**APACHE AIRFLOW ASSESSMENTS**

**ASSESSMENT - 1:**

1. What is Apache Airflow, and how does it work?

Apache Airflow is an open-source tool for authoring, scheduling, and monitoring workflows. Workflows are defined as DAGs (Directed Acyclic Graphs) in Python, where tasks are connected by dependencies. The scheduler triggers tasks according to schedules, and the executor runs them locally or in distributed setups. Execution metadata is stored in a database, while the web UI provides visualization and monitoring.

1. Where does Airflow fit in modern data engineering workflows?

Airflow acts as a central orchestration tool for ETL pipelines, data ingestion, transformations, and analytics workflows. It ensures tasks run reliably with dependency management, retries, and monitoring. It integrates with databases, APIs, cloud storage, and ML pipelines, making it ideal for enterprise-scale, reproducible, and auditable workflows.

1. How is Airflow different from traditional schedulers or other tools like

Prefect or Luigi?

Unlike Windows Task Scheduler, which only run tasks on a fixed schedule, Airflow manages task dependencies, retries, and monitoring. Compared to Luigi or Prefect, Airflow provides enterprise-grade scalability, rich UI, and flexible executors. Luigi is simpler but less extensible, while Prefect focuses on real-time workflows, whereas Airflow excels in batch-oriented orchestration.

1. What are the key components (e.g., DAGs, operators, scheduler, executor) and

how do they interact?

**DAGs:** Define workflow and task dependencies.

**Operators:** Define tasks (Python, SQL, Bash, etc.)

**Scheduler:** Triggers tasks based on DAGs and schedules.

**Executor:** Runs tasks locally or distributed.

**Metadata DB:** Stores task status, logs, and history.

**Web UI:** Visualizes execution and helps with monitoring.

1. Based on your learning, where do you see Airflow being useful in real-time

enterprise or product scenarios?

Airflow is ideal for automating batch ETL, synchronizing multi-source data, updating data warehouses, and ML pipelines. Enterprises use it to maintain audit trails, enforce SLAs, and trigger downstream workflows, while product teams can automate data refreshes, report generation, or feature computation, ensuring reliable and maintainable data operations.

**ASSESSMENT – 2:**

1. What is the role of DAGs in monitoring and auditing pipelines?

In Airflow, DAGs (Directed Acyclic Graphs) define the workflow structure and task dependencies, making it easy to monitor the progress of pipelines. Each DAG run is logged with task start/end times, success or failure states, and execution details. This metadata provides a complete audit trail, enabling teams to trace failures, debug issues, and maintain compliance.

1. How can Airflow be adapted for event-driven workflows (e.g., reacting to

external changes)?

While Airflow is traditionally schedule-based, it can handle event-driven workflows using sensors or external triggers. For example, a FileSensor can watch for new files in storage, or a DAG can be triggered via API calls when upstream processes complete. This allows pipelines to react dynamically to external events instead of relying solely on fixed schedules.

1. Compare Airflow with cron-based scripting, with at least 2 advantages.

Unlike cron, which only runs scripts at fixed times, Airflow provides dependency management, retries, and visibility. Two key advantages over cron are: (1) workflow awareness, ensuring tasks run in the correct order and downstream tasks wait for upstream success, and (2) centralized monitoring and logging, with retries and alerting built-in, unlike cron’s minimal logging.

1. How can Airflow be integrated with external logging/alerting systems?

Airflow can connect to external monitoring and alerting systems such as Slack, email, webhooks, ELK Stack, Splunk, or Prometheus. Task failures, SLA misses, or other events can trigger notifications, while logs and metrics can be forwarded to centralized platforms, enabling proactive pipeline monitoring and incident response.