DATA ORCHESTRATION: THE CORE PILLAR FOR DATAOPS



Data orchestration is the process of coordinating and automating the flow of data across complex pipelines—managing task dependencies, scheduling, failure handling, and recovery end to end. In a DataOps framework, data orchestration is the foundational layer that transforms fragmented, ad hoc data operations into reliable, repeatable workflows. Together, data orchestration and DataOps give data engineering teams the velocity, governance, and operational rigor needed to build and sustain data-driven platforms at scale.

Why do organizations need DataOps?

All organizations want to be data-driven, but there is a significant gap between that ambition and execution. Emerging technologies are immature and not battle-tested—it is the operationalization of technology that is the true path to becoming data-driven.

Most data teams do not plan for "Day 2"—the period that begins after product teams complete development and deploy to production. Do they have an end-to-end process to deploy artifacts? Have they run functional, performance, load, and stress tests? Are they prepared to roll back production changes if problems arise while keeping the lights on?

The disconnect between running POCs and POVs with emergent technologies and successfully deploying real-life use cases to production is widespread. Most of the contributing causes can be addressed by the missing component in the data economy: DataOps. Many organizations practice DataOps, but in an ad hoc, fragmented way—built without guidelines, specifications, or a

formalized process.

Data infrastructures today—spanning ingestion, storage, and processing—are deployed across distributed systems: on-premises, public and private cloud, hybrid, and edge environments. These environments are a complex mix of servers, virtual machines, networking, memory, and CPU where failures are inevitable. Organizations need tools and processes that can quickly perform root cause analysis and reduce mean time to recovery (MTTR).

DataOps eliminates gaps, inefficiencies, and misalignments across every step from data production to consumption. It coordinates and orchestrates development and operationalization in a collaborative, structured, and agile manner—enabling organizations to streamline data delivery, improve productivity through automation, and deploy data-driven applications with trust and governance.

What is DataOps?

DataOps streamlines and automates data processes and operations to speed the building of data products and solutions and to help organizations become data-driven. The goal of DataOps is to move from ad hoc data practices to a formalized data engineering approach with a controlled framework for managing processes and tasks.

DataOps is accomplished through a formal set of processes and tools that detect, prevent, and mitigate issues arising during pipeline development and production deployment. DataOps applies CI/CD principles to develop agile data pipelines and promotes reusability through data versioning to manage integration collaboratively from data producers to consumers. It also reduces the end-to-end time and cost of building, deploying, and troubleshooting data platforms and services.

For organizations investing in digital transformation with analytics, AI, and ML, DataOps is an essential practice for unlocking data assets and improving decision-making. DataOps enables innovation through decentralization — and a data orchestration and workflow automation platform provides the centralized coordination that turns disaggregated data tools into a coherent, governed pipeline. Handling global orchestration across shared infrastructure with inter-domain dependencies and policy enforcement.

Why do organizations need data orchestration?

Data orchestration is necessary because no single tool handles the full complexity of a modern data stack. The data ecosystem today offers a rich landscape of tools, frameworks, and libraries—some accessible through well-defined APIs, others invoked via command lines—that must all be stitched together to build end-to-end data-driven systems.

Most organizations do not manage a handful of pipelines—they manage tens or hundreds, with complex deployment models spanning edge to on-premises to multiple cloud data centers. Components run across different data centers built on distributed architectures with sub-components spread across servers and virtual machines. Every handshaking point is a potential failure point, from network disruptions to server outages.

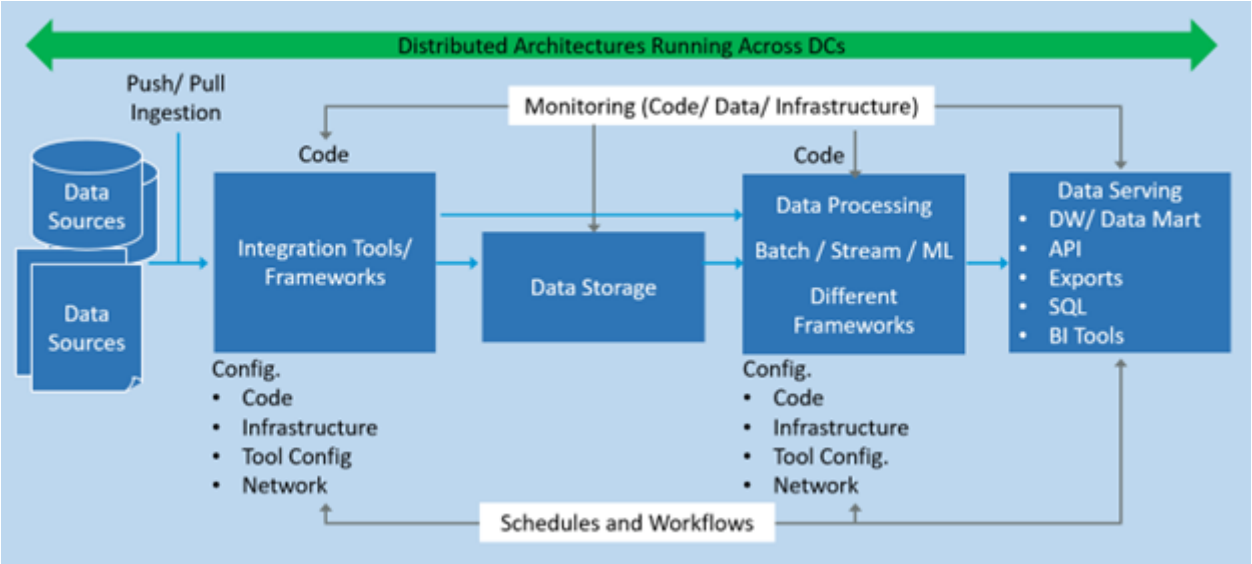


Figure 1. A cross-sectional view of a simple data pipeline.

When building these pipelines, data engineers must guarantee that pipeline code executes as intended, adheres to data engineering best practices, and handles both the happy path and failure scenarios. The goal is to coordinate every component so the entire pipeline is resilient, supports failover, and recovers from the point of failure.

Production pipelines are complex—multiple forks, dependencies, and trigger logic, all managed by the orchestrator's scheduling and workflow engine. Data orchestration controls and coordinates workflows across multiple stakeholders, tracks execution history, triggers jobs, enforces task dependencies, and logs actions with full audit traces for troubleshooting. The top capabilities requested by data engineers and data scientists include ease of use, monitoring, and strong pipeline observability—all delivered by a well-implemented data orchestration layer.

What is data orchestration?

Data orchestration is the glue between tasks across the data stack. It manages dependencies, coordinates tasks in defined workflows, schedules execution, manages outputs, and handles failure scenarios. The orchestration process governs data flow according to orchestration rules and business logic, scheduling and executing the workflows associated with each data flow.

This flow structure is represented as a dependency graph, also called a DAG (directed acyclic graph). Data orchestration is the process of connecting tasks into a chain of logical steps that forms a well-coordinated, end-to-end data pipeline.

Data orchestrators fall into two types: task-driven and data-driven. A task-driven orchestrator focuses solely on workflow execution order, without awareness of what data enters or exits each step. A data-driven orchestrator goes further—it is aware of the data flowing between tasks, including artifact outputs that can be version-controlled and associated with automated tests.

A good data orchestrator lets data teams quickly isolate errors and failures. Data orchestrators can be triggered by time-based schedules or custom-defined event logic, and the infrastructure required by data domains can be unified into a self-service infrastructure-as-a-platform managed through a DataOps framework.

What are data orchestration best practices?

Data orchestration should be configuration-driven to ensure portability — which is also a core principle of a broader workflow orchestration platform that spans the full application and data pipeline stack.

Effective data orchestration requires deliberate design decisions. The following practices improve pipeline resilience, portability, and recoverability.

Decouple DAGs into sub-DAGs

Break complex dependency graphs into the simplest possible sub-DAGs. Smaller, focused sub-DAGs are easier to test, reason about, and re-execute independently when a failure occurs.

Make DAGs fault-tolerant

Design sub-DAGs so that if one component breaks, it can be re-executed without rerunning the entire pipeline from the beginning.

Use configuration-driven orchestration

Configuration-driven data orchestration ensures portability across environments and supports repeatability and reproducibility—critical for teams managing pipelines across development, staging, and production.

Enable single-click recovery with checkpoints

Data orchestration processes should include checkpoints that support single-click recovery from the point of failure, with the ability to retry failed tasks using a configurable backoff strategy.

Conclusion

Enterprises should not build data platforms for data-driven decision-making without first incorporating the principles of data engineering and DataOps. These synergistic capabilities provide organizations with the process formality and velocity needed to scale, while reducing data and technical debt in the long run. Data orchestration is not an optional layer—it is the core pillar on which production-grade DataOps is built.

Frequently asked questions

What is the difference between data orchestration and data integration?

Data integration is the process of combining data from multiple sources into a unified view. Data orchestration manages the broader workflow—scheduling, sequencing, dependency enforcement, failure handling, and recovery—across the entire data pipeline. Data integration tasks typically run as steps within a larger data orchestration workflow.

What is a DAG in data orchestration?

A DAG, or directed acyclic graph, is the dependency structure used in data orchestration

to represent the relationships between pipeline tasks. Each node in a DAG is a task; each edge defines a dependency between tasks. The "acyclic" property means no circular dependencies exist, making pipelines predictable and easier to debug.

What is the difference between task-driven and data-driven orchestration?

A task-driven orchestrator manages workflow execution order without awareness of the data flowing between steps. A data-driven orchestrator is aware of the data and artifact outputs at each stage—enabling version control, automated testing, and stronger data lineage tracking across the pipeline.

How does data orchestration support a DataOps framework?

Data orchestration is the operational core of a DataOps framework. It automates pipeline execution, enforces task dependencies, handles failures with recovery logic, and provides full audit traces for governance. Without data orchestration, DataOps practices remain ad hoc and are difficult to scale across complex, multi-environment data infrastructures.

What triggers a data orchestration workflow?

Data orchestration workflows are triggered in two primary ways: on a time-based schedule (such as hourly or nightly batch jobs) or through custom event-based logic (such as a new data file arriving or an upstream pipeline completing). Most enterprise orchestrators support both trigger types within a single pipeline configuration.

The views and opinions expressed in this post are those of the author and do not necessarily reflect the official position of BMC.