Année d'étude :2023

INFO-H420 Management of Data Science and Business Workflows

Assignment 3 : Workflows with Apache Airflow

Min Zhang 000586970

Yutao Chen 000585954

Prof. Dimitris SACHARIDIS

Nov 24 2023

Table des matières

Listings

Table des figures

1

Exercise 1 (10 points))

— Define the DAG. Create a DAG named `process_web_log` that runs daily.

— Create a task to scan for a log. Create a task named `scan_for_log` that scans a folder `the_logs` for a `log.txt` file and triggers the rest of the workflow.

— Create a task to extract data. Create a task named `extract_data`. This task should extract the `ipaddress` field from the web server log file and save it into a file named `extracted_data.txt`.

— Create a task to transform data. Create a task named `transform_data`. This task should filter out all the occurrences of `ipaddress 198.46.149.143` from `extracted_data.txt` and save the output to a file named `transformed_data.txt`.

— Create a task to load the data. Create a task named `load_data`. This task should archive the file `transformed_data.txt` into a tar file named `weblog.tar`.

— Define the workflow that executes the aforementioned tasks in sequence.

Save the DAG you defined into a file named `process_web_log.py`. In the report, include code snippets describing how you achieved this.

1.1 Assumptions

1.The DAG does not depend on the success of its previous run.

2.The DAG runs on a calendar every day, including weekdays and weekends.

1.2 Solution

1.2.1 DAG definition

Listing 1.1: DAG definition with default arguments.

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(seconds=1),
}

dag = DAG('process_web_log',
          default_args=default_args,
          description='DAG for processing web log',
          schedule_interval='@daily',
          catchup=False,
          tags=['DSBW'])
```

1.2.2 File path definition

Listing 1.2: Definition of file paths for logs and processed data.

```
log_dir = '/usr/local/airflow'
log_file = f'{log_dir}/log.txt'
extracted_data_file = f'{log_dir}/extracted_data.txt'
transformed_data_file = f'{log_dir}/transformed_data.txt'
tar_file = f'{log_dir}/weblog.tar'
```

1.2.3 Function definition

Listing 1.3: Functions for scanning, extracting, transforming, and loading data.

```
def scan_for_log(**context):
    if os.path.isfile(log_file):
        return log_file
```

```python
    else:
        raise ValueError("log.txt not found")

def extract_data(**context):
    task_instance = context['ti']
    log_path = task_instance.xcom_pull(task_ids='scan_for_log')

    with open(log_path, 'r') as file, open(extracted_data_file, 'w') as out_file:
        for line in file:
            ip_address = line.split()[0]  # Assuming IP address is the first element in the log line
            out_file.write(ip_address + '\n')

def transform_data(**context):
    with open(extracted_data_file, 'r') as file, open(transformed_data_file, 'w') as out_file:
        for line in file:
            if '198.46.149.143' not in line:
                out_file.write(line)

def load_data(**context):
    with tarfile.open(tar_file, 'w') as tar:
        tar.add(transformed_data_file, arcname='transformed_data.txt')
```

1.2.4 Task definition

Listing 1.4: Task definitions in the Airflow DAG.

```python
scan_task = PythonOperator(
    task_id='scan_for_log',
    python_callable=scan_for_log,
    provide_context=True,
    dag=dag)

extract_task = PythonOperator(
    task_id='extract_data',
    python_callable=extract_data,
    provide_context=True,
    dag=dag)

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

load_task = PythonOperator(
    task_id='load_data',
    python_callable=load_data,
    provide_context=True,
    dag=dag)
```

2 Exercise 2 (2 points)

Do a test run for each of the tasks you defined. Once all works as expected, do a test run for the workflow. Finally, trigger/run the workflow and monitor a few runs. In the report, document the test runs, and include any findings or observations that you may have from the runs.

2.1 Assumptions

1. These tasks run independently from other tasks or resources on a daily basis.
2. The system is equipped with the necessary dependencies and environment for the tasks to run effectively.
3. The data required for each task is available and accessible at the time of execution.

2.2 CLI Test

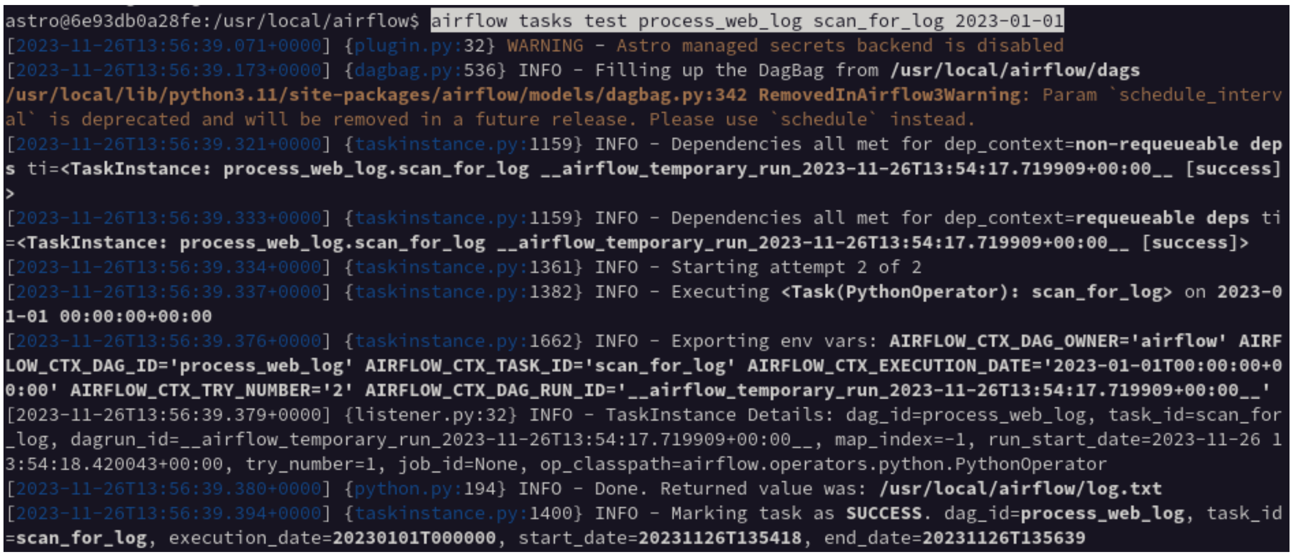
To ensure the robustness and reliability of the `process_web_log` DAG, each individual task was subjected to a rigorous testing procedure using the Airflow CLI. This facilitated a granular evaluation of task logic and functionality in isolation from the workflow. The CLI command used for the test was :

```
airflow tasks test <dag_id> <task_id> <execution_date>
```

The figures below illustrate the outcome of each task test, with a focus on confirming the expected behavior of the task logic and its successful interaction with the required data and resources.

2.2.1 Test of Scan for Log Task

The `scan_for_log` task was tested to validate its capability to locate and identify the log file. As depicted in Figure 2.1, the task executed successfully, confirming the presence of the log file.

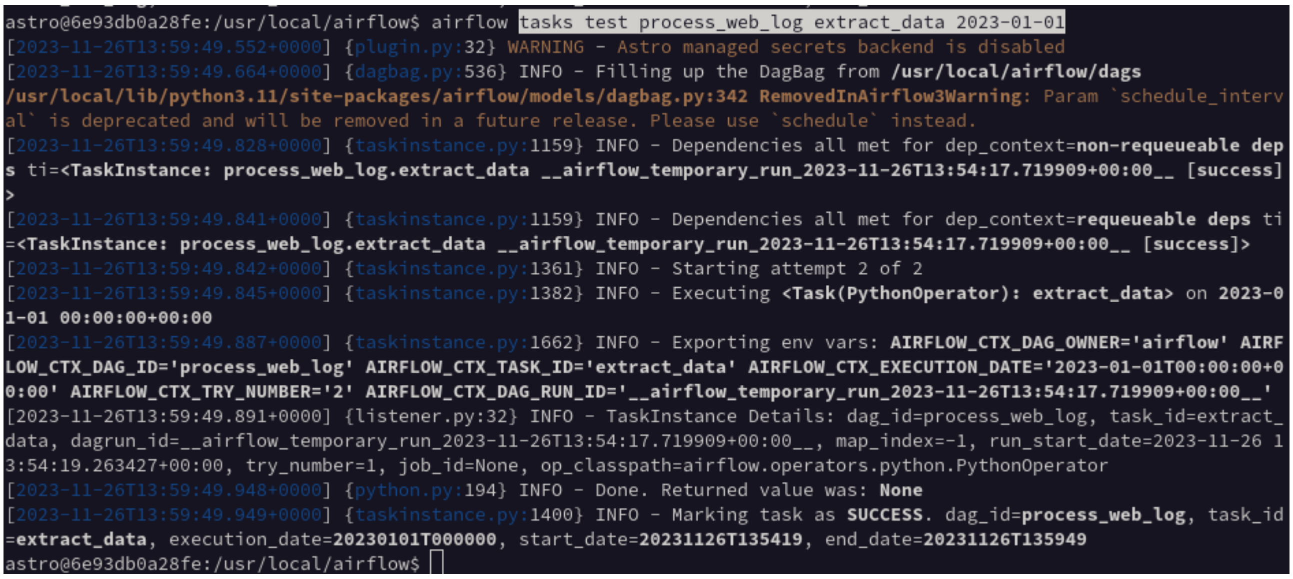


```
astro@6e93db0a28fe:/usr/local/airflow$ airflow tasks test process_web_log scan_for_log 2023-01-01
[2023-11-26T13:56:39.071+0000] {plugin.py:32} WARNING - Astro managed secrets backend is disabled
[2023-11-26T13:56:39.173+0000] {dagbag.py:536} INFO - Filling up the DagBag from /usr/local/airflow/dags
/usr/local/lib/python3.11/site-packages/airflow/models/dagbag.py:342 RemovedInAirflow3Warning: Param `schedule_interv
al` is deprecated and will be removed in a future release. Please use `schedule` instead.
[2023-11-26T13:56:39.321+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable dep
s ti=<TaskInstance: process_web_log.scan_for_log __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]
>
[2023-11-26T13:56:39.333+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti
=<TaskInstance: process_web_log.scan_for_log __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]>
[2023-11-26T13:56:39.334+0000] {taskinstance.py:1361} INFO - Starting attempt 2 of 2
[2023-11-26T13:56:39.337+0000] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): scan_for_log> on 2023-0
1-01 00:00:00+00:00
[2023-11-26T13:56:39.376+0000] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRF
LOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='scan_for_log' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+0
0:00' AIRFLOW_CTX_TRY_NUMBER='2' AIRFLOW_CTX_DAG_RUN_ID='__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__'
[2023-11-26T13:56:39.379+0000] {listener.py:32} INFO - TaskInstance Details: dag_id=process_web_log, task_id=scan_for
_log, dagrun_id=__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__, map_index=-1, run_start_date=2023-11-26 1
3:54:18.420043+00:00, try_number=1, job_id=None, op_classpath=airflow.operators.python.PythonOperator
[2023-11-26T13:56:39.380+0000] {python.py:194} INFO - Done. Returned value was: /usr/local/airflow/log.txt
[2023-11-26T13:56:39.394+0000] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id
=scan_for_log, execution_date=20230101T000000, start_date=20231126T135418, end_date=20231126T135639
```

FIGURE 2.1: test_scan_for_log : The successful detection of the log file necessary for further processing.

2.2.2 Test of Extract Data Task

Following the successful log scanning, the `extract_data` task was assessed. This task's responsibility is to extract the requisite data fields from the log file. Figure 2.2 showcases a successful extraction, laying the groundwork for the subsequent transformation step.

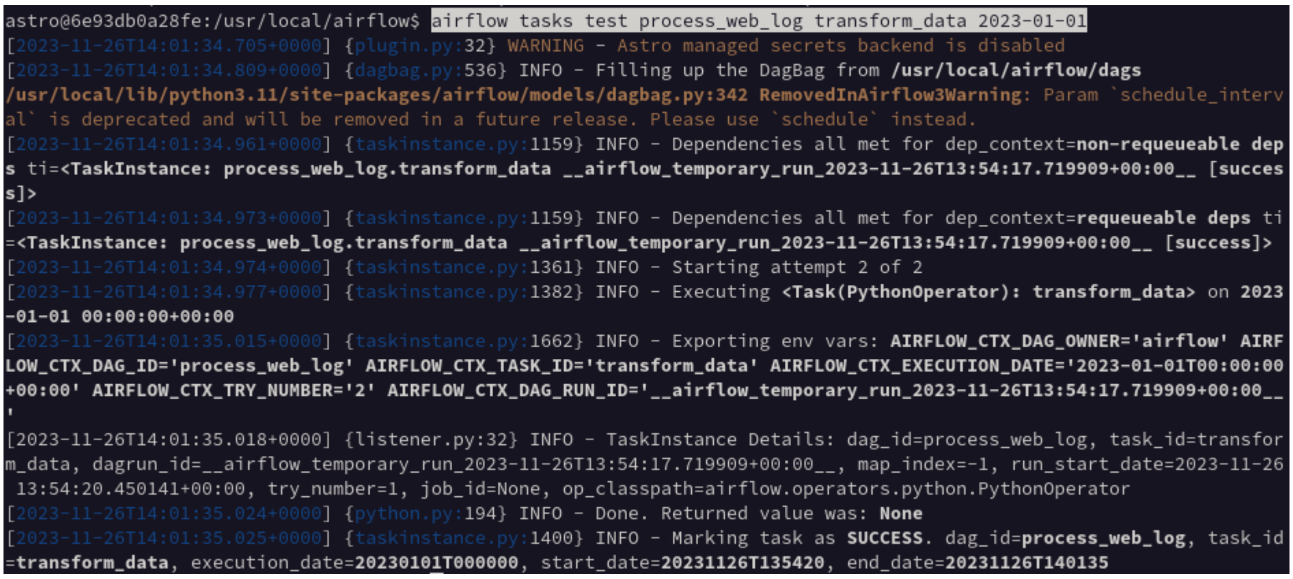


```
astro@6e93db0a28fe:/usr/local/airflow$ airflow tasks test process_web_log extract_data 2023-01-01
[2023-11-26T13:59:49.552+0000] {plugin.py:32} WARNING - Astro managed secrets backend is disabled
[2023-11-26T13:59:49.664+0000] {dagbag.py:536} INFO - Filling up the DagBag from /usr/local/airflow/dags
/usr/local/lib/python3.11/site-packages/airflow/models/dagbag.py:342 RemovedInAirflow3Warning: Param `schedule_interv
al` is deprecated and will be removed in a future release. Please use `schedule` instead.
[2023-11-26T13:59:49.828+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable dep
s ti=<TaskInstance: process_web_log.extract_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]
>
[2023-11-26T13:59:49.841+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti
=<TaskInstance: process_web_log.extract_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]>
[2023-11-26T13:59:49.842+0000] {taskinstance.py:1361} INFO - Starting attempt 2 of 2
[2023-11-26T13:59:49.845+0000] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): extract_data> on 2023-0
1-01 00:00:00+00:00
[2023-11-26T13:59:49.887+0000] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRF
LOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='extract_data' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+0
0:00' AIRFLOW_CTX_TRY_NUMBER='2' AIRFLOW_CTX_DAG_RUN_ID='__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__'
[2023-11-26T13:59:49.891+0000] {listener.py:32} INFO - TaskInstance Details: dag_id=process_web_log, task_id=extract_
data, dagrun_id=__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__, map_index=-1, run_start_date=2023-11-26 1
3:54:19.263427+00:00, try_number=1, job_id=None, op_classpath=airflow.operators.python.PythonOperator
[2023-11-26T13:59:49.948+0000] {python.py:194} INFO - Done. Returned value was: None
[2023-11-26T13:59:49.949+0000] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id
=extract_data, execution_date=20230101T000000, start_date=20231126T135419, end_date=20231126T135949
astro@6e93db0a28fe:/usr/local/airflow$
```

FIGURE 2.2: test_extract_data : Extraction of data fields verified, ensuring data integrity for the next phase.

2.2.3 Test of Transform Data Task

The `transform_data` task's purpose is to apply the necessary transformations to the extracted data. The test, as shown in Figure 2.3, confirmed that the task performed the transformations correctly and the output was as intended.

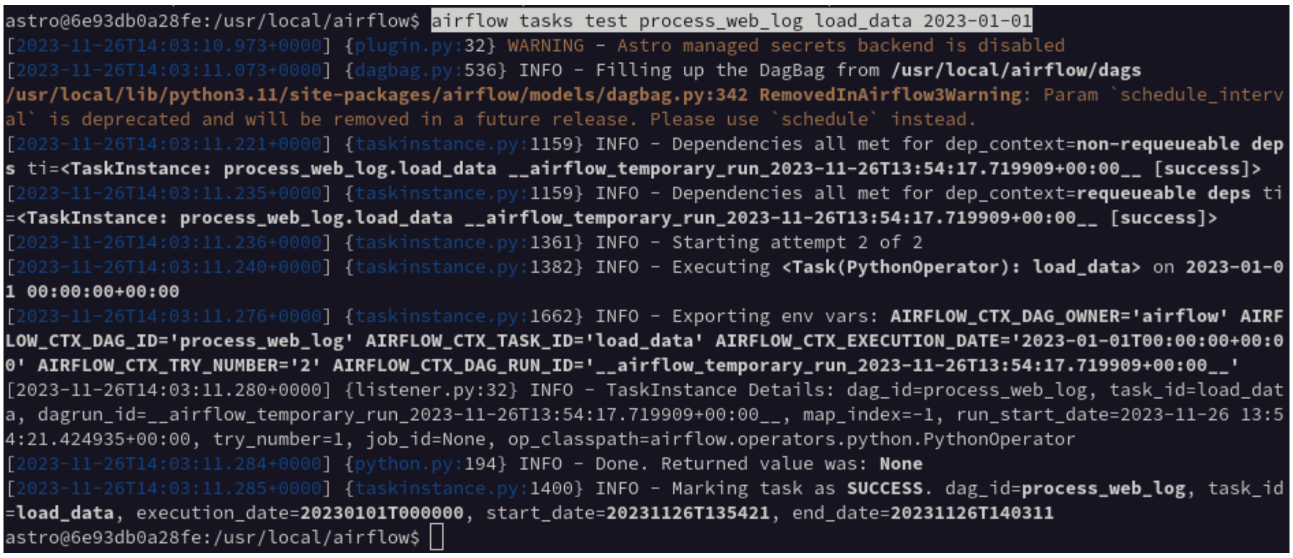


```
astro@6e93db0a28fe:/usr/local/airflow$ airflow tasks test process_web_log transform_data 2023-01-01
[2023-11-26T14:01:34.705+0000] {plugin.py:32} WARNING - Astro managed secrets backend is disabled
[2023-11-26T14:01:34.809+0000] {dagbag.py:536} INFO - Filling up the DagBag from /usr/local/airflow/dags
/usr/local/lib/python3.11/site-packages/airflow/models/dagbag.py:342 RemovedInAirflow3Warning: Param `schedule_interv
al` is deprecated and will be removed in a future release. Please use `schedule` instead.
[2023-11-26T14:01:34.961+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable dep
s ti=<TaskInstance: process_web_log.transform_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [succes
s]>
[2023-11-26T14:01:34.973+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti
=<TaskInstance: process_web_log.transform_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]>
[2023-11-26T14:01:34.974+0000] {taskinstance.py:1361} INFO - Starting attempt 2 of 2
[2023-11-26T14:01:34.977+0000] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): transform_data> on 2023
-01-01 00:00:00+00:00
[2023-11-26T14:01:35.015+0000] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRF
LOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='transform_data' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00
+00:00' AIRFLOW_CTX_TRY_NUMBER='2' AIRFLOW_CTX_DAG_RUN_ID='__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__
'
[2023-11-26T14:01:35.018+0000] {listener.py:32} INFO - TaskInstance Details: dag_id=process_web_log, task_id=transfor
m_data, dagrun_id=__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__, map_index=-1, run_start_date=2023-11-26
 13:54:20.450141+00:00, try_number=1, job_id=None, op_classpath=airflow.operators.python.PythonOperator
[2023-11-26T14:01:35.024+0000] {python.py:194} INFO - Done. Returned value was: None
[2023-11-26T14:01:35.025+0000] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id
=transform_data, execution_date=20230101T000000, start_date=20231126T135420, end_date=20231126T140135
```

FIGURE 2.3: test_transform_data : Validation of data transformation logic and output formatting.

2.2.4 Test of Load Data Task

Lastly, the `load_data` task was evaluated to ensure it could successfully load the transformed data to the target destination. The success of this task, as observed in Figure 2.4, signifies the readiness of the data for any downstream processes or storage solutions.

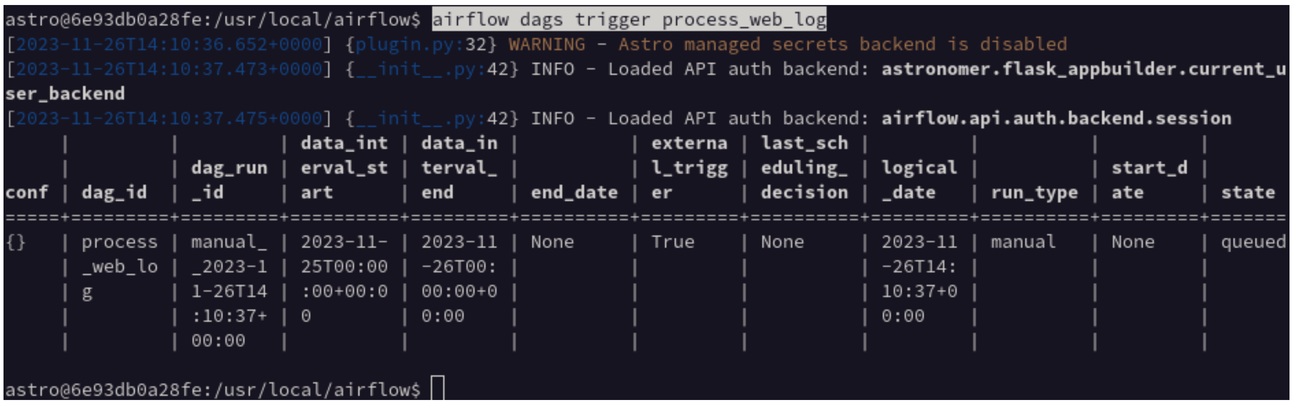


```
astro@6e93db0a28fe:/usr/local/airflow$ airflow tasks test process_web_log load_data 2023-01-01
[2023-11-26T14:03:10.973+0000] {plugin.py:32} WARNING - Astro managed secrets backend is disabled
[2023-11-26T14:03:11.073+0000] {dagbag.py:536} INFO - Filling up the DagBag from /usr/local/airflow/dags
/usr/local/lib/python3.11/site-packages/airflow/models/dagbag.py:342 RemovedInAirflow3Warning: Param `schedule_interv
al` is deprecated and will be removed in a future release. Please use `schedule` instead.
[2023-11-26T14:03:11.221+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable dep
s ti=<TaskInstance: process_web_log.load_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]>
[2023-11-26T14:03:11.235+0000] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti
=<TaskInstance: process_web_log.load_data __airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__ [success]>
[2023-11-26T14:03:11.236+0000] {taskinstance.py:1361} INFO - Starting attempt 2 of 2
[2023-11-26T14:03:11.240+0000] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): load_data> on 2023-01-0
1 00:00:00+00:00
[2023-11-26T14:03:11.276+0000] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRF
LOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='load_data' AIRFLOW_CTX_EXECUTION_DATE='2023-01-01T00:00:00+00:0
0' AIRFLOW_CTX_TRY_NUMBER='2' AIRFLOW_CTX_DAG_RUN_ID='__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__'
[2023-11-26T14:03:11.280+0000] {listener.py:32} INFO - TaskInstance Details: dag_id=process_web_log, task_id=load_dat
a, dagrun_id=__airflow_temporary_run_2023-11-26T13:54:17.719909+00:00__, map_index=-1, run_start_date=2023-11-26 13:5
4:21.424935+00:00, try_number=1, job_id=None, op_classpath=airflow.operators.python.PythonOperator
[2023-11-26T14:03:11.284+0000] {python.py:194} INFO - Done. Returned value was: None
[2023-11-26T14:03:11.285+0000] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id
=load_data, execution_date=20230101T000000, start_date=20231126T135421, end_date=20231126T140311
astro@6e93db0a28fe:/usr/local/airflow$
```

FIGURE 2.4: test_load_data : Confirmation of the loading mechanism's operational status and data readiness.

2.2.5 Triggering the Whole Workflow

Having each task confirmed for operational readiness, the entire workflow was triggered to observe the cohesive execution of the tasks in a sequence that mimics the production environment. Figure 2.5 reflects the successful execution of the entire workflow as a unified process, demonstrating the DAG's effectiveness in orchestrating the defined tasks.



```
astro@6e93db0a28fe:/usr/local/airflow$ airflow dags trigger process_web_log
[2023-11-26T14:10:36.652+0000] {plugin.py:32} WARNING - Astro managed secrets backend is disabled
[2023-11-26T14:10:37.473+0000] {__init__.py:42} INFO - Loaded API auth backend: astronomer.flask_appbuilder.current_u
ser_backend
[2023-11-26T14:10:37.475+0000] {__init__.py:42} INFO - Loaded API auth backend: airflow.api.auth.backend.session
     |          |          | data_int | data_in |           | externa | last_sch |          |          |          |
     |          | dag_run  | erval_st | terval_ |           | l_trigg | eduling_ | logical  |          | start_d  |
conf | dag_id   | _id      | art      | end     | end_date  | er      | decision | _date    | run_type | ate      | state
=====+==========+==========+==========+=========+===========+=========+==========+==========+==========+==========+=======
{}   | process  | manual_  | 2023-11- | 2023-11 | None      | True    | None     | 2023-11  | manual   | None     | queued
     | _web_lo  | _2023-1  | 25T00:00 | -26T00: |           |         |          | -26T14:  |          |          |
     | g        | 1-26T14  | :00+00:0 | 00:00+0 |           |         |          | 10:37+0  |          |          |
     |          | :10:37+  | 0        | 0:00    |           |         |          | 0:00     |          |          |
     |          | 00:00    |          |         |           |         |          |          |          |          |

astro@6e93db0a28fe:/usr/local/airflow$
```

FIGURE 2.5: trigger_whole_process : The culmination of individual task tests in the successful execution of the full workflow.

2.3 GUI Test

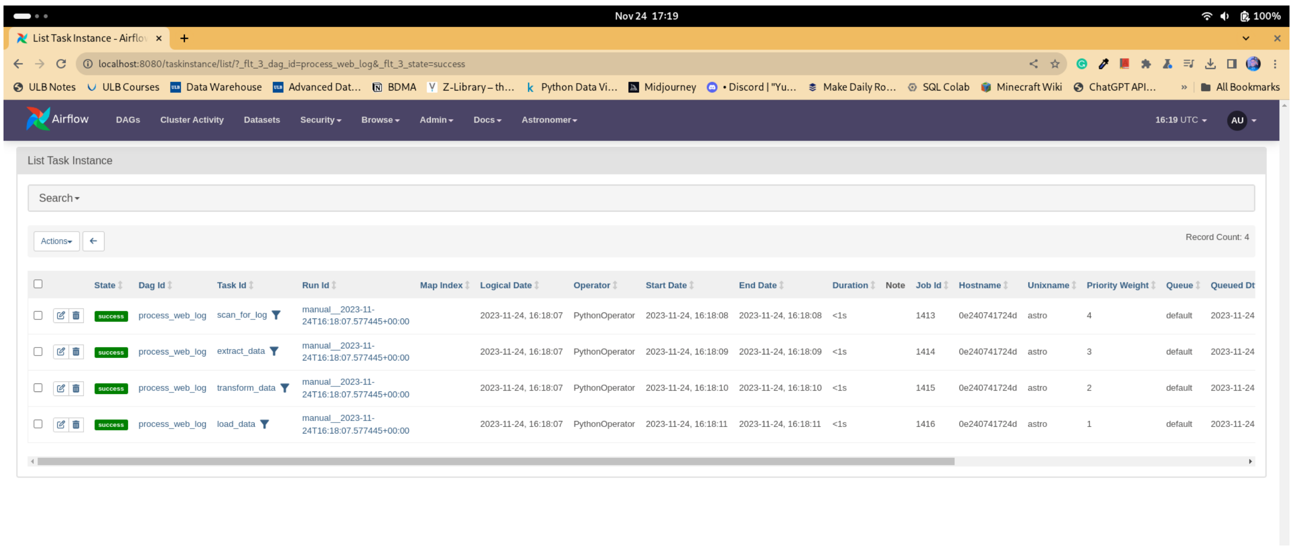
A comprehensive graphical user interface (GUI) testing was conducted to visually monitor the execution of the defined tasks within the `process_web_log` DAG. This involved observing and recording the run logs, execution times, and statuses of each task to ensure operational transparency and identify any potential bottlenecks or errors.

2.3.1 Overview of Task Execution

Figure 2-1 presents a consolidated view of the execution of the four tasks. It provides a snapshot of each task's running status, execution time, and duration, offering insights into the workflow's overall efficiency and task coordination.

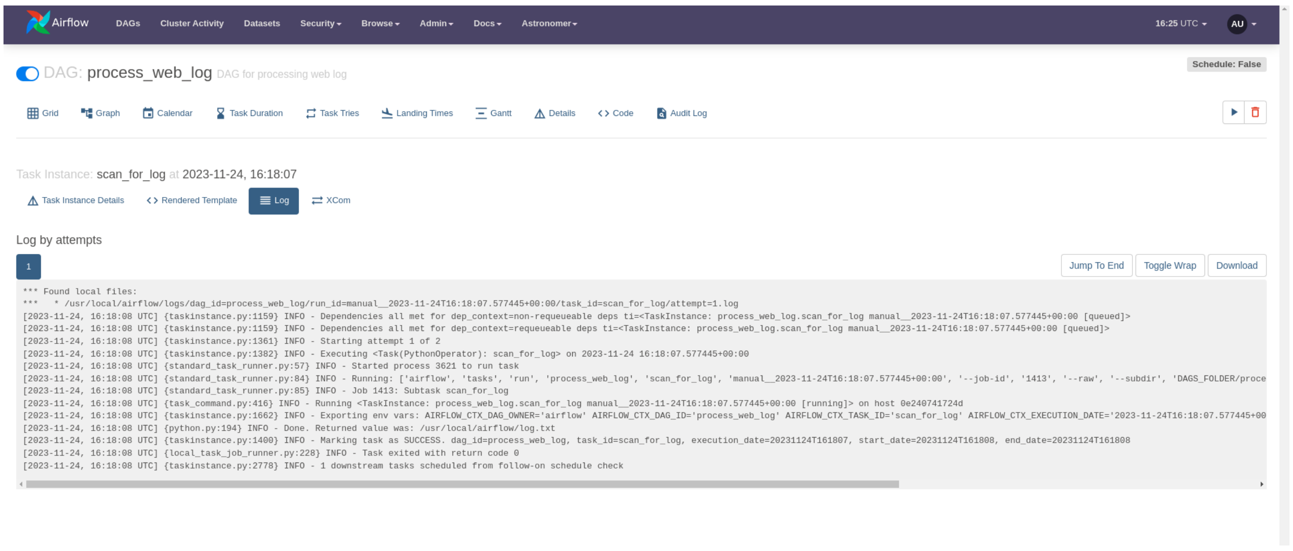
2.3.2 Detailed Run Logs

The run logs for each task were meticulously documented to trace the execution flow and debug any issues that arose during the tasks' lifecycle. Figures 2-2 to 2-5 capture the detailed logs, providing a granular look into the internal operations of each task.



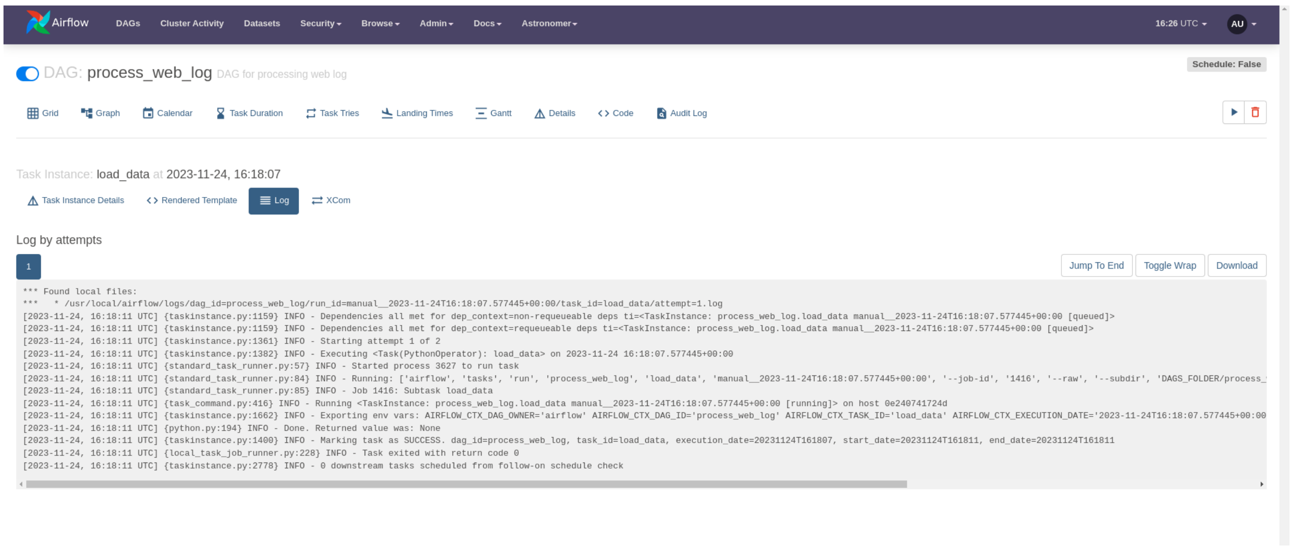
State	Dag Id	Task Id	Run Id	Map Index	Logical Date	Operator	Start Date	End Date	Duration	Note	Job Id	Hostname	Unixname	Priority Weight	Queue	Queued Dt
success	process_web_log	scan_for_log	manual__2023-11-24T16:18:07.577445+00:00		2023-11-24, 16:18:07	PythonOperator	2023-11-24, 16:18:08	2023-11-24, 16:18:08	<1s		1413	0e240741724d	astro	4	default	2023-11-24
success	process_web_log	extract_data	manual__2023-11-24T16:18:07.577445+00:00		2023-11-24, 16:18:07	PythonOperator	2023-11-24, 16:18:09	2023-11-24, 16:18:09	<1s		1414	0e240741724d	astro	3	default	2023-11-24
success	process_web_log	transform_data	manual__2023-11-24T16:18:07.577445+00:00		2023-11-24, 16:18:07	PythonOperator	2023-11-24, 16:18:10	2023-11-24, 16:18:10	<1s		1415	0e240741724d	astro	2	default	2023-11-24
success	process_web_log	load_data	manual__2023-11-24T16:18:07.577445+00:00		2023-11-24, 16:18:07	PythonOperator	2023-11-24, 16:18:11	2023-11-24, 16:18:11	<1s		1416	0e240741724d	astro	1	default	2023-11-24

FIGURE 2.6: Consolidated results overview of the four tasks, illustrating their execution status and timing.



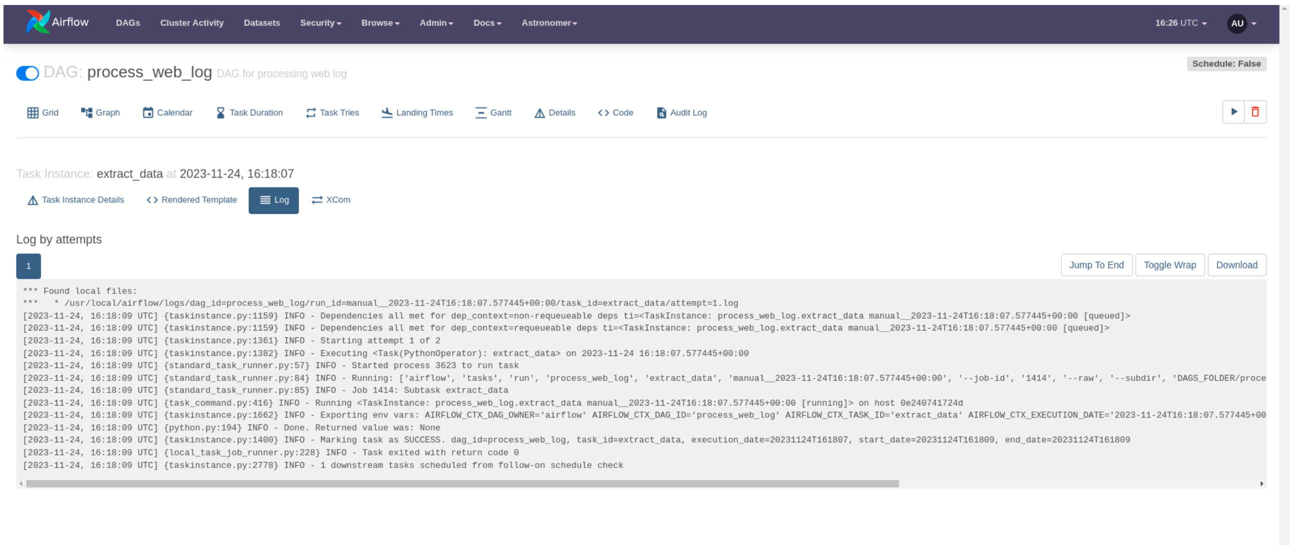
```
*** Found local files:
***   * /usr/local/airflow/logs/dag_id=process_web_log/run_id=manual__2023-11-24T16:18:07.577445+00:00/task_id=scan_for_log/attempt=1.log
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: process_web_log.scan_for_log manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: process_web_log.scan_for_log manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 2
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): scan_for_log> on 2023-11-24 16:18:07.577445+00:00
[2023-11-24, 16:18:08 UTC] {standard_task_runner.py:57} INFO - Started process 3621 to run task
[2023-11-24, 16:18:08 UTC] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'process_web_log', 'scan_for_log', 'manual__2023-11-24T16:18:07.577445+00:00', '--job-id', '1413', '--raw', '--subdir', 'DAGS_FOLDER/proce
[2023-11-24, 16:18:08 UTC] {standard_task_runner.py:85} INFO - Job 1413: Subtask scan_for_log
[2023-11-24, 16:18:08 UTC] {task_command.py:416} INFO - Running <TaskInstance: process_web_log.scan_for_log manual__2023-11-24T16:18:07.577445+00:00 [running]> on host 0e240741724d
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='scan_for_log' AIRFLOW_CTX_EXECUTION_DATE='2023-11-24T16:18:07.577445+00
[2023-11-24, 16:18:08 UTC] {python.py:194} INFO - Done. Returned value was: /usr/local/airflow/log.txt
[2023-11-24, 16:18:08 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id=scan_for_log, execution_date=20231124T161807, start_date=20231124T161808, end_date=20231124T161808
[2023-11-24, 16:18:08 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-11-24, 16:18:08 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

FIGURE 2.7: Detailed run log for Task 1, highlighting the step-by-step execution process.



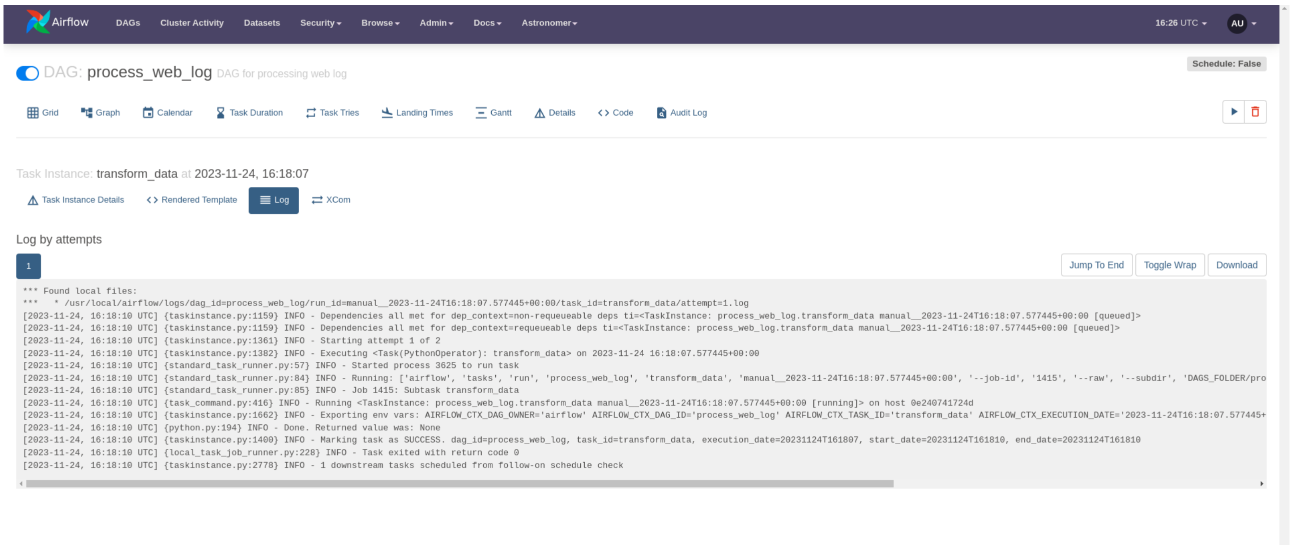
```
*** Found local files:
***   * /usr/local/airflow/logs/dag_id=process_web_log/run_id=manual__2023-11-24T16:18:07.577445+00:00/task_id=load_data/attempt=1.log
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: process_web_log.load_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: process_web_log.load_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 2
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): load_data> on 2023-11-24 16:18:07.577445+00:00
[2023-11-24, 16:18:11 UTC] {standard_task_runner.py:57} INFO - Started process 3627 to run task
[2023-11-24, 16:18:11 UTC] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'process_web_log', 'load_data', 'manual__2023-11-24T16:18:07.577445+00:00', '--job-id', '1416', '--raw', '--subdir', 'DAGS_FOLDER/process_
[2023-11-24, 16:18:11 UTC] {standard_task_runner.py:85} INFO - Job 1416: Subtask load_data
[2023-11-24, 16:18:11 UTC] {task_command.py:416} INFO - Running <TaskInstance: process_web_log.load_data manual__2023-11-24T16:18:07.577445+00:00 [running]> on host 0e240741724d
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='load_data' AIRFLOW_CTX_EXECUTION_DATE='2023-11-24T16:18:07.577445+00:00
[2023-11-24, 16:18:11 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-11-24, 16:18:11 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id=load_data, execution_date=20231124T161807, start_date=20231124T161811, end_date=20231124T161811
[2023-11-24, 16:18:11 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-11-24, 16:18:11 UTC] {taskinstance.py:2778} INFO - 0 downstream tasks scheduled from follow-on schedule check
```

FIGURE 2.8: Execution log for Task 2, displaying the task's interactions and data handling.



```
*** Found local files:
***   * /usr/local/airflow/logs/dag_id=process_web_log/run_id=manual__2023-11-24T16:18:07.577445+00:00/task_id=extract_data/attempt=1.log
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: process_web_log.extract_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: process_web_log.extract_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 2
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): extract_data> on 2023-11-24 16:18:07.577445+00:00
[2023-11-24, 16:18:09 UTC] {standard_task_runner.py:57} INFO - Started process 3623 to run task
[2023-11-24, 16:18:09 UTC] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'process_web_log', 'extract_data', 'manual__2023-11-24T16:18:07.577445+00:00', '--job-id', '1414', '--raw', '--subdir', 'DAGS_FOLDER/proce
[2023-11-24, 16:18:09 UTC] {standard_task_runner.py:85} INFO - Job 1414: Subtask extract_data
[2023-11-24, 16:18:09 UTC] {task_command.py:416} INFO - Running <TaskInstance: process_web_log.extract_data manual__2023-11-24T16:18:07.577445+00:00 [running]> on host 0e240741724d
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='extract_data' AIRFLOW_CTX_EXECUTION_DATE='2023-11-24T16:18:07.577445+00
[2023-11-24, 16:18:09 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-11-24, 16:18:09 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id=extract_data, execution_date=20231124T161807, start_date=20231124T161809, end_date=20231124T161809
[2023-11-24, 16:18:09 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-11-24, 16:18:09 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

FIGURE 2.9: Log entries for Task 3, showing the transformation steps and outcomes.

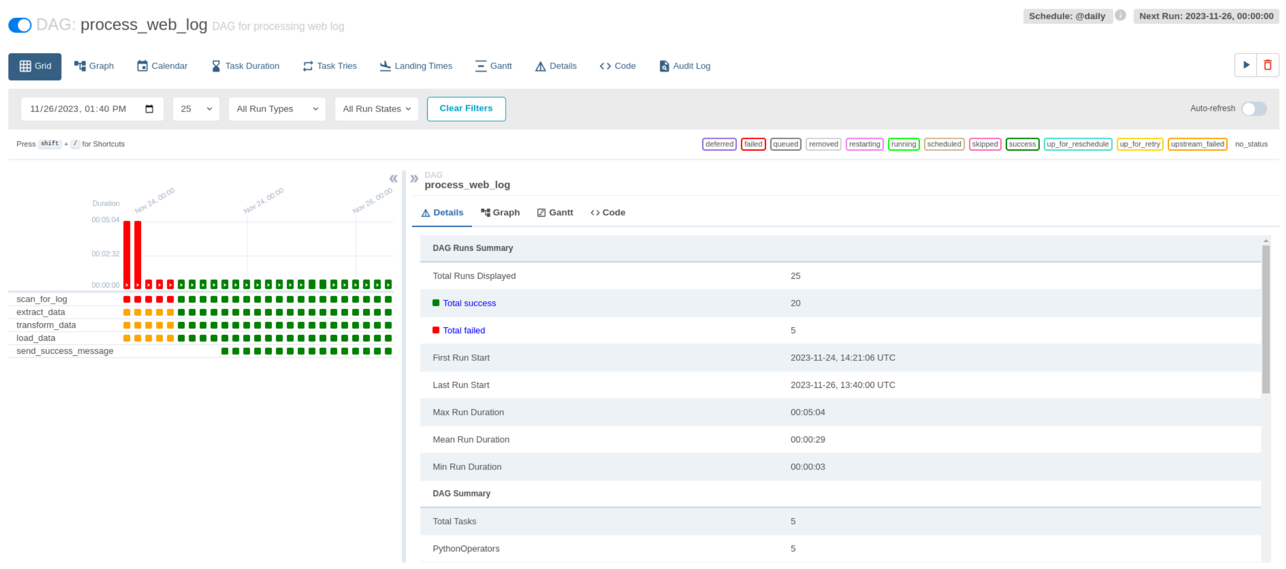


```
*** Found local files:
***   * /usr/local/airflow/logs/dag_id=process_web_log/run_id=manual__2023-11-24T16:18:07.577445+00:00/task_id=transform_data/attempt=1.log
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=non-requeueable deps ti=<TaskInstance: process_web_log.transform_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1159} INFO - Dependencies all met for dep_context=requeueable deps ti=<TaskInstance: process_web_log.transform_data manual__2023-11-24T16:18:07.577445+00:00 [queued]>
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1361} INFO - Starting attempt 1 of 2
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1382} INFO - Executing <Task(PythonOperator): transform_data> on 2023-11-24 16:18:07.577445+00:00
[2023-11-24, 16:18:10 UTC] {standard_task_runner.py:57} INFO - Started process 3625 to run task
[2023-11-24, 16:18:10 UTC] {standard_task_runner.py:84} INFO - Running: ['airflow', 'tasks', 'run', 'process_web_log', 'transform_data', 'manual__2023-11-24T16:18:07.577445+00:00', '--job-id', '1415', '--raw', '--subdir', 'DAGS_FOLDER/pro
[2023-11-24, 16:18:10 UTC] {standard_task_runner.py:85} INFO - Job 1415: Subtask transform_data
[2023-11-24, 16:18:10 UTC] {task_command.py:416} INFO - Running <TaskInstance: process_web_log.transform_data manual__2023-11-24T16:18:07.577445+00:00 [running]> on host 0e240741724d
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1662} INFO - Exporting env vars: AIRFLOW_CTX_DAG_OWNER='airflow' AIRFLOW_CTX_DAG_ID='process_web_log' AIRFLOW_CTX_TASK_ID='transform_data' AIRFLOW_CTX_EXECUTION_DATE='2023-11-24T16:18:07.577445+
[2023-11-24, 16:18:10 UTC] {python.py:194} INFO - Done. Returned value was: None
[2023-11-24, 16:18:10 UTC] {taskinstance.py:1400} INFO - Marking task as SUCCESS. dag_id=process_web_log, task_id=transform_data, execution_date=20231124T161807, start_date=20231124T161810, end_date=20231124T161810
[2023-11-24, 16:18:10 UTC] {local_task_job_runner.py:228} INFO - Task exited with return code 0
[2023-11-24, 16:18:10 UTC] {taskinstance.py:2778} INFO - 1 downstream tasks scheduled from follow-on schedule check
```

FIGURE 2.10: Operational log for Task 4, detailing the data loading and finalization stages.

2.3.3 Workflow Scheduling and Results

Figure 2-6 depicts the scheduling sequence and execution results of the workflow comprising the four tasks. It chronicles the tasks' interdependencies and execution order, which are pivotal for understanding the DAG's orchestration and pinpointing the critical path of the data pipeline.



DAG Runs Summary	
Total Runs Displayed	25
🟩 Total success	20
🟥 Total failed	5
First Run Start	2023-11-24, 14:21:06 UTC
Last Run Start	2023-11-26, 13:40:00 UTC
Max Run Duration	00:05:04
Mean Run Duration	00:00:29
Min Run Duration	00:00:03
DAG Summary	
Total Tasks	5
PythonOperators	5

FIGURE 2.11: Schematic representation of the workflow execution, showcasing the scheduling and results of the four-task sequence.

2.4 Conclusion

Working with Airflow for the `process_web_log` DAG has been a real learning curve. Running it on Linux meant grappling with Docker, which was new to me. I spent hours troubleshooting why Airflow couldn't find my log files, until I realized they needed to be inside the Docker container, not on my local machine.

I also learned how Airflow relies on a database to keep track of everything. When I made changes in Docker and didn't update the database, it led to some confusing errors. It's clear now how important that database is for keeping the workflow running smoothly.

All this has given me a much better grip on how Airflow works, especially the bits about Docker and databases that you don't really think about at first. I'm looking forward to getting even more familiar with Airflow and building more complex and reliable data workflows down the line.

3 Exercise 3 (3 points)

The objective of this exercise was to augment the existing `process_web_log` DAG with an additional task. This new task sends out a notification upon the successful execution of the workflow. The flexibility of the implementation meant we could choose from various messaging platforms. We opted to send the notification to a Slack channel, which is widely used for team communication and can easily integrate with Airflow through webhooks.

3.1 Implementation

To implement the notification feature, a Python function was written to post a message to a specified Slack channel using the channel's webhook URL. This function was then used as a callable in an Airflow PythonOperator task. Below is the code snippet that outlines the implementation of the Slack notification task :

Listing 3.1: Python code for sending a Slack notification.

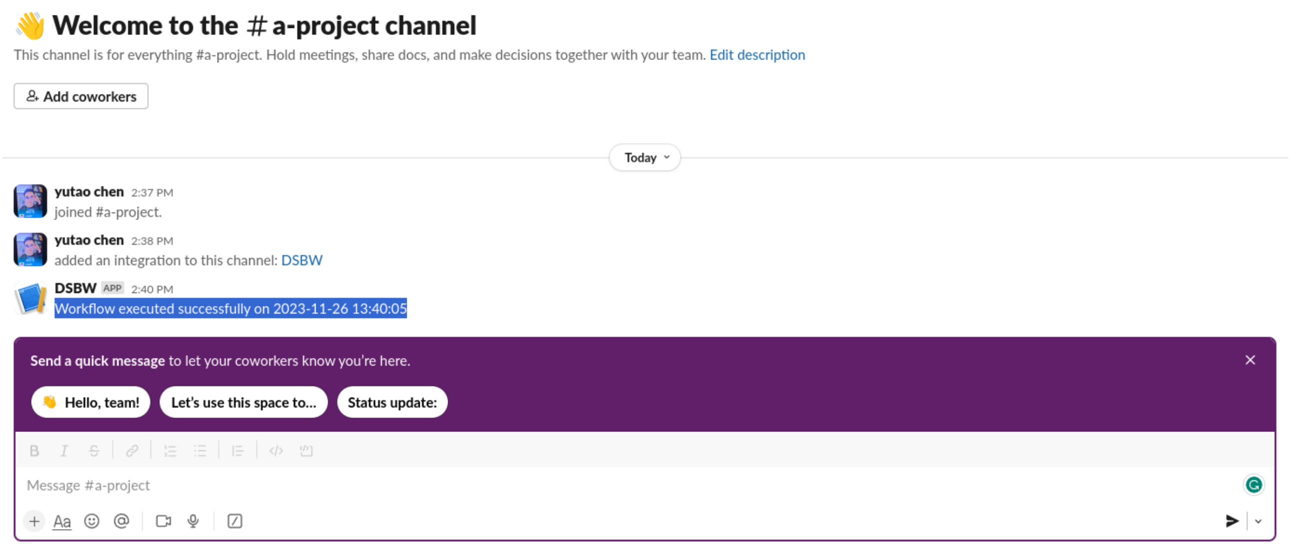
```python
def send_slack_message(**context):
    webhook_url = "https://hooks.slack.com/services/..."
    message = "Workflow executed successfully on " + \
              datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    data = {"text": message}
    response = requests.post(webhook_url, json=data)
    print("Message sent to Slack: Status code", response.
       status_code)

success_message_task = PythonOperator(
    task_id='send_success_message',
    python_callable=send_slack_message,
    provide_context=True,
    dag=dag)
```

This task was placed at the end of the DAG to ensure that the message would only be sent after the successful completion of all preceding tasks.

3.2 Results and Evidence of Execution

Upon triggering the workflow, all tasks, including the new Slack notification task, executed as expected. The message delivered to the Slack channel served as both a confirmation of successful execution and a real-time update to the team monitoring the workflow. The screenshot below captures the message as it appeared in the Slack channel, providing evidence of the task's successful execution.


👋 Welcome to the # a-project channel

This channel is for everything #a-project. Hold meetings, share docs, and make decisions together with your team. Edit description

Add coworkers

Today

yutao chen 2:37 PM
joined #a-project.

yutao chen 2:38 PM
added an integration to this channel: DSBW

DSBW APP 2:40 PM
Workflow executed successfully on 2023-11-26 13:40:05

Send a quick message to let your coworkers know you're here.

FIGURE 3.1: Slack message confirming successful DAG execution.

3.3 Conclusion

The addition of the Slack notification task to the `process_web_log` DAG represents a significant step toward operationalizing our workflow. It not only ensures immediate feedback on the status of the DAG but also enhances the transparency of the data pipeline's performance. This exercise has reinforced the importance of integrating communication tools within data workflows, paving the way for more sophisticated alerting mechanisms in the future.

4
Appendix

Listing 4.1: DAG definition with default arguments.

```python
from airflow import DAG
from airflow.operators.python_operator import PythonOperator
from datetime import datetime, timedelta
import os
import tarfile
import requests

default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2023, 1, 1),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(seconds=1),
}

dag = DAG('process_web_log',
          default_args=default_args,
          description='DAG for processing web log',
          schedule_interval='@daily',
          catchup=False,
          tags=['DSBW'])
```

```python

# Update the path to the new location of your log file inside the Docker container
log_dir = '/usr/local/airflow'
log_file = f'{log_dir}/log.txt'
extracted_data_file = f'{log_dir}/extracted_data.txt'
transformed_data_file = f'{log_dir}/transformed_data.txt'
tar_file = f'{log_dir}/weblog.tar'

def scan_for_log(**context):
    if os.path.isfile(log_file):
        return log_file
    else:
        raise ValueError("log.txt␣not␣found")

def extract_data(**context):
    task_instance = context['ti']
    log_path = task_instance.xcom_pull(task_ids='scan_for_log')

    with open(log_path, 'r') as file, open(extracted_data_file, 'w') as out_file:
        for line in file:
            ip_address = line.split()[0]  # Assuming IP address is the first element in the log line
            out_file.write(ip_address + '\n')

def transform_data(**context):
    with open(extracted_data_file, 'r') as file, open(transformed_data_file, 'w') as out_file:
        for line in file:
            if '198.46.149.143' not in line:
                out_file.write(line)

```

```
def load_data(**context):
    with tarfile.open(tar_file, 'w') as tar:
        tar.add(transformed_data_file, arcname='transformed_data.txt')

def send_slack_message(**context):
    webhook_url = "https://hooks.slack.com/services/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"
    message = "Workflow␣executed␣successfully␣on␣" + datetime.now().strftime("%Y-%m-%d␣%H:%M:%S")
    data = {"text": message}
    response = requests.post(webhook_url, json=data)
    print("Message␣sent␣to␣Slack:␣Status␣code", response.status_code)


# Define tasks
scan_task = PythonOperator(
    task_id='scan_for_log',
    python_callable=scan_for_log,
    provide_context=True,
    dag=dag)

extract_task = PythonOperator(
    task_id='extract_data',
    python_callable=extract_data,
    provide_context=True,
    dag=dag)

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

load_task = PythonOperator(
    task_id='load_data',
    python_callable=load_data,
    provide_context=True,
    dag=dag)

success_message_task = PythonOperator(
    task_id='send_success_message',
    python_callable=send_slack_message,
    provide_context=True,
    dag=dag)


# Set up the workflow
scan_task >> extract_task >> transform_task >> load_task >>
   success_message_task
```