Day 1

- Define a Data Pipeline
- Execute a SQL request with the PostgresOperator
- Execute a Python function with the PythonOperator
- Execute an HTTP request against an API
- Wait for something to happen with Sensors
- Use Hooks to access secret methods
- Exchange data between tasks

create_table >> is_api_available >> extract_user >> process_user >> store_user

- Check Airflow Environment
- Explore Airflow UI
    - Example DAGs
    - Views
- Write Own DAG
    - Operators
        - Action
        - Transfer
        - Sensors
    - Providers
        - Connections to other data pipelines
        - Adding New Functionalities and Interactions
    - Hooks
        - Abstracts Interactions With External Services
        - Storing Using Hooks
    - DAG Scheduling
        - Start
        - Interval
        - End
        - Created a DagRun
    - A Dag is Triggered After
        - startDate / Last Trigger + Interval
        - Runs Between Data Interval Start and Data Interval End
    - Backfilling
        - Catchup & History Call Dag Runs

Component	Description
Web Server	The UI for Airflow, where you can monitor and manage your DAGs, tasks, and logs.
Scheduler	Schedules and triggers DAG runs based on their schedule and dependencies.
Executor	Defines how and on which system tasks are executed.
Queue	Stores tasks that are waiting to be executed.
Metastore	Stores metadata about DAGs, tasks, and other objects in Airflow.
Node	A machine or instance that runs Airflow components.
Worker Node	A node that runs Airflow Celery workers.
Rapid MQ	A message queue used by Celery to communicate between nodes.
Redis	A data structure store used by Airflow for caching and other purposes.
Celery	A distributed task queue used by Airflow to parallelize task execution.
Dag Run Object	A record of a DAG run, including its state (e.g., running, success, failed).
Task Instance	A record of a specific task within a DAG run, including its state and execution information.
Subprocess	A separate process created by the executor to execute a task.

- One Node Architecture
    - Web Server
    - Scheduler
    - Executor
    - Queue
    - Database

- Multi-Node Architecture
    - Node 1
        - Web Server
        - Scheduler
        - Executor
    - Node 2
        - Metastore
        - Queue
    - Celery
    - Kubernetes
    - Airflow Celery Worker
    - Redis
    - Rapid MQ
- Dag
    - Dag Run Object
    - Task Instance
    - Subprocess

Parts of DAG:

1. Default arguments:
    - start_date
    - owner
    - retries
    - retry_delay
    - email_on_failure
    - email_on_retry
2. DAG instantiation:
    - dag_id
    - schedule_interval
    - default_args
3. Operators:
    - task_id
    - bash_command
    - python_callable
    - retries
    - retry_delay
    - email_on_failure
4. Tasks:
    - task_id
    - operator
    - retries
    - retry_delay
    - email_on_failure
5. Dependencies:
    - upstream_task_ids
    - downstream_task_ids
6. Trigger rules:
    - all_success
    - all_failed
    - one_success
    - one_failed
    - none_failed
    - none_skipped
7. DAG configuration:
    - max_active_runs
    - catchup

- default_view
- orientation
- sla_miss_callback

https://airflow.apache.org/docs/apache-airflow/2.5.1/docker-compose.yaml

pip install apache-airflow-providers-postgres==5.4.0
pip install apache-airflow-providers-sqlite

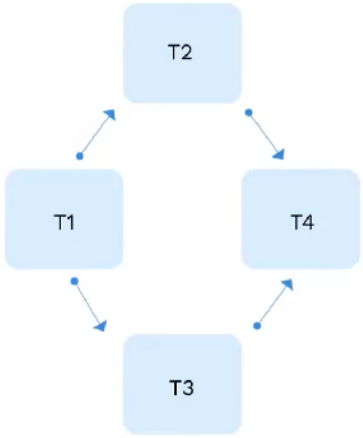
Testing Each Task:
docker compose ps
docker exec -it step_4_examples-airflow-scheduler-1 /bin/bash

Check CSV File:
ls /tmp
docker exec -it step_4_examples-airflow-postgres-1 /bin/bash

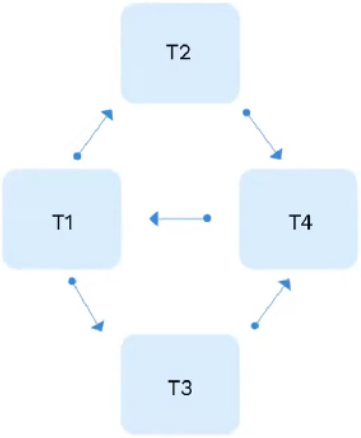
psql -Uairflow
Select * from users

WEB SERVER	→	Flask server running with Gunicorn serving the UI
SCHEDULER	→	Daemon responsible for scheduling jobs
METASTORE	→	A database where all metadata are stored
EXECUTOR	→	Defines **how** tasks are executed
WORKER	→	Process **executing** the tasks, defined by the executor

DAGs



VALID



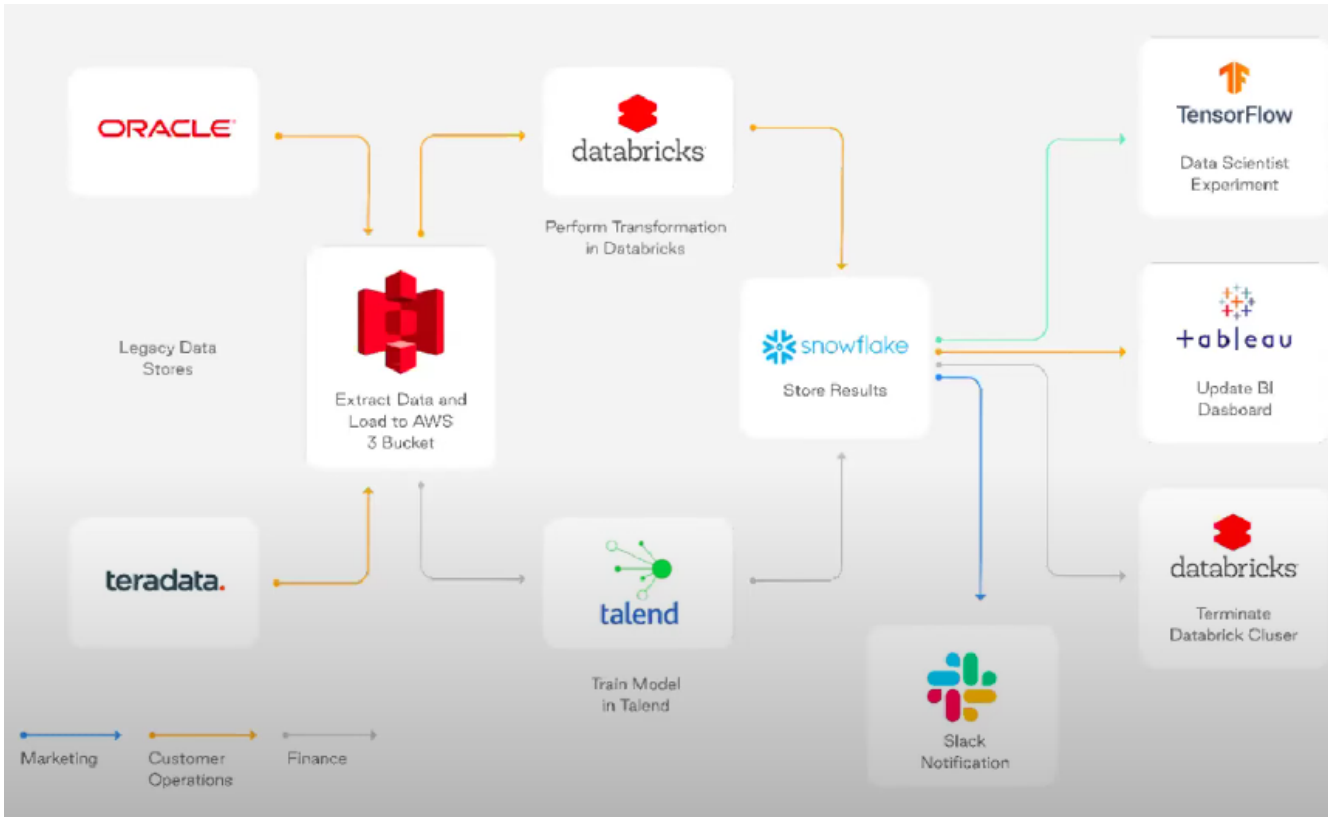
X

INVALID

Operators

```
operator
   file = open("myfile", "r")
   print (f.read() )
```

- Action Operators
- Transfer Operators
- Sensor Operators


ORACLE
databricks
Perform Transformation in Databricks
TensorFlow
Data Scientist Experiment
Legacy Data Stores
Extract Data and Load to AWS 3 Bucket
snowflake
Store Results
+ableau
Update BI Dasboard
teradata.
talend
Train Model in Talend
databricks
Terminate Databrick Cluser
Slack Notification
Marketing
Customer Operations
Finance