ETL_toll_data.py	Sangeeta K	False
conditional_dataset_and_time_based_timetable	/home/airflow/.local/lib/python3.9/site-packages/airflow/example_dags/example_datasets.py	airflow	True
consume_1_and_2_with_dataset_expressions	/home/airflow/.local/lib/python3.9/site-packages/airflow/example_dags/example_datasets.py	airflow	True

Skills Network Labs

← → ↻ 🔍 https://labs.cognitiveclass.ai/v2/tools/cloud-ide-kubernetes?ulid=d02183b822add4a49ba0fcb27498237d7bba7cf

Lab guide

Appt

Inbox

What's new

Support

Reset lab

operational

1. Submit the DAG. Use CLI or Web UI to show that the DAG has been properly submitted.
Take a screenshot showing that the DAG you created is in the list of DAGs. Name the screenshot `submit_dag.jpg`.

Note: If you don't find your DAG in the list, you can check for errors using the following command in the terminal:
2. Unpause and trigger the DAG through CLI or Web UI.
3. *Take a screenshot* of DAG unpause on CLI or the GUI. Name the screenshot `unpause_trigger_dag.jpg`.
4. *Take a screenshot* of the tasks in the DAG run through CLI or Web UI. Name the screenshot `dag_tasks.jpg`.
5. *Take a screenshot* the DAG runs for the Airflow console through CLI or Web UI. Name the screenshot `dag_runs.jpg`.

Screenshot checklist

You should have the following screenshots with `.jpg` or `.png` extension:

1. `submit_dag.jpg`
2. `unpause_trigger_dag.jpg`

File Edit Selection View Go Run Terminal Help

SK... ①

Welcome Apache Airflow

ETL_toll_data.py X

DATABASES

BIG DATA

Apache Air...

CLOUD EMBEDDABLE

AI

OTHER

airflow > dags > ETL_toll_data.py

17 retries': 1,

18 'retry_delay': timedelta(minutes=5),

19 }

20 }

21 # ---- DAG definition ----

thela@theiadocker-sangeetaramd: /home/project X

Launch ...

```
/home/airflow/.local/lib/python3.9/site-packages/airflow/configuration.py:747 Deprecati
onWarning: The auth_backend option in [api] has been renamed to auth_backends - the old
setting has been used, but please update your config.
/home/airflow/.local/lib/python3.9/site-packages/airflow/configuration.py:761 FutureWarn
ing: The auth_backend setting in [api] has had airflow.api.auth.backend.session added
in the running config, which is needed by the UI. Please update your config before Apa
che Airflow 3.0.
/home/airflow/.local/lib/python3.9/site-packages/airflow/configuration.py:738 Deprecati
onWarning: The sql_alchemy_conn option in [core] has been moved to the sql_alchemy_conn
option in [database] - the old setting has been used, but please update your config.
/home/airflow/.local/lib/python3.9/site-packages/airflow/settings.py:195 DeprecationWarn
ing: The sql_alchemy_conn option in [core] has been moved to the sql_alchemy_conn opti
on in [database] - the old setting has been used, but please update your config.
/home/airflow/.local/lib/python3.9/site-packages/airflow/models/base.py:72 DeprecationW
arning: The sql_alchemy_conn option in [core] has been moved to the sql_alchemy_conn op
tion in [database] - the old setting has been used, but please update your config.
```

dag_id	run_id	state	execution_dat	start_date	end_date
ETL_toll_data	manual_2025-10-14T20:51:46+00:00	success	2025-10-14T20:51:46+00:00	2025-10-14T20:51:47.760943	2025-10-14T20:52:06.801196
ETL_toll_data	manual_2025-10-14T20:45:48+00:00	success	2025-10-14T20:45:48+00:00	2025-10-14T20:45:49.280913	2025-10-14T20:46:14.533192

thela@theiadocker-sangeetaramd: /home/project\$

Ln 1, Col 1 Go Live LF UTF-8 Spaces 4 Python