Apache Airflow – Theoretical Assessment Answers

Section A – Basics

1. **What is Apache Airflow and why is it used?**
Apache Airflow is an open-source platform for programmatically authoring, scheduling, and monitoring workflows. It is used to automate data pipelines, manage dependencies, and ensure reliable execution of complex tasks.

2. **Define a DAG. What does each part of the acronym stand for?**
DAG stands for Directed Acyclic Graph. It represents a collection of tasks with defined dependencies where:
- Directed: The workflow moves in a specific direction.
- Acyclic: No task can depend on itself (no loops).
- Graph: The structure connecting tasks.

3. **Explain the difference between a DAG and a Task.**
A DAG is the overall workflow (a collection of tasks and their relationships). A Task is a single unit of work or operation within that workflow.

4. **Why should workflows be "Directed Acyclic Graphs" in Airflow?**
Because tasks must have a clear, non-circular order of execution. Cycles would cause infinite loops and make dependency resolution impossible.

Section B – Core Concepts

1. **Describe the role of the following Airflow components:**
- **Webserver:** Provides the user interface for viewing DAGs, logs, and task status.
- **Scheduler:** Monitors DAG definitions and schedules tasks according to dependencies and intervals.
- **Metadata Database:** Stores information about DAG runs, task instances, and configurations.

2. **What is the purpose of the airflow db init command?**
It initializes the metadata database by creating all necessary Airflow tables and default records.

3. **What is the significance of start_date and schedule_interval in a DAG?**
- **start_date:** Specifies when the DAG should start running.
- **schedule_interval:** Defines how often the DAG should be triggered (e.g., daily, hourly).

4. **What does catchup=False do, and when would you use it?**
It tells Airflow not to run missed DAG runs from the past. Used when only the most recent run matters (e.g., real-time data processing).

Section C – Operators & Execution

1. **What is an Operator? Give two examples.**
An Operator defines a single task in a DAG (what to do). Examples: **BashOperator** (runs bash commands) and **PythonOperator** (runs Python functions).

2. **How does Airflow handle task failures and retries?**
Each task can specify retry settings (number of retries, delay between retries). If a task fails, Airflow automatically retries it based on those settings.

3. **What is XCom and how is it useful?**
XCom (Cross-Communication) allows tasks to share small data (messages or results). It enables

passing information between dependent tasks.

4. **Explain the difference between BashOperator and PythonOperator.**
- **BashOperator:** Executes shell commands or scripts.
- **PythonOperator:** Executes a Python function directly within Airflow.

Section D – Real-World Use

1. **Give one real-world example where Airflow can be used for ETL.**
Airflow can orchestrate an ETL pipeline that extracts data from APIs, transforms it with Spark, and loads it into a data warehouse like Snowflake or BigQuery.

2. **Why is it recommended to keep DAG scripts lightweight and avoid heavy computations inside them?**
DAG files are parsed frequently by the scheduler. Heavy computations slow parsing and can block scheduling. Actual data processing should be delegated to Operators.

3. **Why should every DAG have a unique dag_id?**
Each DAG must have a unique identifier so Airflow can track its metadata, runs, and states independently.

4. **How does Airflow ensure workflows run in the correct order?**
By using task dependencies (set via set_upstream / set_downstream or bitshift operators). Airflow schedules and executes tasks based on these defined relationships.