WHAT IS A DATA PIPELINE? A COMPLETE GUIDE



A data pipeline is a series of automated steps for moving data from one or more sources to a designated destination, often transforming it along the way. Raw, disparate pieces of data enter one end, undergo processes like cleaning, restructuring, and enrichment, and emerge at the other end as usable insights. Data pipelines are the foundation of every analytics dashboard, machine learning model, and data-driven operational decision.

You could think of a data pipeline like an airport baggage system: bags (data) enter the conveyor system, get scanned (validation), sorted (transformation), and routed to the correct flight (destination database). If one belt jams, everything backs up—just like a bottleneck in a data pipeline.

What's the difference between a data pipeline and ETL?

A data pipeline and an ETL pipeline are not the same thing, though the terms are often used interchangeably.

A data pipeline is an umbrella term for any set of processes that move data from one system to another. This includes simple data ingestion, real-time streaming, batch processing, and complex multi-step workflows.

An ETL pipeline (Extract, Transform, Load) is a specific type of data pipeline. Its purpose is to extract data from sources, transform it into the right format, and load it into a destination system like a data warehouse or database. All ETL pipelines are data pipelines, but not all data pipelines are ETL pipelines. A pipeline that moves raw data without transforming it, or streams data in real time, is still

a data pipeline—but not an ETL pipeline.

Is SQL a data pipeline?

No. SQL (Structured Query Language) is a language used to query, manage, and manipulate data in relational databases—it's a tool, not a workflow.

A data pipeline is the full automated process that moves data from one place to another. SQL can be used inside a data pipeline to filter, join, or transform data, but SQL doesn't constitute the pipeline itself. If you're building a house, SQL is the hammer and saw—essential tools, but not the entire construction project.

Why do data pipelines matter?

Data pipelines are essential because organizations collect data from too many sources—customer interactions, social media, sales transactions, website logs, IoT devices, and internal applications—to manage manually. Without a systematic way to collect, process, and deliver this data, it quickly becomes unmanageable rather than useful.

Data pipelines power the data-driven decisions we encounter every day: personalized recommendations, real-time fraud alerts, and analytics dashboards. For DataOps teams specifically, pipelines help ensure the reliability, scalability, and governance that organizations need to use data confidently instead of being overwhelmed by it.

What are the benefits of a well-executed data pipeline?

Data pipelines don't just move data—they make it fit for purpose, delivering it where and when it's needed. Five benefits go beyond the basic transport function:

Enabling analytics and business intelligence: Pipelines feed cleaned, structured data into data warehouses and analytical platforms, allowing analysts to discover trends, identify opportunities, and monitor performance.

Fueling machine learning and AI: AI models require large volumes of high-quality, pretreated data. Data pipelines help ensure models get the data they need to learn and make accurate predictions.

Ensuring data quality and governance: As data gets cleaned, validated, and standardized, data pipelines support greater confidence in data-driven decisions. They also enforce governance rules for compliance and security.

Improving operational efficiency: By integrating data from various systems, pipelines provide a holistic view of operations, automating workflows and flagging issues in real time.

Facilitating data democratization: Pipelines make data accessible and understandable to more people within an organization, empowering more teams to make informed decisions by connecting data sources to decision-makers.

Without strong data pipelines, organizations can fly blind—making decisions based on intuition rather than evidence.

What are the core components of a data pipeline?

Every data pipeline is made up of five essential components. Understanding these elements reveals how data flows and transforms from its original source to its final destination.

1. Source: Where your data lives

Source is the origin point of your data—the starting line of the pipeline. The type of source determines how the data will be extracted. Common data sources include:

- Databases: relational (e.g., MySQL) and NoSQL (e.g., MongoDB)
- Applications: CRM systems (e.g., Salesforce), ERPs (e.g., SAP), marketing automation platforms
- APIs: third-party services, social media platforms, public data feeds
- Files: CSVs, JSON, XML, Parquet, Avro—often stored in cloud storage (e.g., Azure Blob)
- Streaming data: real-time event streams from IoT devices, website clicks, financial transactions
- Logs: system logs, web server logs, application logs

2. Extraction: Getting your data out

Extraction is the step where data is pulled from its original source—often involving different file types, formats, and sometimes unstable or slow source connections. The goal of extraction is to get a raw copy of the data without altering the source system.

Three common extraction methods:

- Batch extraction: Data is pulled in chunks at scheduled intervals (e.g., nightly, hourly). Used for data that doesn't change frequently or where immediate updates aren't critical.
- Incremental extraction: Only new or changed data since the last pipeline run is extracted. Faster than full extraction, but requires change-detection techniques like timestamps, version numbers, or Change Data Capture (CDC).
- Streaming extraction: Data is continuously pulled from sources as events occur, typically using message queues or event streaming platforms like Kafka or Kinesis.

3. Transformation: Cleaning and shaping your data

Transformation is usually the most complex part of a data pipeline. This is where the messiness of raw data gets cleaned and turned into actionable information. The goal is to ensure data quality, consistency, and suitability for the intended destination. Common transformation steps include:

- Cleaning: Removing duplicates, handling missing values, correcting errors
- Filtering: Selecting only relevant rows or columns
- Aggregating: Summarizing or categorizing data (e.g., total sales per day)
- Joining or merging: Combining data from multiple sources using common keys (e.g., joining customer data with order data)
- Standardizing or normalizing: Ensuring consistent data types, formats, and units (e.g., standardizing currency codes)
- Enriching: Adding new data points by looking up external information or deriving new features (e.g., adding geographical data based on an IP address)

- Structuring: Converting unstructured or semi-structured data into a structured format

4. Loading: Delivering your data

Once extracted and transformed, data is loaded into the system where it will be used—a database, data warehouse, or analytics platform—so applications, reports, and analytics tools can access it. Three common loading strategies:

- Full load: The entire destination table or dataset is overwritten with new, transformed data. Simpler to implement, but resource-intensive for large datasets.
- Incremental load: Only new or changed records are appended to the destination. More efficient, but requires diligent management of data updates and deletions.
- Streaming load: Data is continuously loaded as it arrives, often into specialized real-time databases or analytical engines.

5. Destination: Where your data rests

Destination is the final storage location where processed data is available for consumption by analysts, data scientists, and applications. Common destinations include:

- Data warehouses: Optimized for complex queries and reporting on large volumes of historical data (e.g., Snowflake)
- Data lakes: Hold raw or semi-structured data at scale for advanced analytics and machine learning (e.g., Azure Data Lake Storage)
- Databases: Operational systems for everyday applications like websites or apps (e.g., MongoDB)
- Business intelligence (BI) tools: Software that turns data into dashboards and reports (e.g., Tableau)
- File storage: Simple storage for archiving or later processing

Note: loading and destination are conceptually distinct but closely related in practice. Loading is the action of writing processed data into a system; destination is the place where that data ends up and stays until it's needed. Loading is like putting groceries into the fridge. Destination is the actual fridge.

Bringing it all together—an e-commerce scenario: Source (CRM database) Extract (SQL query for new customer orders) Transform (clean addresses, calculate total order value, join with product details) Load (insert into data warehouse) Destination (data warehouse for reporting). The entire process runs automatically on a schedule, ensuring a continuous flow of refined information.

What are the three types of data pipelines?

Just as there are different ways to transport goods, there are different types of data pipelines—each optimized for specific needs around speed, volume, and complexity.

Batch processing: the daily shuttle

Batch processing works like a commuter train on a defined schedule. It picks up a large group of passengers (data) at scheduled times and delivers them to their destination. Data is collected over a

period, then processed as a single, large batch.

- Characteristics: High latency (data may be hours or days old); processes large volumes efficiently; often scheduled during off-peak hours
- Use cases: Nightly reports, monthly financial summaries, loading historical data into a data warehouse, running complex analytical jobs that don't require immediate results

Real-time streaming: the instant delivery service

Real-time streaming works like an instant delivery service. As soon as a package (data event) is created, it's picked up, processed almost immediately, and delivered to its destination with minimal delay.

- Characteristics: Low latency (data is typically seconds or milliseconds old); handles continuous streams of individual events; requires infrastructure optimized for speed
- Use cases: Fraud detection, real-time personalized recommendations, IoT sensor data analysis, monitoring system health, live dashboards

Hybrid approaches: the best of both worlds

Many organizations combine batch and streaming pipelines. Two common hybrid patterns:

- Lambda architecture: Uses separate batch and streaming layers. The streaming layer provides real-time views; the batch layer processes historical data for accuracy and completeness. Results from both are then merged.
- Kappa architecture: A simpler approach that handles both real-time and historical processing using a single streaming engine, often by replaying streams.

Choosing the right pipeline type depends entirely on your business requirements for data freshness, volume, and complexity.

What are the main challenges in data pipeline management?

Building a data pipeline is one thing; keeping it running smoothly and reliably is another. Here are the most common data pipeline management challenges—and the best practices for each.

Ensuring data quality: Data can be inconsistent, incomplete, or incorrect at the source, leading to garbage in, garbage out. Best practices: implement data validation rules at every stage; use data profiling tools to understand data characteristics; create data quality checks within transformation steps; use data observability platforms to detect anomalies early.

Scalability and performance: As data volumes grow or requirements shift to real-time, pipelines can become slow or break entirely. Best practices: design for scalability from the outset; use distributed processing frameworks (e.g., Apache Spark); use cloud-native services that scale automatically; implement incremental loading strategies; optimize queries and transformation logic.

Security and compliance: Data pipelines handle sensitive information, requiring stringent security and compliance measures. Best practices: encrypt data at rest and in transit; implement strong access controls (least privilege); audit data access and changes; redact sensitive data during transformation where necessary.

Monitoring and alerting: Without proper monitoring, pipeline failures or data quality issues can go undetected and impact downstream applications. Best practices: implement comprehensive monitoring for pipeline health, performance metrics, and data quality metrics; set up automated alerts for critical failures, latency breaches, or data anomalies; use dashboards for operational visibility.

What are the top data pipeline use cases?

Data pipelines are versatile, powering applications across industries. Seven key examples of how pipelines transform raw data into actionable insights:

- Business intelligence and reporting: Aggregating sales data, customer demographics, and marketing spend into a data warehouse for daily, weekly, or monthly reports and dashboards that guide strategic decisions
- Customer 360-degree view: Combining data from CRM, sales, support, and marketing platforms to create a holistic profile of each customer, enabling personalized experiences and targeted campaigns
- Fraud detection: Ingesting real-time financial transactions, social media activity, and user behavior to identify suspicious patterns and instantly flag potential fraud
- IoT analytics: Collecting streams of data from sensors (e.g., factory machines, smart city devices) to monitor performance, predict maintenance needs, and optimize operations
- Personalized recommendations: Processing user browsing history, purchase data, and demographic information to power content recommendations on streaming platforms
- Log analytics: Consolidating logs from applications and servers to monitor system health, troubleshoot issues, and detect security threats in real time
- ML model training: Preparing, cleaning, and feeding large datasets to machine learning models for tasks like image recognition, natural language processing, or predictive analytics

What makes a data pipeline modern?

Modern data pipelines share several key characteristics that distinguish them from traditional approaches:

- Cloud-native and serverless: Modern pipelines use cloud services (e.g., AWS, Azure) that scale automatically and reduce operational overhead
- ELT-first approach: Instead of transforming data before loading, modern pipelines often load raw data into a cloud data warehouse (e.g., Snowflake, BigQuery) and transform it within the warehouse using SQL—leveraging the destination's compute power and enabling greater flexibility
- Data lake integration: Modern pipelines frequently integrate with data lakes to store vast amounts of raw, multi-structured data for future use, advanced analytics, and machine learning
- Real-time capabilities: Streaming technologies like Apache Kafka and Amazon Kinesis increasingly power modern pipelines for immediate insights
- Orchestration and automation: Tools like Apache Airflow or cloud-native orchestrators automate scheduling, manage dependencies, and monitor pipeline health
- Data observability: Modern pipelines go beyond basic monitoring to actively track the health, quality, and lineage of data—detecting anomalies and ensuring data trustworthiness

- Data governance and security by design: Security, privacy, and compliance are built into the pipeline architecture from the start, not added as an afterthought
- Flexibility and agility: Modular components make modern pipelines easier to adapt to new data sources and changing business requirements

How do traditional and modern data pipelines compare?

Modern data pipelines are designed to be more agile, scalable, cost-effective, and resilient than traditional approaches. Here's a side-by-side comparison:

Feature	Traditional Data Pipeline	Modern Data Pipeline
Scalability	Limited by fixed resources and batch processing constraints	Highly scalable and elastic, using cloud infrastructure to adjust resources automatically
Processing	Primarily batch processing (e.g., hourly, daily)	Supports both batch and continuous, real-time processing
Flexibility	Less flexible; requires significant manual adjustments for changes	More flexible and adaptable; uses metadata to handle changes automatically
Infrastructure	Traditional, monolithic, on-premises systems	Cloud-native and microservices-based, with independent compute resources
Automation	Limited automation	High automation, including automated restarts and retries
Data access	Data access can be restricted	Democratizes data access and enables self-service management
Real-time capabilities	Lower latency due to batching; not typically real-time	Low latency with options for real-time processing and immediate data availability

What are the most common data pipeline tools?

The data pipeline tool landscape is broad and evolving. Four categories to consider:

Data integration platforms: Comprehensive ETL/ELT tools, often with visual interfaces and pre-built connectors. Examples: Talend, Informatica. A good fit for teams that want an end-to-end solution without heavy coding; businesses with multiple data sources needing pre-built connectors; and organizations prioritizing ease of use and quick deployment.

Cloud-native services: Major cloud providers offer services specifically designed for scalable data pipelines. Examples: Amazon Kinesis, Google BigQuery. A good fit for companies already invested in a specific cloud ecosystem; teams needing scalable, cost-effective solutions for batch and streaming; and use cases requiring tight integration with other cloud services.

Open-source frameworks: Flexible, developer-friendly options for orchestration and processing. Examples: Apache Airflow, Apache Kafka. A good fit for engineering teams with strong technical skills; organizations wanting maximum flexibility and control; and scenarios with custom requirements or large-scale data processing needs.

Enterprise workflow orchestration platforms like Control-M are ideal for teams that need to orchestrate BigQuery data pipelines alongside mainframe, cloud, and on-premises workloads — providing SLA management and audit capabilities that native GCP scheduling does not offer. A good fit for large enterprises with complex, mission-critical workflows; businesses needing robust scheduling, compliance, and audit capabilities; and teams managing cross-platform jobs spanning mainframe, cloud, and on-premises systems with high reliability requirements.

For large enterprises with mission-critical data pipelines, a data pipeline orchestration platform like Control-M provides robust scheduling, cross-platform dependency management, and compliance capabilities that open-source tools alone cannot deliver at scale.

The right tool depends on your budget, team skill set, data volumes, and real-time requirements.

5 key data pipeline takeaways

Whether you're learning about data pipelines for the first time or refreshing on the fundamentals, here are the most important points to carry forward:

1. Understand the pipeline lifecycle

A data pipeline isn't just about moving data—it involves extraction, transformation, loading, orchestration, monitoring, and governance.

2. Orchestration is key

Orchestration ensures repeatability, scalability, and observability across the entire data pipeline.

3. Embrace automation and CI/CD

Integrating data pipelines into CI/CD workflows enables faster, safer changes. DataOps applies DevOps principles to data: automated testing, deployment, and version control for pipelines.

4. Prioritize data quality and monitoring

Pipelines can fail silently if data quality isn't checked. Implement validation, anomaly detection, and alerts to catch issues early. Observability is critical for trust and compliance.

5. Design for scalability and flexibility

Modern data pipelines must handle batch and streaming, adapt to schema changes, and scale with data growth. Cloud-native and modular architectures are essential for agility.

Bonus tip: Learn the full ecosystem—ETL tools, orchestration frameworks, cloud services—and how they fit together to build pipelines that serve your organization at scale.

Frequently asked questions about data pipelines

How is a data pipeline different from a data warehouse?

A data pipeline is the process that moves and transforms data; a data warehouse is a destination where processed data is stored for analysis. Data pipelines feed data warehouses—the pipeline handles transport and preparation, while the warehouse provides the storage and query environment. The two are complementary, not interchangeable.

What skills do you need to build a data pipeline?

Building a data pipeline typically requires proficiency in SQL and at least one programming language (Python and Scala are common), familiarity with data transformation concepts (ETL/ELT), and working knowledge of at least one orchestration or workflow tool. Cloud platform experience (AWS, Azure, or GCP) is increasingly essential for modern data pipeline work.

How do you test a data pipeline?

Data pipeline testing involves validating data at multiple stages: confirming that extraction produces complete, unaltered source data; verifying that transformation logic produces expected outputs; and checking that loaded data matches expected row counts, data types, and values at the destination. Automated testing frameworks integrated into CI/CD pipelines can run these checks continuously and catch regressions early.

What is data pipeline latency?

Data pipeline latency is the time between a data event occurring at the source and that data becoming available at the destination. Batch pipelines have high latency (hours or days); real-time streaming pipelines have low latency (seconds or milliseconds). Acceptable latency depends on the use case—fraud detection requires near-zero latency, while monthly financial reporting can tolerate daily batch processing.

How does BMC Control-M help manage data pipelines?

BMC Control-M is an enterprise workflow orchestration platform that automates, schedules, and monitors complex data pipelines across environments—including mainframe, cloud, and on-premises systems. Control-M provides centralized visibility, dependency management, and audit capabilities, making it well-suited for mission-critical pipelines that require high reliability, compliance, and cross-platform coordination.

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