

# Airflow Mini-Project (DAG Scheduling) Readme

## Scope

With this mini-project, we utilize Apache Airflow to orchestrate a pipeline:

- Download the daily price data with one minute interval for the two symbols. Each symbol will have a separate task, Task1 (t1) and Task2 (t2), which run independently and in parallel.
- -Save both datasets into CSV files and load them to a data location for query. Each symbol will have a separate task, Task3 (t3) and Task4 (t4), which run independently and in parallel.
- Run your custom query on the downloaded dataset for both symbols, Task5 (t5). Before this step executes, all previous tasks must complete.
- Use Celery Executor in Airflow to run the job.

The source data has the following schema.

Column	Type
date time	STRING
open	DECIMAL
high	DECIMAL (highest price within the time interval)
low	DECIMAL (lowest price within the time interval)
close	DECIMAL (the last price of the time interval)
adj close	DECIMAL
volume	DECIMAL

## Airflow- set up

With this project, we will set up Airflow using docker following the below link:

[Running Airflow in Docker — Airflow Documentation \(apache.org\)](https://airflow.apache.org/docs/apache-airflow/1.10.0/docker-compose.html)

Basically, we will use Docker Compose to create and run multi-container Docker applications, which include:

- airflow-scheduler - The scheduler monitors all tasks and DAGs, then triggers the task instances once their dependencies are complete.
- airflow-webserver - The webserver is available at <http://localhost:8080>.
- airflow-worker - The worker that executes the tasks given by the scheduler.
- airflow-init - The initialization service.
- postgres - The database.
- redis - The redis - broker that forwards messages from scheduler to worker.

For more details of set up the system, please read 'Airflow Mini-Project System Setup Steps.pdf'

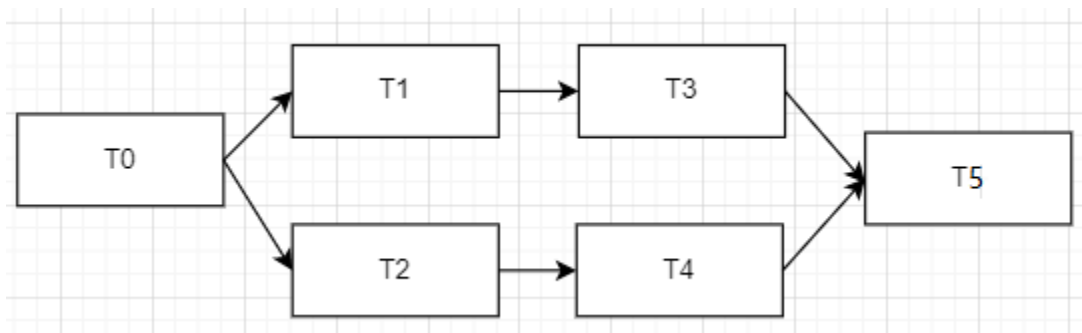
## DAG

A DAG (Directed Acyclic Graph) is the core concept of Airflow, collecting Tasks together, organized with dependencies and relationships to say how they should run.

In this project, we define a DAG named 'dag\_marketvol'. It defines six tasks and dictates the order in which they have to run, and which tasks depend on what others. The DAG is defined in python script named 'dag\_marketvol.py'.

### Task definition

Task dependencies::



Task detailst:

Task-id	Type & script	Description
T0	(BasOperator) cr_folder.sh	To create a temporary directory for current day's data download (/tmp/data/20220617)
T1	(PythonOperator) Download_stock (symbol='AAPL')	Download data file for symbol 'AAPL' and save to the temp folder created at step T0 (i.e,/tmp/data/20220617/AAPL_20220617.csv).
T2	(PythonOperator) Download_stock (symbol='TSLA')	Download data file for symbol 'TSLA' and save to the temp folder (for example, /tmp/data/20220617/TSLA_20220617.csv)
T3	(BasOperator) mv_file1.sh	Move the download 'AAPL' data file (AAPL_20220617.csv) to data location '/tmp/query/'
T4	(BasOperator) mv_file2.sh	Move the download 'TSLA' data file ( TSLAL_20220617.csv) to data location '/tmp/query/'
T5	(PythonOperator) get_stock_avg	Query both files that are located in the query folder ( '/tmp/query') using Pandas library

Python method for downloading

Method	Functions
download_stock (symbol)	Download the data files of the stock of the given symbol

Python methods for Query

Method	Functions
read_stock((file_folder, symbol)	Read data file of the given symbol from the specified local utilizing Pandas library. For example, it reads data from file /tmp/query/AAPL_20220618.csv and return dataframe
get_stock_avg(file_folder )	Read both AAPL, TSLA data files of current date (such as AAPL_20220618.csv TSLA_20220618.csv) from the given location and display the counts and average 'open' value.

## DAG Scheduling

### Requirement:

- Start time and date: 6 PM on the current date.
- Job interval: runs once daily.
- Only runs on weekdays (Mon-Fri).
- If failed: retry twice with a 5-minute interval.

### Implementation:

We define the DAG as below to meet the requirements:

```
default_args = {
    'owner': 'airflow',
    'depends_on_past': False,
    'start_date': datetime(2022, 6, 18),
    'retries': 2,
    'retry_delay': timedelta(minutes=5)
}

dag = DAG(
    dag_id=...,
    default_args=default_args,
    description=...,
    schedule_interval='0 18 * * 1-5',
)
```