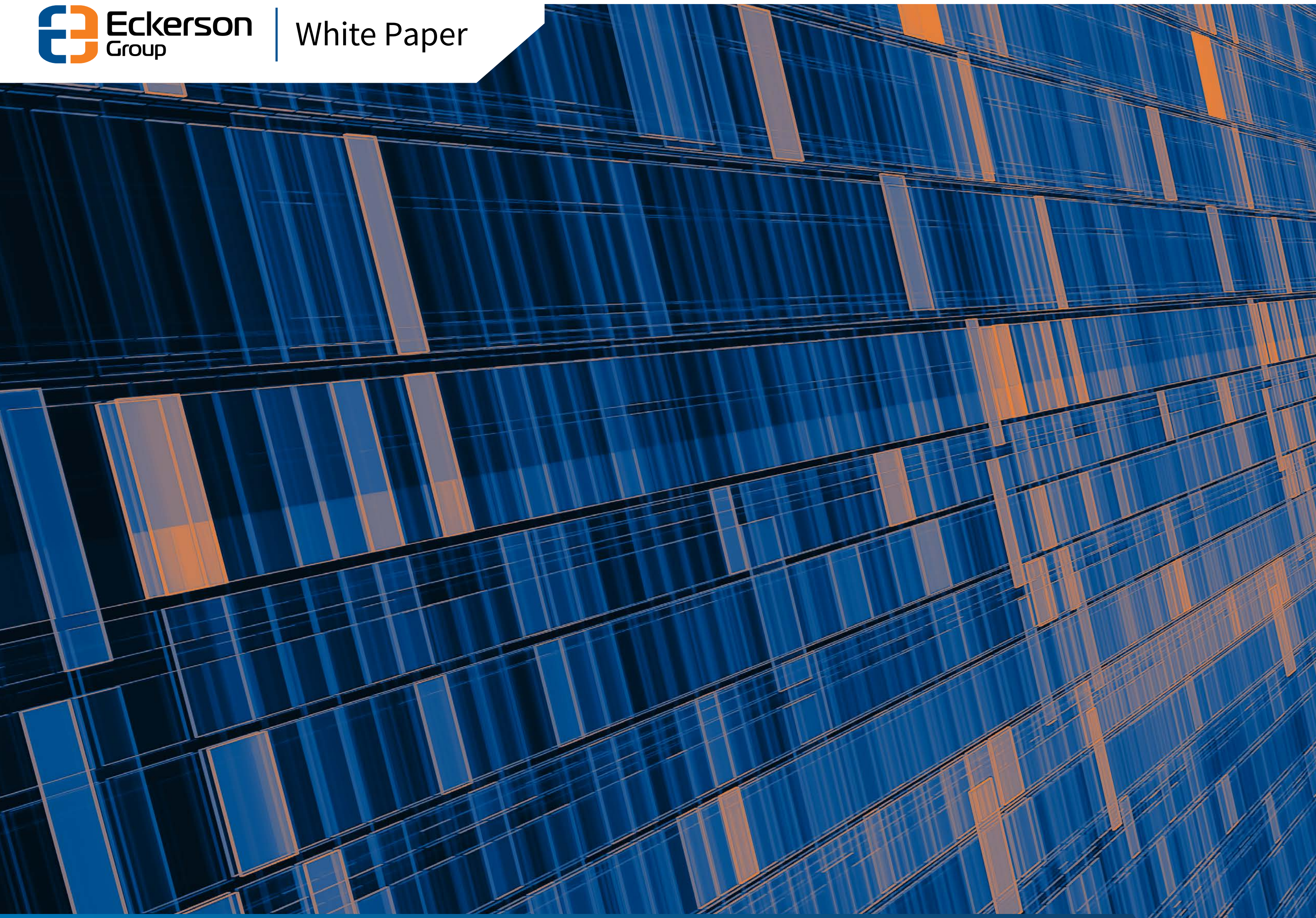
Eckerson Group | White Paper



Data Pipeline Orchestration

Modernizing Workflows to Drive Success with AI/ML

BY SUMIT PAL

JANUARY 2024

RESEARCH SPONSORED BY

databricks

About the Author

Sumit Pal was a VP analyst in data management and analytics for Gartner, where he advised CTOs, CDOs, CDAOs, enterprise architects, and data architects on data strategy, data architectures, implementation, and choosing tools, frameworks, and vendors for building data platforms and end-to-end data-driven systems. Sumit has advised organizations and CDOs with guidance, insights, and analysis on best practices to solve data challenges.

Sumit has more than 30 years of experience in the software and data industry spanning companies from start-ups to enterprises. His book is SQL on Big Data: Technology, Architecture, and Innovation (Apress, 2016), and he has developed a massive open online course on big data. Sumit has also hiked to Mt. Everest Base Camp.

About Eckerson Group

Eckerson Group is a global research, consulting, and advisory firm that helps organizations get more value from data. Our experts think critically, write clearly, and present persuasively about data analytics. They specialize in data strategy, data architecture, self-service analytics, master data management, data governance, and data science. Organizations rely on us to demystify data and analytics and develop business-driven strategies that harness the power of data. **Learn what Eckerson Group can do for you!**

Eckerson Group
RESEARCH CONSULTING ADVISORY
GET • MORE • VALUE • FROM • YOUR • DATA

About This Report

Research for this report comes from discussions with Databricks and various practitioners. This report is sponsored by Databricks, which has exclusive permission to syndicate its content.

Table of Contents

Executive Summary

Data will decide who wins the AI race. The problem is that data engineering teams are drowning in technical and organizational complexity that slows them down. The disaggregated data stack is causing organizations to duct-tape end-to-end data solutions. Organizations need to repeatedly combine data from diverse and disparate data sources and make it available for downstream applications, data science projects, and AI systems. Data pipeline orchestration helps standardize and simplify these workflows to make AI and ML successful.

Key Takeaways

Data pipeline orchestration, also known as data orchestration, has become a critical success factor for data engineering teams that support AI/ML initiatives.

> *Many organizations do not sufficiently address the critical last mile of data pipeline management. This last mile centers on data orchestration—i.e., scheduling and optimization of pipelines and the larger workflows they help drive.*

> *For most organizations, data orchestration is in the early to mid-stage of maturity.*

> *Current data orchestration tools place a lot of responsibility on data teams to build, manage and maintain pipelines as they are based on imperative models.*

> *Data orchestration is a fast-moving space with new tools emerging to overcome challenges with prior generation tools.*

> *Next-generation tools add declarative capabilities to streamline self-service for pipeline development, eliminating long implementation time and easing maintenance across the data lifecycle.*

> *These emergent tools also simplify the process of building, tracking, deploying, and managing data pipelines. In addition, they help integrate pipelines with applications and tools and manage resources efficiently.*

> *Cloud vendors lack native data orchestration and workflow management tools. Most of them are building managed services on top of open-source tools.*

> *Most data orchestration tools have insufficient support for streaming workloads.*

Recommendations

Data engineers and DataOps teams should:

> *Select data orchestration tools early in the project. Do not leave this decision to the DataOps team to consider during deployment. Data orchestration tool selection should be formally planned with a proof of concept, tested, and reviewed by all stakeholders.*

> *Leverage data orchestration tools only for data orchestration. Don’t run expensive, compute-heavy tasks within the data orchestration tools.*

> *Avoid manual and ad hoc approaches for data orchestration with scripts and non-standard approaches.*

> *Carefully evaluate cloud-native solutions, especially those that are based on open-source software, as they may be too restrictive for certain use cases.*

Introduction

Data teams deliver data assets and products. To make this happen, data engineers manage a spectrum of tasks that includes infrastructure provisioning and setting up and managing data connections for ingestion. They also must manage data security, privacy, storage, processing, and testing. Because technologies depend on each other in a variety of ways, data, DevOps, and DataOps engineers must manage a myriad of tools, frameworks, and services as part of an end-to-end ecosystem. Defining, testing, managing, troubleshooting, and deploying data pipelines with complex workflows is incredibly challenging and time-consuming—and getting worse.

Definition. Data pipeline orchestration (referred to here as data orchestration) is the process of managing the complex interoperability of technologies in a modern data stack. Data orchestration provides the necessary coordination, scheduling, and automation to guarantee reproducibility and reusability of complex interdependent data flows. When designed and implemented well, it powers BI and AI insights as well as data-driven products and solutions.The success of being data-driven depends on DataOps, data orchestration, and automation.

Data pipeline orchestration is the process of composing, building, and streamlining complex data pipelines, plus reliably and efficiently releasing data artifacts to production. It enables automation and accelerates data flow from diverse sources for operational and analytics initiatives to drive business value.

The success of becoming data-driven depends on DataOps, data orchestration, and automation.

Old approach. Data orchestration was traditionally addressed with homegrown scripted tools and a lot of manual effort—with little or no automation, monitoring, or recoverability. This approach did not scale for midsize and large organizations whose data pipelines continue to grow. Cumbersome point-to-point Integrations and custom worked only for smaller, slow-changing environments.

Modern data pipelines. The modern data stack is a disaggregation of rich tools, frameworks, and libraries, rather than a unified platform for building end-to-end, data-driven systems. In this fragmented ecosystem, data teams stitch together diverse tools and services to build functioning data platforms, solutions, and services.

Data pipeline orchestration is the process of composing, building, and streamlining complex data pipelines, plus reliably and efficiently releasing data artifacts to production.

Many organizations have hundreds of interdependent data pipelines that deliver heterogeneous data across diverse data sources, targets, and locations. There is inherent complexity in distributed multi-modal data pipelines, creating challenges for data managers:

> How do we keep track of data sets?

> How do we ensure that data sets are available at the time data pipelines execute?

Many modern data pipelines run on a distributed architecture with dependent subcomponents spanning multiple servers and virtual machines. Production data pipelines are complex, with forks and dependencies and different mixes of trigger logic for the tasks. There are innumerable handshakes across these components, with multiple failure points from networks to servers.

Many organizations have hundreds of interdependent data pipelines that deliver heterogeneous data across diverse data sources, targets, and locations.

When building such complex interdependent pipelines, data teams need to guarantee that pipelines execute tasks as intended. Proper exception, error, and flow handling is critical. This becomes more important for large data sets with demanding service-level agreements (SLAs). Data orchestration coordinates the different components and ensures pipeline resiliency, handling failover and recoverability from the failure point.

Market Evolution

Data orchestration products have evolved to keep pace with rapid changes in the data management space. Today we are in the fifth generation of data orchestration, with tools that span capabilities from the third generation onward.

> **First-generation data orchestration tools** were ad hoc solutions based on cron jobs (1990s) and scripts.

> **Second-generation tools** such as **SSIS** (SQL Server Integration Services) (2005) and Oracle OWB offered a graphical interface to reduce scripting.

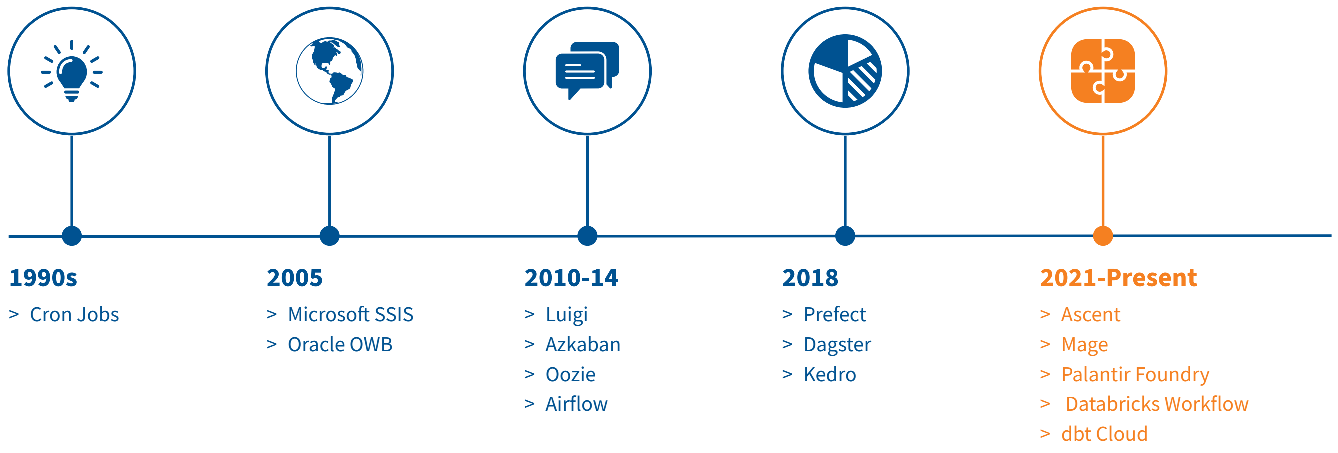
> **Third-generation tools** such as Airflow (2014), Luigi (2012), and Oozie (2010) used standard code-centric mechanisms to build workflows. They focused on task-driven approaches to orchestration, decoupling task management from execution.

> **Fourth-generation tools,** based on Python and SQL-like tools such as Prefect (2018), Kedro, Dagster (2019), and dbt, use declarative and functional approaches to classify functions as tasks and create DAGs (directed acyclic graphs). These tools are primarily data-driven, with inherent knowledge of data types that empower testing, version control, state management, and recoverability.

> **Fifth-generation systems** are fully managed declarative pipelines with minimal code to build end-to-end pipelines, such as Ascent, Palantir Foundry, and data lake–specific solutions such as Databricks Workflows.

Figure 1 shows the evolution of the data orchestration market over the last few decades.

Figure 1. Data Orchestration Market Evolution



The near future. Going forward, vendors will add capabilities for automated recoverability from failures at the software and hardware level. They are incorporating assistants based on large language models, enabling domain experts to self-serve their data flows declaratively (as in SQL, where one declares the results needed, not the steps to achieve those results). They are also becoming Kubernetes native and transitioning to becoming fully managed platforms. These next-generation orchestrators support streaming pipelines and include serverless capabilities that help relieve data and DevOps teams from provisioning and managing infrastructure.

We are now in the fifth generation of data orchestration, with tools spanning capabilities from the third generation onward.

Challenges

Data orchestration tools and frameworks need to overcome several challenges to become effective. Some of these challenges include:

- **Complex workflows.** Modern data orchestration tools (i.e., data orchestrators) need to reduce the complexity of interconnecting diverse components of data pipelines across a spectrum of tasks and jobs. To achieve this, data orchestrators need to provide the right level of abstraction for personas who build pipelines with UI, CLI, or APIs with self-service.
- **Reliability.** In data-driven organizations, data orchestration becomes the critical part of the data supply chain. Data orchestrators need to ensure data flows are reliable across a multitude of failure scenarios with different failure-handling mechanisms. This plays an important role in minimizing data downtime.
- **Non-functional requirements.** Data orchestrators use a variety of components, from web servers to schedulers, workflow tasks, and DAG management engines. They must manage, coordinate, and ensure there is no single point of failure (SPOF). Another deployment and architectural challenge is ensuring high availability and fault tolerance across these different processes running on distributed systems.
- **Scaling.** Scaling data orchestration, scheduling, and workflow management in large enterprises that have hundreds and thousands of pipelines running 24-7 and across multiple tenants with a large number of data pipelines is extremely challenging. Orchestration engines should scale data-related operations through parallel processing and reusable pipelines, ensuring robustness as data volume and velocity rises.
- **Integration.** Data orchestrators need to integrate with enterprise tools and systems that span security, governance, and data cataloging. They need to support a wide array of prebuilt connectors and plug-ins for diverse data sources through REST APIs, database exports, cloud platforms, and messaging systems. Data orchestrators also need to integrate with machine learning (ML), artificial intelligence (AI) tools, and business process management tools across the enterprise.

Adoption Drivers

Three types of strategic initiatives contribute to enterprise investment in data orchestrators: digital transformation, data modernization, and advanced analytics.

1. As enterprises digitally transform businesses, they create software and hardware processes that need to be orchestrated.
2. The adoption of cloud platforms creates new interdependencies between on-premises systems and other cloud systems.
3. To adopt advanced analytics projects, organizations must integrate AI and ML projects with a complex web of data pipelines.

Given the increasing complexity of evolving data sources and requirements, there is a need to enable seamless data flow across the pipelines in enterprises by scheduling, automating, and monitoring the workflows to ensure recoverability, reproducibility, and resiliency. Some of the major adoption drivers include:

> **Automation.** Data orchestration makes it possible to accelerate data flow, automate repetitive tasks, streamline data pipelines, and operationalize them with increased efficiency accelerating data processing.

> **Integrate.** Data orchestrators integrate and interconnect different components into workflows with tasks and steps. They provision resources, manage schedules, provide state management across jobs, and enforce dependencies through a DAG.

> **Velocity.** Orchestration engines trigger jobs and coordinate tasks within jobs to enforce dependencies, log actions, and maintain traces for troubleshooting. This enhances the productivity of data pipelines with monitoring, testing, and quick feedback loops with repeatability, reusability, and reproducibility.

Guiding Principles for Tool Selection

In the rapidly evolving landscape of data engineering, various data orchestration tools offer distinct capabilities for managing, orchestrating, and automating data workflows. These workflows range from open-source software to commercial, from managed and self-hosted with low-code to code-heavy options.

Data orchestration solutions center around the concept of a direct acyclic graph (DAG)—that is, a graph with no loops or cycles—describing a collection of tasks that need to be executed. The DAG incorporates the interrelationships and dependencies of those tasks.

For every type of data orchestration tool—all of them have a few common components, such as *operators*—the fundamental unit of abstraction is a *task.* Operators are instantiated with specific arguments to define tasks. Each task consumes inputs and produces outputs. These data artifacts may be files, services, or in-memory data structures. All data orchestration tools also have a scheduler, workflow manager, and web server.

The next section discusses the common characteristics of data orchestration tools across the spectrum.

Must-Have Characteristics of Modern Data Orchestration Tools

Data teams should follow these guidelines when selecting data orchestrators:

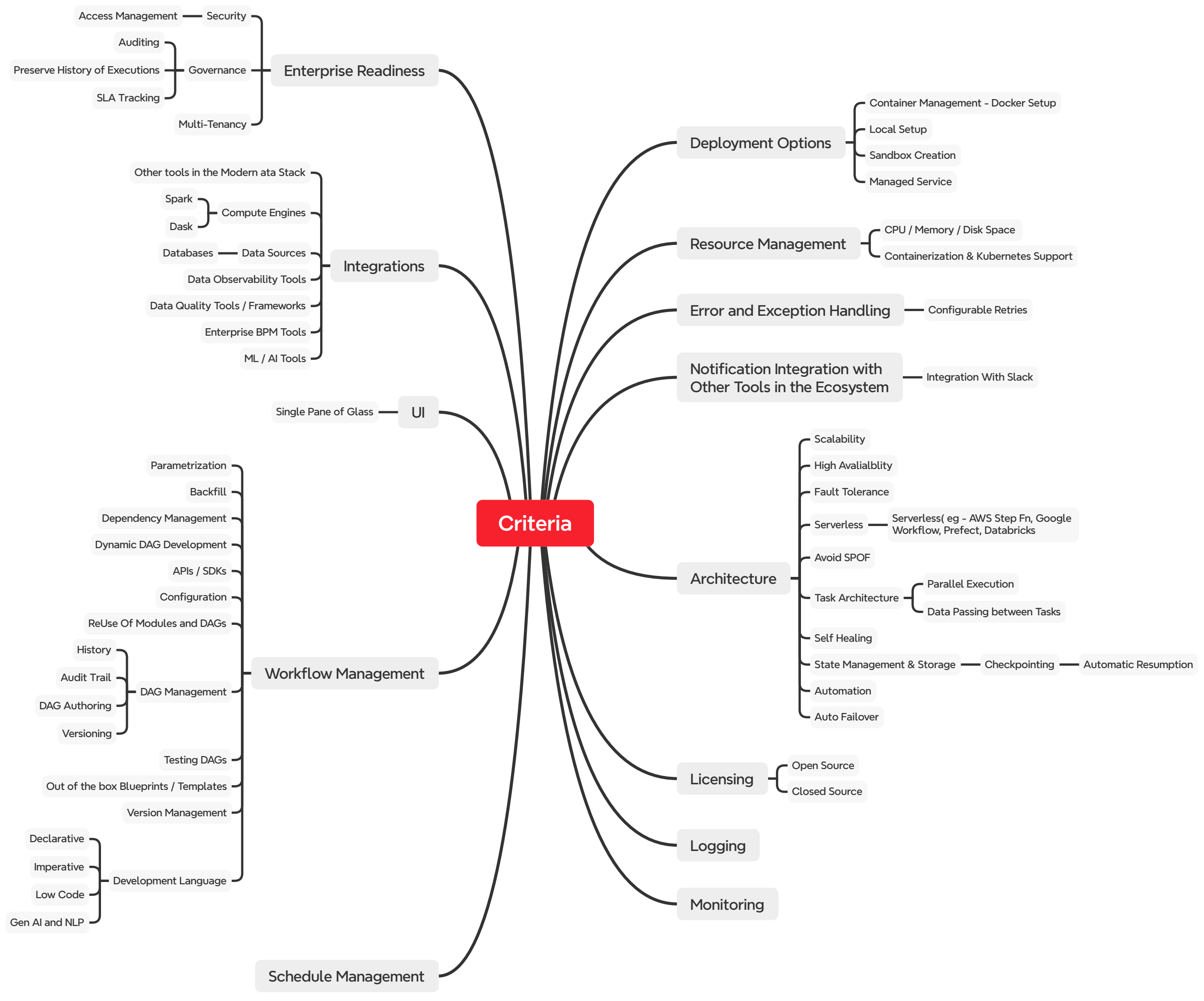
1. **Selection.** Choose these tools carefully, as the wrong choice can increase total cost of ownershipTCO, lower return on investment, ROI and hamper an organization's ability to become data-driven. Assess your organizational readiness and core competencies before selecting.

2. **Ease of use.** Evaluate whether a declarative or an imperative framework is best for the organization based on flexibility, ease of deployment, and turnaround time.

3. **Integration.** Ensure selected tools integrate with other tools and frameworks in the ecosystem—especially data observability, data quality, and data integration tools, as well as external third-party tools.

Figure 2 shows the different criteria data teams can use to understand and select data orchestration tools and frameworks.

Figure 2. Data Orchestration Requirements as a Mind Map



Architecture. For most organizations, the complexity and sheer volume of data pipelines is ever-increasing. Data orchestration framework should not only address current data processing requirements but scale with future growth without requiring substantial re-engineering time and infrastructure changes. It is imperative that the architecture be scalable for each component, highly available, and does not have a SPOF. Scaling als necessitates complex resource management capabilities and where particular tasks are executed on allocated resources in the most efficient and optimized manner.

The architecture should run orchestration processes in parallel with distributed tasks to increase workflow efficiency and pass data between tasks efficiently.

Wherever possible the architecture should embrace serverless capabilities to alleviate provisioning & managing infrastructure to reduce operational overhead. Any orchestration framework should provide flexibility to execute a diverse set of tasks and be expandable to implement different task types as needed.

Workflow management. The most important capability for a data orchestration tool is to build, manage, and schedule workflows. Workflow definition and task dependency management is a critical capability that differentiates data orchestration tools.

Build:

> Workflow CRUD operations with APIs, CLI, and UI that organize workflows for easy management with tagging and facets

> Parameterized & dynamic workflows

> Rapid conversion of functions to tasks, for example with Python decorators.

> No-code, low-code, drag-and-drop interface that integrates with LLM based assistants that enable non-technical domain experts to build data pipelines.

> Bindings to different languages for defining workflows with autocomplete, and code validation integrated in the IDE.

Operationalize:

> Secure workflow operations based on user identity, role, and access.

> DAG serialization and deserialization for workflow portability across systems.

> Configuration-based declarative pipeline definition to define orchestration, data flows, and dependencies.

> Modular orchestration in workflows that allows complex DAGs to be broken down into sub-DAGs for better management and reusability across teams.

> Conditional execution of tasks that creates branching logic in workflows with complex task dependencies.

Enterprise readiness. Data orchestration tools should be enterprise-ready and integrate with enterprise security frameworks and support SSO, RBAC, worker groups, and user access controls. Governance features such as data lineage tracking and metadata management help ensure compliance and maintain a clear understanding of job history and integrate with code & data version tools for job definition and version management of job runs.

UI capabilities. When running complex workflows, it's essential to have a clear place for observing what went wrong and for quickly taking relevant action. Ideally, tools should offer a single pane of glass, providing a unified point of control to create, manage, and observe different workflows. Additionally, they should enable users to drill down for granular insights into metrics for data pipelines.

These tools should integrate with enterprise security and access controls, offer a Command Line Interface (CLI), and allow integration with CI/CD flows to generate automated scripts. Furthermore, they should feature an intuitive UI with dashboards, such as Gantt charts and graphs. These dashboards should display job status, visualize task dependencies, and enable users to drill down into job history and resource consumption details.

Resource management. Data orchestration tools should provide configuration driven approaches for capacity planning of the associated tasks and their processes—in terms of memory, CPU and disk usage. This is especially important when workflow complexity grows with multiple tenants in an enterprise setup.

Data orchestration tools should keep track of resource consumption and integrate it with the dashboard. Managed cloud services report and monitor costs, and put proactive guardrails in place to ensure teams don't accidentally consume more resources than budgeted.

Deployment options. Fast deployments cycles that can be automated are critical for data pipelines to improve data product velocity. Orchestrators should integrate with deployment tools, spin up new environments, expand capacity of existing deployments and create separate deployments to isolate and support multi-tenancy. Ideally for large enterprises data orchestrators should deploy and manage workflows across availability zones, regions and across multiple clouds.

Testing. Testing is an important but often overlooked aspect of data orchestrators. Test workflows, their scheduling, testing failure scenarios both functionally as well as across the non-functional requirements. Data orchestration tools should include tests across handshaking points in a workflow, either natively or through third party tools and libraries.

Monitoring, logging, auditing. Data orchestration tools should offer centralized access to monitoring and observability of task statuses, landing times, and execution durations through logs and traces. This diagnostic information should be actionable, either manually or through rule-guided automation, to take necessary actions that affect the execution of the data pipeline.

Data orchestration tools should either natively support configuration-driven data monitoring and observability capabilities or integrate with external tools. They should provide alerts and notifications through diverse channels when failures occur or when service-level agreements are violated. They should store all the metadata for lineage and auditing as well as for training machine-learning algorithms powering the data observability tools.

Schedule management. Scheduling workflows and managing and monitoring schedules are integral to data orchestrators. Data orchestrators should be able to trigger simple, ad hoc parameterized jobs and be able to create customized complex schedules and manage them. They should provide a user interface that is intuitive to allow monitoring of the scheduled DAGs, as well as configuration-based and API-based approaches to work with schedules. Next-generation tools are trending to make task execution be "data-aware," as it is often difficult to predict tasks' completion time.

Error and exception handling. It is critical for data orchestration tools to quickly detect failures, errors, and exceptions in data pipelines and recover where possible. In the modern data ecosystem with its distributed architecture, failure is not a question of IF, but of WHEN. These failures can be caused by bad data or faulty servers. Whatever the cause—these issues need to be quickly detected and remedial action must be taken.

Data orchestration tools should incorporate (at the core level) retry techniques to recover from failures. They should offer a concise interface for configuring retry behavior with settings such as exponential backoff. They should handle timeouts gracefully. Tasks and pipelines will fail, so data orchestration tools should allow rescheduling, replay, reproduce, and retry capabilities. They should also provide capabilities to easily roll back any task or subtask and make data pipelines idempotent (meaning they produce the same results when rerun) wherever possible.

Integration. For the highly disaggregated data ecosystem to be successful, any data tool, framework, or library needs to integrate with other internal and external systems. Data orchestration tools in particular need to integrate with enterprise process and application orchestration tools as well as with ML and AI operational and orchestration tools.

Data orchestration tools need to integrate across internal and external business processes, share and transfer data across processes and systems, and support bidirectional data flows for synchronization and migration. They should be able to connect and work with plugins to data sources, such as databases, APIs, Git, cloud providers, OpenAI ChatGPT, and much more.

Out-of-the-box data orchestration should support a wide array of prebuilt connectors and plugins that simplify integration with almost any data source. They should provide capabilities to communicate with REST- and GraphQL-based APIs.

They should integrate data lineage and data observability and data quality tools and store and manage metadata that pipelines emit every time they run. This metadata should be integrated with the UI layer of the data orchestration tool to allow data teams to observe health and performance of mission-critical data pipelines and the data that flows through them and quickly troubleshoot issues as they happen.

Best practices. Some best practices in data orchestration include:

> Ensure data orchestrators provide different data personas with the ability to create, manage, and maintain data pipelines for their workloads with self-service.

> Reduce code (imperative/declarative) when creating DAGs with parameterization, build templates and blueprints, reuse them, and make DAGs configurable.

> Ensure data pipeline definitions are decoupled from actual orchestration definitions.

> Reduce data downtime and blast radius by enabling data teams to be alerted on the right channels when failure happens or when SLAs are violated.

> Choose orchestrators that integrate with data observability tools to provide insights and visibility into state of data pipelines.

> Choose orchestrators that validate input and output schemas. Define schema checkpoints before and after tasks to ensure schema validity.

> Capture and manage operational metadata, including audit logs and lineage for observability.

> Design automated distribution process coupled to CI (continuous integration) / CD (continuous deployment) process.

Databricks Workflows

Most data orchestration tools stand alone, with limited integration to data lakehouses and data warehouses. This leads to increased development, deployment, and troubleshooting time, with back-and-forth across different systems to build, manage, and operate data pipelines at scale. This introduces longer feedback loops and reduces velocity.

Unified. With the addition of a fully managed orchestration capabilities, the Databricks Data Intelligence Platform has become a one-stop shop for data ingestion and ETL/ELT. Users don't need to shift back and forth between third-party orchestration tools and Databricks to build, manage, and deploy data pipelines that combine data engineering, analytics and ML/AI. Previously, data teams created workflows with third-party tools and executed Databricks and Spark jobs by using an API. These tools were not integrated with the data lakehouse, making data practitioners unproductive, resulting in higher TCO and no single pane of glass to manage myriad jobs with integrated data governance and access controls.

Operationalization. Data personas can build workflows on any cloud without managing complex infrastructure. This mitigates operational overhead, allowing data teams to focus on workflows rather than infrastructure and deployment challenges. Databricks Workflows can be used by all personas, and workflows can be built using UI, API and code, including Python, and Terraform.

Databricks Workflows offers some unique capabilities:

> Configuration-driven repair and rerun when workflow fails by running failed tasks automatically.

> Integration with code versioning systems to build and update workflows with committed code.

> Supports a rich task API to set and retrieve values and share context from upstream tasks and systems, making it easier to customize task dependency to outcomes.

> Integrate diverse task types from dbt (an open source data build tool) to SQL, allowing teams to orchestrate and coordinate complex jobs and chaining with parent-child and nested dependencies.

> Diverse ways to create and execute jobs with CLI (command line interface), API, UI, and a single pane of glass with a unified matrix view across tenants to monitor and diagnose health across executions.

> Set **callbacks** for failures and SLA misses and track long-running workflows.

> Integration with **Unity Catalog** for automatic data lineage and unified governance.

> Fine-grained notification of alerts, with control over users and groups to be alerted across job stages, as well as alerts types, events, and recipients.

> Tracking costs with tags associated with jobs and clusters. For Databricks Workflows, the associated cost is just the compute time.

> Ability to associate tags with jobs, allowing easy findability across UI, CLI and APIs.

Best Practices

When working with the Databricks Workflows, some best practices include:

> Jobs should be owned by **service principals** (i.e. machine users) and not individual users.

> Always tag jobs for easy search and discovery.

> Leverage integrated version control for notebooks to select the last committed job.

> Use **job clusters** (ephemeral compute) that are active only as long as needed for the job to complete.

Databricks Workflows has simplified lakehouse customers' workload and mitigated the challenges of learning, setup, deployment, maintenance, and troubleshooting of separate systems for building lakehouses and orchestrating workflows.

Customer Feedback Across Case Studies

Databricks' customer-driven move to build an integrated data orchestration and workflow with the rest of the Databricks ecosystem has reaped rich dividends for multiple customers across diverse verticals in the form of a better user experience and a smaller footprint, resulting in higher deployment velocity (e.g. **Wood Mackenzie** and **YipitData**). Databricks is the first data lakehouse platform to provide this capability.

Databricks Workflows has lowered the barrier to entry for building and orchestrating data pipelines. Its features (configurable out of the box) and templates allow users to quickly define, schedule, manage, and monitor data pipelines, empowering customers to quickly create reusable workflows that promote productivity, data quality, transparency, and an easier path to troubleshoot and fix pipeline issues.

Customers find that cognitive overhead is reduced when working with an intuitive, single-pane-of-glass UI that is natively integrated with Databricks workspace. Onboarding new users is faster and easier. The UI enables users to search and identify running jobs among hundreds of simultaneously running pipelines.

Integration with other tools in the ecosystem, such as data and code versioning and data observability, has considerably reduced time to market, and Databricks Workflows' capability of cluster reuse has resulted in cost savings of more than 50% for some customers (such as **Ahold Delhazie**).

Conclusion

Many organizations do not sufficiently address the critical last mile of data pipeline management. This last mile centers on data orchestration—that is, the scheduling and optimization of pipelines and the larger workflows they help drive. Data engineers and data ops teams building data driven systems and services with data pipelines should:

> Select data orchestration tools early in the project. Do not leave this decision to the DataOps team to consider during deployment. Data orchestration tool selection should be formally planned with a proof of concept, tested, and reviewed by all stakeholders.

- Leverage data orchestration tools only for data orchestration. Don't run expensive, compute-heavy tasks within the data orchestration tools.
- Avoid manual and ad hoc approaches for data orchestration with scripts and non-standard approaches.
- Carefully evaluate cloud-native solutions, especially those that are based on open source software, as they may be too restrictive for certain use cases.

About Eckerson Group

Wayne Eckerson, a globally-known author, speaker, and consultant, formed **Eckerson Group** to help organizations get more value from data and analytics. His goal is to provide organizations with expert guidance during every step of their data and analytics journey.

Eckerson Group helps organizations in three ways:

> **Our thought leaders** publish practical, compelling content that keeps data analytics leaders abreast of the latest trends, techniques, and tools in the field.

> **Our consultants** listen carefully, think deeply, and craft tailored solutions that translate business requirements into compelling strategies and solutions.

> **Our advisors** provide competitive intelligence and market positioning guidance to software vendors to improve their go-to-market strategies.

Eckerson Group is a global research, consulting, and advisory firm that focuses solely on data and analytics. Our experts specialize in data governance, self-service analytics, data architecture, data science, data management, and business intelligence.

Our clients say we are hard-working, insightful, and humble. It all stems from our love of data and our desire to help organizations turn insights into action. We are a family of continuous learners, interpreting the world of data and analytics for you.

Get more value from your data. Put an expert on your side. **Learn what Eckerson Group can do for you!**

Eckerson Group

RESEARCH CONSULTING ADVISORY

GET • MORE • VALUE • FROM • YOUR • DATA

About the Sponsor

Databricks is the Data and AI company. More than 10,000 organizations worldwide — including Comcast, Condé Nast, and over 50% of the Fortune 500 — rely on the Databricks Data Intelligence Platform to unify their data, analytics and AI. Databricks is headquartered in San Francisco, with offices around the globe. Founded by the original creators of Delta Lake, Apache SparkTM, and MLflow, Databricks is on a mission to help data teams solve the world's toughest problems. To learn more, follow Databricks on **Twitter, LinkedIn,** and **Facebook.**