Introduction & Architecture

Integration 3.0 is an in-house ETL solution that provides a visual, low-code environment for building business logic and integrating external systems with the o9 ecosystem.

- It replaces traditional SSIS-based SQL Server execution with massive parallel processing via Spark on Kubernetes clusters.
- Primary storage utilizes Cloud Data Lakes (AWS S3, Google Cloud Storage, Azure Data Lake Storage) rather than SQL servers.
- Parquet is the standard storage format, optimized for reading and storage via its columnar structure.
- Airflow is integrated into the backend to generate and orchestrate Data Directed Acyclic Graphs (DAGs).

Data Lake Architecture

Integration 3.0 organizes the o9 Data Lake into distinct zones to track data progression.

Zone	Description
Raw Layer	Stores inbound data exactly as it arrives from source systems without modification.
Staging Layer	Processes data strictly by converting the file format to Parquet for efficiency, without cleaning it.
Curate Layer	Stores refined, trusted data that has been cleaned and transformed, ready for GraphCube integration.

Core Pipeline Components

Pipelines are constructed using three foundational components.

- **DataStores:** Serve as the primary, parameterizable input/output sources (e.g., Cloud Stores, SFTP, Snowflake, BigQuery).
- **DataNodes:** Nested within DataStores, representing a single data entity such as a SQL table, flat file, or GraphCube dimension.
- **Transformers:** Connect DataNodes to transform and move data using compute actions (PySpark), operational tasks (API execution, alerts), or file transfers.

Scheduling & Execution

- Pipelines support On-Demand execution, Time-Based scheduling (via Cron expressions), and Event-Based scheduling that triggers on an OR condition if at least one upstream pipeline completes.
- Pipelines can be triggered directly from UI Action Buttons, passing runtime parameters to the execution.
- Users can view the exact next scheduled execution time, pause or resume executions natively, and compare historical performance using Runtime Graphs.
- Batch executions group pipeline metrics via a Batch ID and can be visually tracked in real-time through a node-based flowchart in the Scheduling Info tab.

Parameterization & Profiling

- **Runtime ParamSets:** Users can substitute values dynamically during execution and utilize a "Param Set Where Used" tool to trace affected pipelines.
- **Workspace Parameters:** Defined globally, allowing reuse across multiple pipelines via the {{o9userlv}} alias.
- **Dynamic Updates:** Parameter values can be programmatically updated mid-execution for downstream use.
- **Spark T-Shirt Sizes:** Default profiles (Small, Medium, Large, XLarge) automatically allocate appropriate driver/executor memory and cores, eliminating the need for manual tuning.
- **Security:** Sensitive parameters are encrypted automatically, while non-sensitive parameters remain accessible in Tenant System Settings.

Monitoring, Alerts, & Troubleshooting

- **Health Checks:** The "Monitor Services" view tracks real-time status for backend components like MongoDB, Redis, and Airflow.
- **LLM Log Summary:** An AI-powered summarizer reads task-level logs to provide plain-text explanations and resolution guidance for pipeline failures.
- **SLA Alerts:** The system monitors batch job health and sends proactive emails for delayed starts, long runs, or failures, featuring deep links directly to the error logs.
- **Rerun & Recovery:** Failed pipelines feature a direct "Rerun" option to resume execution from the exact point of failure.
- **Spark Profiling:** Users can utilize the Spark ELK dashboard to visualize memory and CPU usage of executors over time.
- **Troubleshooting Tip:** If system parameters go missing after a migration, users should create a new DataStore to trigger the system to automatically regenerate them.

Advanced Features & Tasks

- **Agentic EDA:** AI-powered Exploratory Data Analysis lets users ask natural language questions about flat files to generate instant statistics and visualizations.
- **JupyterHub Integration:** Developers can launch JupyterHub to interactively write PySpark code and seamlessly sync the validated code back to the pipeline's transformer.
- **Multi-Tenant Data Sharing:** Linked Data Nodes allow data to be shared directly within the o9 Data Lake across environments, facilitating inter-tenant pipeline dependencies.
- **Delta Share & Databricks:** Allows direct data ingress via Delta Share with query filters, and enables o9 Delta tables to be registered externally in a customer's Databricks workspace via Unity Catalog.
- **Config 2.0 (C2):** Components are natively modularized for Git syncing using Integration Common and Integration Regular blocks.
- **File Management:** The Int3 File Manager operates as a UI-native cloud file browser, permitting folder navigation, previews up to 10MB, and multi-file uploads.