Airflow - Theoretical Assessment

Section A – Basics

1. **What is Apache Airflow and why is it used?**

Apache Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. It is used to orchestrate complex data pipelines and batch jobs, ensuring tasks run in a specific order, handling dependencies, and providing a UI for monitoring.

2. **Define a DAG. What does each part of the acronym stand for?**

A DAG (Directed Acyclic Graph) is the entire workflow definition. It's a collection of all tasks organized by their relationships and dependencies.

- Directed: Connections between tasks have a defined direction.
- Acyclic: There are no loops; a task cannot lead back to a preceding task.
- Graph: A structure of vertices (tasks) and edges (dependencies).

3. **Explain the difference between a DAG and a Task.**

A DAG is the complete workflow structure and configuration. A Task is a single, defined unit of work within that DAG (e.g., run a script, execute a SQL query). A DAG is composed of one or more Tasks.

4. **Why should workflows be "Directed Acyclic Graphs" in Airflow?**

Workflows must be DAGs to ensure a clear, unambiguous execution path and guarantee termination. The Directed nature defines the order, and the Acyclic nature prevents infinite loops and deadlocks, ensuring the workflow eventually completes.

Section B – Core Concepts

1. **Describe the role of the following Airflow components:**

- Webserver: Serves the Airflow User Interface (UI) for visualizing DAGs, monitoring run status, and viewing logs.
- Scheduler: The brain of Airflow. It monitors all DAGs, decides which tasks need to run, handles dependencies, and submits tasks for execution.
- Metadata Database: Stores the state of Airflow, including task statuses, DAG configurations, variables, and historical execution records.

2. **What is the purpose of the airflow db init command?**

The command is used to initialize the Airflow Metadata Database for the first time. It creates all the necessary tables and schema required for Airflow to store its internal state and operate.

3. **What is the significance of start_date and schedule_interval in a DAG?**

- start_date: Defines the earliest time from which the scheduler should begin considering runs.
- schedule_interval: Defines how often the DAG should run (e.g., daily, hourly). The first run will occur after the start_date plus one schedule_interval.

4. **What does catchup=False do, and when would you use it?**

catchup=False tells the Scheduler not to run missed schedules between the defined start_date and the current time (when the DAG is first enabled). You use it when your DAG processes current data and running historical intervals would be pointless or harmful.

Section C – Operators & Execution

1. **What is an Operator? Give two examples.**

An Operator is a template for a task that defines a specific action to be performed. They are the building blocks of Tasks.

Example:

- BashOperator: Executes a shell command or script.
- PythonOperator: Executes a Python function.

2. **How does Airflow handle task failures and retries?**

When a task fails, Airflow marks its state as "failed." If the task is configured with a number of retries, the Scheduler will automatically re-attempt to run the task after a specified retry_delay until the maximum retries are exhausted.

3. **What is XCom and how is it useful?**

XCom stands for "cross-communication." It is a mechanism that allows Tasks within the same DAG run to exchange small amounts of metadata (e.g., file paths, IDs). It is useful for passing an output from one task to be used as an input for a subsequent task.

4. **Explain the difference between BashOperator and PythonOperator .**

The BashOperator is used to execute shell commands or scripts (e.g., system utilities, existing shell logic). The PythonOperator is used to execute a native Python function (e.g., data manipulation, custom API calls).

Section D – Real-World Use

1. **Give one real-world example where Airflow can be used for ETL.**

Airflow can orchestrate a nightly data warehouse ETL pipeline: It runs tasks to Extract (pull data from a source database), Transform (clean and aggregate data using Spark or a Python script), and Load (insert the processed data into a data warehouse table like Snowflake or Redshift).

2. **Why is it recommended to keep DAG scripts lightweight and avoid heavy computations inside them?**

DAG scripts are constantly parsed and evaluated by the Airflow Scheduler. Heavy computations in the DAG file will slow down the Scheduler, impacting performance across all DAGs. The DAG file should only define the workflow; the heavy work should be delegated to the Tasks, which run on the dedicated Workers.

3. **Why should every DAG have a unique dag_id?**

The dag_id is the primary unique identifier for the workflow in the Metadata Database. Airflow uses it to track the state, run history, and configuration of that specific DAG. Non-unique IDs would cause conflicts and prevent Airflow from correctly managing the workflows.

4. **How does Airflow ensure workflows run in the correct order?**

Airflow ensures order by strictly enforcing the dependencies defined between tasks (e.g., task_A >> task_B). The Scheduler will only submit a task for execution once all of its upstream dependent tasks have successfully completed.