### Databricks Jobs Workflow

**Step 1: Data Ingestion**

* Instead of uploading CSVs manually to DBFS, use **Auto Loader** with cloud storage (ADLS, S3, GCS) for scalable, incremental ingestion.
* Maintain schema evolution to handle changes in IoT data.

**Step 2: Notebook Modularization**

* Consolidate notebooks where possible (e.g., ingestion + transformation can be handled in a single pipeline).
* Use **Delta Lake** for raw, cleaned, and curated layers instead of plain CSV/JSON, ensuring ACID compliance.
* Leverage parameterization at the job level rather than widget-based single file execution.

**Step 3: Data Quality Validation**

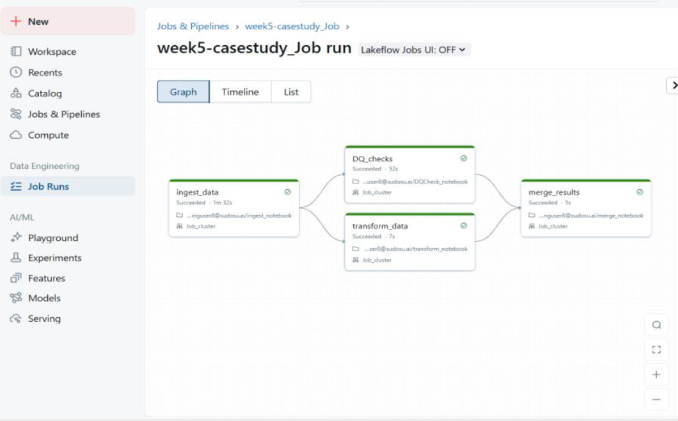
* Integrate **PyDeequ checks directly into the pipeline**, validating during transformation.
* Store validation results in a structured Delta table rather than JSON, making them queryable for reporting and monitoring.
* Automate alerting for failures with Databricks SQL alerts or monitoring dashboards.

**Step 4: Job Orchestration**

* Replace manual fan-out/fan-in with **Task Orchestration** in Databricks Jobs (task dependencies, retries, conditional execution).
* Where transformations are independent, leverage **parallel tasks** to reduce runtime.
* For automation, use **Databricks Workflows with GitHub Actions or Azure DevOps** for CI/CD, ensuring libraries and cluster configurations are version-controlled.

**Step 5: Observability & Optimization**

* Enable **job monitoring, retry logic, and cluster autoscaling** for efficient resource usage.
* Use **DBFS only for temp/debug data**; production data should live in cloud object storage.
* Add **Delta Live Tables (DLT)** or structured streaming if continuous ingestion and quality checks are needed.



ADF pipeline And Data Flow Structure:

