Part 2: Analytical Brief (Conceptual)

Write a short internal technical note (max 500 words) that answers:

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

DAGs provide a declarative, code-defined blueprint of monitoring and audit workflows. Each task encapsulates a check (schema sanity, data freshness, thresholds, SLAs), and dependencies ensure the correct order (e.g., pull → validate → log → notify). Because Airflow tracks task state, retries, and logs centrally, audit evidence (inputs, outputs, validations) becomes reproducible and observable, which is essential for compliance and incident root-cause analysis.

How can Airflow be adapted for event-driven workflows (e.g., reacting to external changes)?

Although Airflow is traditionally schedule-driven, it can be made event-responsive in several ways: trigger DAGs via an API/Webhook when upstream systems change; use Sensors (e.g., file, S3, external task) to block until an event occurs; or pair Airflow with a message bus (e.g., Kafka, Pub/Sub) where a lightweight listener triggers a DAG run upon an event. On Kubernetes, ExternalTaskSensor and deferrable operators reduce resource usage while waiting for events. Combining short schedules (e.g., @hourly) with idempotent, incremental tasks also approximates near–real-time response when true streaming isn't required.

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

1. Dependency awareness & retries: Airflow models task dependencies explicitly and handles retries with backoff and alerting. Cron only fires commands at times; it has no concept of upstream/downstream success, causing brittle chains of shell scripts.
2. Observability & lineage: Airflow's UI, logs, and metadata capture task runs, parameters, and outcomes; operators integrate with warehouses and clouds, improving lineage and auditability. Cron lacks centralized monitoring, making failure triage and compliance evidence harder. Additional advantages include scheduling semantics (catchup, backfills), templating with Jinja, and a rich provider ecosystem for databases, clouds, and ML tools.

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

Airflow can emit logs to external systems (e.g., S3, GCS, Elasticsearch) by configuring remote logging, enabling centralized dashboards and retention. Alerts can be sent via EmailOperator, Slack/MS Teams callbacks, PagerDuty, Opsgenie, or custom on-failure callbacks per task or DAG. For SIEM/SOAR integration, push structured results (like the /tmp/audit_result.json) to a log pipeline (Fluent Bit/Filebeat → Elasticsearch/Splunk) or post events to a webhook endpoint. Using Airflow callbacks (on_failure_callback, on_success_callback) lets you guarantee that audit outcomes generate notifications and tickets automatically.