DZone

TREND REPORT

JUL 2025

DATA ENGINEERING

Scaling Intelligence With the Modern Data Stack

BROUGHT TO YOU IN PARTNERSHIP WITH

DATASTAX
an IBM Company

Table of Contents

Meet the Team

DZone's Content and Community team can be found reviewing contributor pieces, working with authors and sponsors, and coordinating with our researcher and designer to deliver valuable, high-quality content to global DZone-ians.

Carisse Dumaua	*Content Editor*
Melissa Habit	*Senior Manager, Production Ops*
Lucy Marcum	*Acquisitions Editor*
Dominique Roller	*Community Specialist*
Ging Villena	*Content Editor*

ORIGINAL RESEARCH

Key Research Findings

An Analysis of Results From DZone's 2025 Data Engineering Survey

G. Ryan Spain, Freelance Software Engineer, former Engineer & Editor at DZone

Across the globe, companies are leveling up their data capabilities and analytics maturity. While organizations have become increasingly aware of the copious new technologies at our disposal, it's now about how we can use these in a thoughtful, efficient, and strategic way. We understand the nuances and capabilities, but how do we apply them to our own teams, workflows, and business requirements?

The focus is on **consolidation** and **optimization** — companies are scrutinizing costs, monitoring performance, and leveraging orchestration more deeply than ever before. A constantly maturing data stack means regular shifts across teams, tooling, and processes.

In this year's *Data Engineering* Key Research Findings, DZone explores the evolved capabilities and use cases in modern data engineering, including the rise of generative AI, trends in data warehouse, lake, and lakehouse usage, data pipeline architecture and integration, and more.

In June and July, DZone surveyed software developers, architects, and other IT professionals in order to gain insight on the current state of data engineering in the software development space.

Methods

We created a survey and distributed it to a global audience of software professionals. Question formats included mainly single and multiple choice, with options for write-in responses in some instances. The survey was disseminated via email to DZone and TechnologyAdvice opt-in subscriber lists as well as promoted on dzone.com, in the DZone Core Slack workspace, and across various DZone social media channels.

The data for this report were gathered from responses submitted between June 27 and July 15, 2025; we collected 123 complete and partial responses. Our margin of error for the results of this survey is 10%, at a confidence level of 95%, and this report treats comparative results of 10% or less as insignificant. Segmented responses may have lower confidence levels due to smaller sample sizes.

Demographics

Before diving in, we noted key audience details in order to establish a more solid impression of the sample from which results are derived:

- 32% of respondents described their primary role in their organization as "Developer/engineer," and 10% described "Developer team lead." No other role that we provided was selected by more than 10% of respondents.*
- 67% of respondents said they are currently developing "Web applications/Services (SaaS)," 47% said "Enterprise business applications," and 20% said "Native mobile apps."
- "Python" (80%) was the most popular language ecosystem used at respondents' companies, followed by "Java" (43%), "JavaScript (client-side)" (41%), "Node.js (server-side JavaScript)" (26%), and "C#" (23%).
- Regarding responses on the primary language respondents use at work, the most popular was "Python" (50%), followed by "Java" (22%). No other language was selected by more than 10% of respondents.
- On average, respondents said they have 14.65 years of experience as an IT professional, with a median of 13 years.
- 32% of respondents work at organizations with < 100 employees or reported being self-employed, 19% work at organizations with 100-999 employees, and 44% work at organizations with 1,000+ employees.*

Note: For brevity, throughout the rest of these findings, we will use the term "developer" or "dev" to refer to **anyone actively involved in the creation and release of software, regardless of role or title. Additionally, we will define "small" organizations as having < 100 employees, "mid-sized" organizations as having 100-999 employees, and "large" organizations as having 1,000+ employees.*

Major Research Targets

In our 2025 Data Engineering survey, we aimed to gather data regarding various topics related to the following two major research targets:

1. Data storage and management
2. Data pipelines

In this report, we review some of our key research findings.

Research Target One: Data Storage and Management

Our research related to **data storage** and **management** centered around three key topic areas:

1. Data residency and cloud storage
2. Data security strategies and concerns
3. Data observability and sprawl management

Data Residency and Cloud Storage

We asked the following questions:

- *Where does your organization currently store its data and/or database solutions?*
- *Why is your organization migrating, or considering migrating, its databases to the cloud?*
- *Does your organization keep data in any of the following locations?*

Results:

Figure 1. Data storage prevalence: cloud vs. hybrid vs. on-prem [*n*=109]

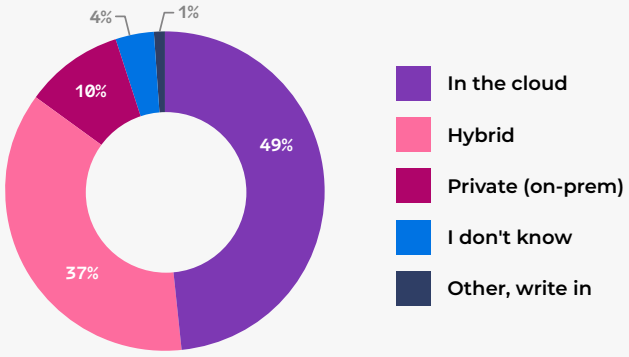


Figure 2. Reasons for cloud database migration [*n*=109]

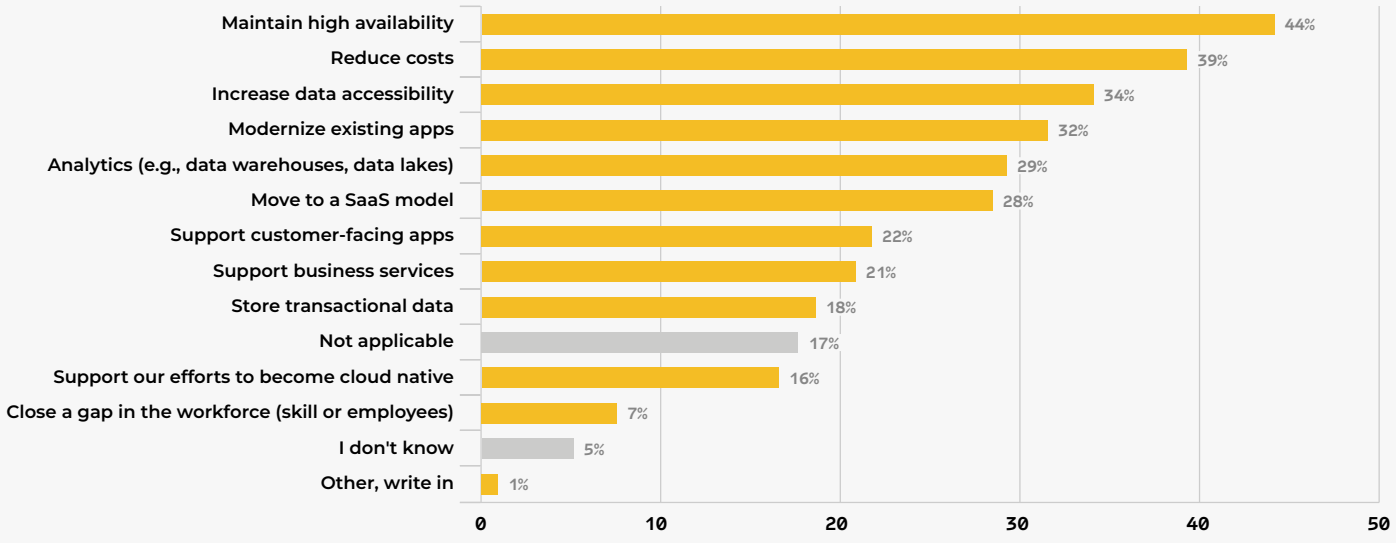


Table 1. Use of data warehouses, lakes, and lakehouses*

Location	Yes	No	I don't know	*n*=
Data warehouse	55%	28%	17%	102
Data lake	47%	36%	17%	104
Data lakehouse	27%	40%	33%	95

% of rows

OBSERVATIONS

A large majority of respondents (86%) reported that their organization uses some sort of cloud data storage, with nearly half of respondents saying their organization primarily stores data/database solutions in the cloud. Compared to the results of our 2024 Data Engineering survey, response rates for "In the cloud" rose sharply, while response rates for "Hybrid" and "Private" storage both decreased (additional details available in Table 2).

Segmenting the results by organization size, we found that respondents at small organizations were less likely than others to say their organization uses "Hybrid" data storage (details in Table 3).

80% of respondents selected at least one of the 11 reasons we provided for migrating databases to the cloud, and over half of respondents (52%) selected three or more. The most commonly selected reasons for cloud database migration were to "Maintain high availability," "Reduce costs," "Increase data accessibility," and "Modernize existing apps." Compared to last year's results, response rates for "Maintain high availability" and "Analytics (e.g., data warehouses, data lakes)" decreased, but there were no other significant year-over-year (YOY) changes (details available upon request).

Looking at the result subsets by respondents' organization size, we noted the following (full data available in Table 4):

- Respondents at **large organizations** were more likely than others to say their organization migrated to cloud databases to "Reduce costs," "Modernize existing apps," "Support... efforts to become cloud native," and for "Analytics."
- Respondents at **mid-sized organizations** were less likely than others to say their organization migrated to cloud databases to "Modernize existing apps" and "Support customer-facing apps."
- Respondents at **small organizations** were less likely than others to say their organization migrated to cloud databases to "Increase data accessibility."

Over half of respondents said that their organization stores data in data warehouses, and nearly half said their organization stores data in data lakes. Year over year, there were no significant changes in response rates for organizations' uses of data warehouses, data lakes, or data lakehouses.

Segmented by organization size, we observed the following (additional details in Table 5):

- Respondents at **large organizations** were more likely than others to say their organization stores data in data warehouses, data lakes, and data lakehouses.
- Respondents at **small organizations** were less likely than others to say their organization stores data in data warehouses.

ADDITIONAL TABLES

Table 2. Data storage prevalence: 2024–2025

Storage Type	2024	2025	Δ*
In the cloud	30%	49%	+19
Hybrid	48%	37%	-11
Private (on-premises)	20%	10%	-10
I don't know	2%	4%	+2
n=	109	110	

Change shown as percentage points

Table 3. Data storage prevalence by organization size*

Storage Type	Organization Size			Overall
	1-99	100-999	1,000+	
In the cloud	54%	55%	44%	49%
Hybrid	26%	40%	48%	37%
Private (on-premises)	14%	5%	8%	10%
I don't know	6%	0%	0%	4%
n=	35	20	48	108

**% of columns*

Table 4. Reasons for cloud database migration by organization size*

Reason	Organization Size			Overall
	1-99	100-999	1,000+	
Maintain high availability	40%	55%	48%	44%
Reduce costs	29%	25%	56%	39%
Increase data accessibility	29%	40%	40%	34%
Modernize existing apps	29%	15%	46%	32%
Analytics (e.g., data warehouses, data lakes)	20%	25%	42%	29%
Move to a SaaS model	31%	40%	25%	28%
Support customer-facing apps	26%	10%	27%	22%
Support business services	17%	25%	25%	21%
Store transactional data	20%	20%	19%	18%
Support our efforts to become cloud native	11%	5%	23%	16%
Close a gap in the workforce	14%	0%	6%	7%
Other, write in	0%	0%	2%	1%
Not applicable	17%	20%	8%	17%
I don't know	9%	0%	2%	5%
n=	35	20	48	109

**% of columns*

Table 5. Use of data warehouses, lakes, and lakehouses by organization size*

Location	Organization Size			Overall
	1-99	100-999	1,000+	
Data warehouse	37%	55%	67%	52%
Data lake	29%	25%	71%	45%
Data lakehouse	14%	15%	38%	24%
n=	35	20	48	108

**% of columns*

CONCLUSIONS

Cloud data storage appears to be the dominant model for most organizations, with a growing share now relying primarily on the cloud for storing data and database solutions. This shift away from hybrid and private storage approaches compared to last year suggests continuing movement toward cloud-native architectures.

Smaller organizations seem less likely to maintain hybrid environments, which may reflect simpler infrastructure needs or a desire to avoid the complexity of mixed deployment models.

Organizations commonly cite high availability, cost reduction, increased accessibility, and modernization as leading motivations for migrating databases to the cloud. Although interest in maintaining high availability and leveraging cloud-based analytics appears to have declined slightly compared to last year, most other drivers have remained stable. Larger organizations are especially likely to link cloud migration to broader strategic initiatives, such as modernization and cloud-native transformation, compared to mid-sized and small organizations, which may prioritize more immediate objectives.

Data warehouses and data lakes continue to be widely used for organizational storage needs, with no major YOY shifts in usage patterns. Larger organizations are more likely to adopt all three storage models — warehouses, lakes, and lakehouses — likely due to broader data requirements and platform diversity. In contrast, smaller organizations tend to rely less on data warehouses, possibly because of scale, staffing, or tooling limitations.

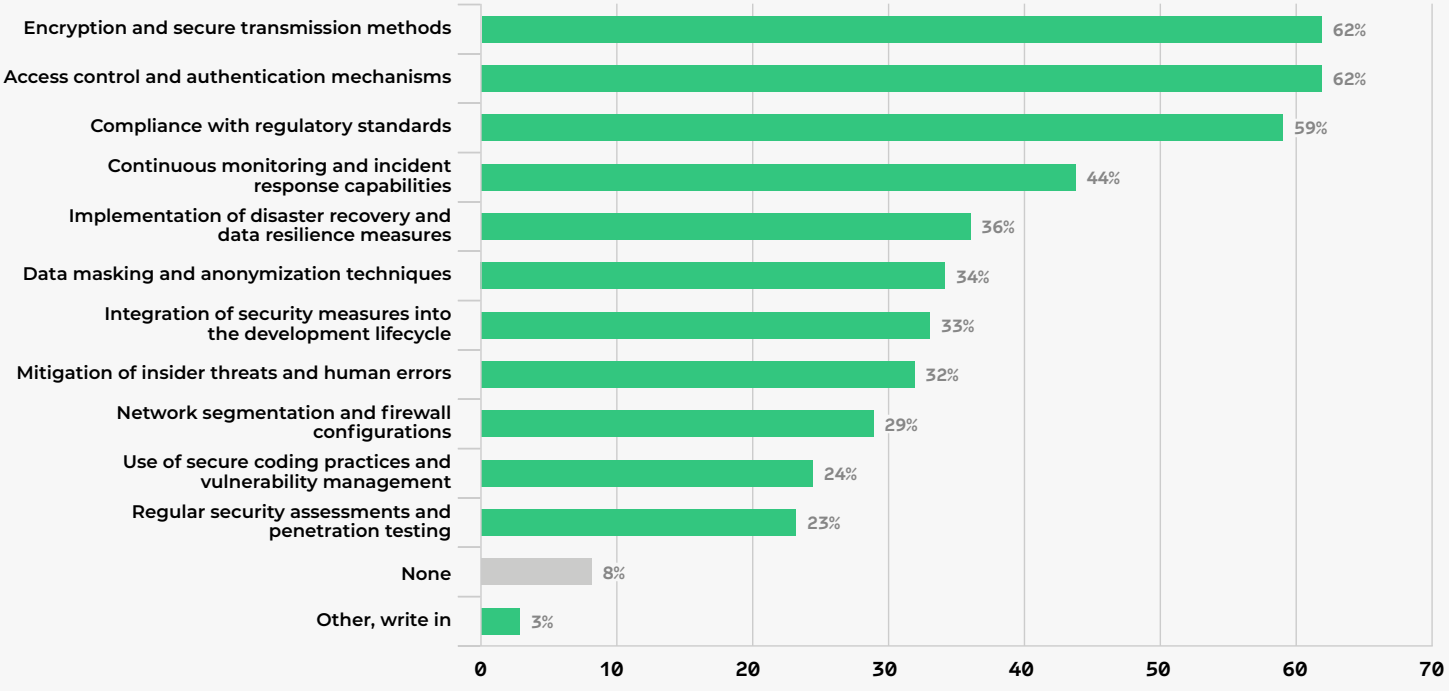
Data Security Strategies and Concerns

We asked the following questions:

- *What strategies and/or architectural considerations do you regularly examine for ensuring data security in the software you currently contribute to?*
- *Which of the following security concerns from the OWASP Data Security Top 10 do you believe to be significant potential threats to software you currently contribute to?*

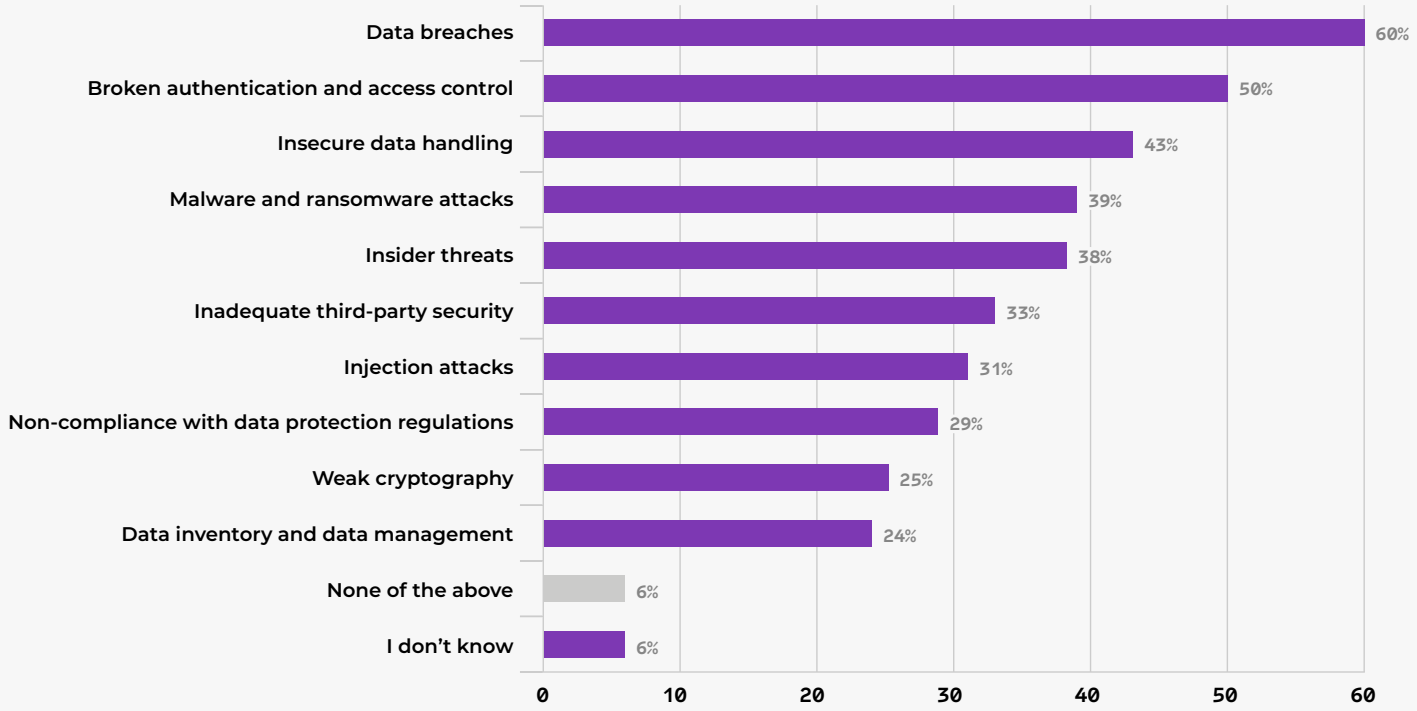
Results:

Figure 3. Strategies for ensuring data security [*n*=109]



SEE FIGURE 4 ON NEXT PAGE

Figure 4. OWASP security concerns [*n*=109]



OBSERVATIONS

A large majority of respondents (91%) said they use at least one strategy/architectural consideration for ensuring data security, and nearly half (47%) selected five or more of the 11 provided options. The most commonly chosen strategies were "Access control and authentication mechanisms," "Encryption and secure transmission methods," and "Compliance with regulatory standards."

Compared to our 2024 results, response rates for many strategies sharply declined, including "Access control and authentication mechanisms," "Encryption and secure transmission methods," "Implementation of disaster recovery and data resilience measures," and "Data masking and anonymization techniques" (details can be found in Table 6).

Looking at response patterns by organization size, we noted the following trends (further details in Table 7):

- Respondents at **mid-sized organizations** were more likely than others to say they use "Access control and authentication mechanisms," "Data masking and anonymization techniques," and "Regular security assessments and penetration testing."
- Respondents at **small organizations** were less likely than others to say they use "Compliance with regulatory standards," "Data masking and anonymization techniques," and "Integration of security measures into the development lifecycle."

87% of respondents expressed concern about at least one of the OWASP Data Security Top 10, and over half (53%) selected four or more. The most commonly selected security concerns were "Data breaches," "Broken authentication and access control," and "Insecure data handling." Concerns over "Insecure data handling," "Injection attacks," and "Data inventory and data management" all decreased significantly from last year's survey results (YOY details can be found in Table 8).

Segmented by organization size, we found the following (additional data in Table 9):

- Respondents at **mid-sized organizations** were more likely than others to have concerns over "Inadequate third-party security" and "Non-compliance with data protection regulations," and less likely than others to have concerns over "Broken authentication and access control" and "Injection attacks."
- Respondents at **small organizations** were more likely than others to express concerns over "Weak cryptography."

ADDITIONAL TABLES

Table 6. Strategies for ensuring data security: 2024–2025

Security Consideration	2024	2025	Δ*
Access control and authentication mechanisms	77%	62%	-15
Encryption and secure transmission methods	75%	62%	-13
Compliance with regulatory standards	66%	59%	-7
Continuous monitoring and incident response capabilities	50%	44%	-6
Implementation of disaster recovery and data resilience measures	58%	36%	-22
Data masking and anonymization techniques	55%	34%	-21
Integration of security measures into the SDLC	48%	33%	-15
Mitigation of insider threats and human errors	41%	32%	-9
Network segmentation and firewall configurations	44%	29%	-15
Use of secure coding practices and vulnerability management	46%	24%	-22
Regular security assessments and penetration testing	44%	23%	-21
Other, write in	4%	3%	-1
None	3%	8%	+5
n=	108	109	

**Change shown as percentage points*

Table 7. Strategies for ensuring data security by organization size*

	Organization Size			
Security Consideration	1-99	100-999	1,000+	Overall
Access control and authentication mechanisms	60%	90%	60%	62%
Encryption and secure transmission methods	71%	70%	60%	62%
Compliance with regulatory standards	60%	65%	63%	59%
Continuous monitoring and incident response capabilities	29%	60%	54%	44%
Implementation of disaster recovery and data resilience measures	40%	30%	40%	36%
Data masking and anonymization techniques	26%	50%	38%	34%
Integration of security measures into the SDLC	23%	40%	42%	33%
Mitigation of insider threats and human errors	26%	45%	35%	32%
Network segmentation and firewall configurations	26%	35%	33%	29%
Use of secure coding practices and vulnerability management	26%	20%	27%	24%
Regular security assessments and penetration testing	20%	45%	19%	23%
Other, write in	3%	0%	2%	3%
None	6%	0%	4%	8%
n=	35	20	48	109

**% of columns*

SEE TABLE 8 ON NEXT PAGE

Table 8. OWASP security concerns: 2024–2025

Security Concern	2024	2025	Δ*
Data breaches	60%	60%	0
Broken authentication and access control	46%	50%	+4
Insecure data handling	63%	43%	-20
Malware and ransomware attacks	46%	39%	-7
Insider threats	35%	38%	+3
Inadequate third-party security	35%	33%	-2
Injection attacks	42%	31%	-11
Non-compliance with data protection regulations	35%	29%	-6
Weak cryptography	24%	25%	+1
Data inventory and data management	41%	24%	-17
I don't know	6%	6%	0
None of the above	2%	6%	+4
n=	105	109	

**Change shown as percentage points*

Table 9. OWASP security concerns by organization size*

	Organization Size			
Security Concern	1-99	100-999	1,000+	Overall
Data breaches	66%	70%	56%	60%
Broken authentication and access control	60%	40%	52%	50%
Insecure data handling	43%	40%	48%	43%
Malware and ransomware attacks	37%	45%	42%	39%
Insider threats	43%	45%	33%	38%
Inadequate third-party security	31%	45%	31%	33%
Injection attacks	40%	20%	31%	31%
Non-compliance with data protection regulations	29%	45%	25%	29%
Weak cryptography	40%	20%	17%	25%
Data inventory and data management	26%	35%	19%	24%
I don't know	3%	0%	10%	6%
None of the above	6%	0%	2%	6%
n=	35	20	48	109

**% of columns*

CONCLUSIONS

Most organizations seem to use multiple strategies to ensure data security, though the overall adoption of key security practices (such as access control, encryption, disaster recovery, and data anonymization) appears to have declined compared to the previous year. This may reflect shifts in organizational priorities, increasing reliance on default security measures from cloud providers, or consolidation of tools and responsibilities. Developers tend to be most concerned about data breaches, broken authentication, and insecure data handling. Declining concerns over insecure handling, injection attacks, and data inventory issues may suggest growing confidence in handling these risks or a shift in focus toward other threat areas.

Data Observability and Sprawl Management

We asked the following questions:

- *Which of the following tools does your organization use for data observability?*
- *What strategies/tools do you use to manage data duplication, fragmentation, or sprawl in your data architecture?*

Results:

Figure 5. Use of data observability tools [*n*=107]

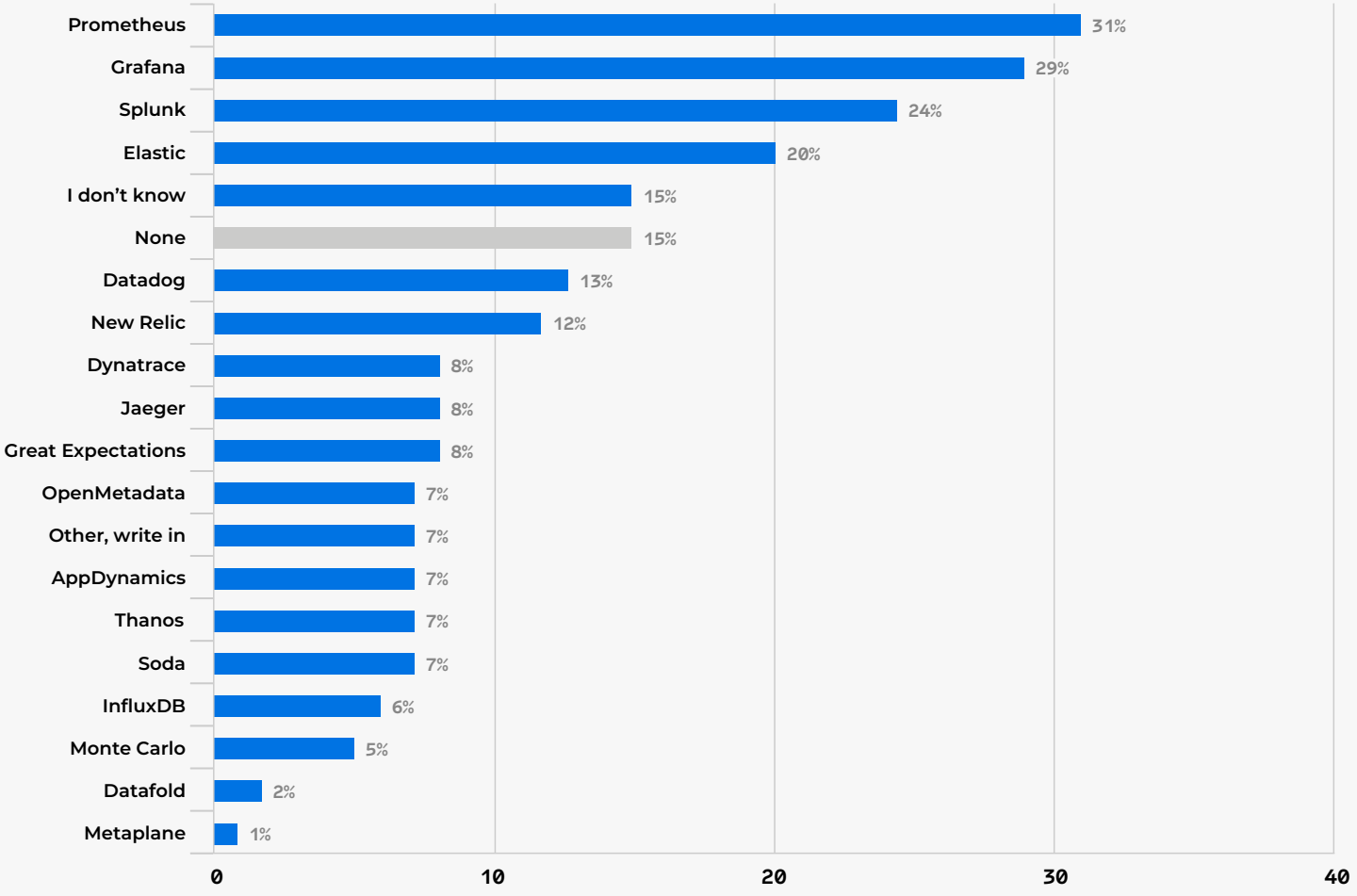
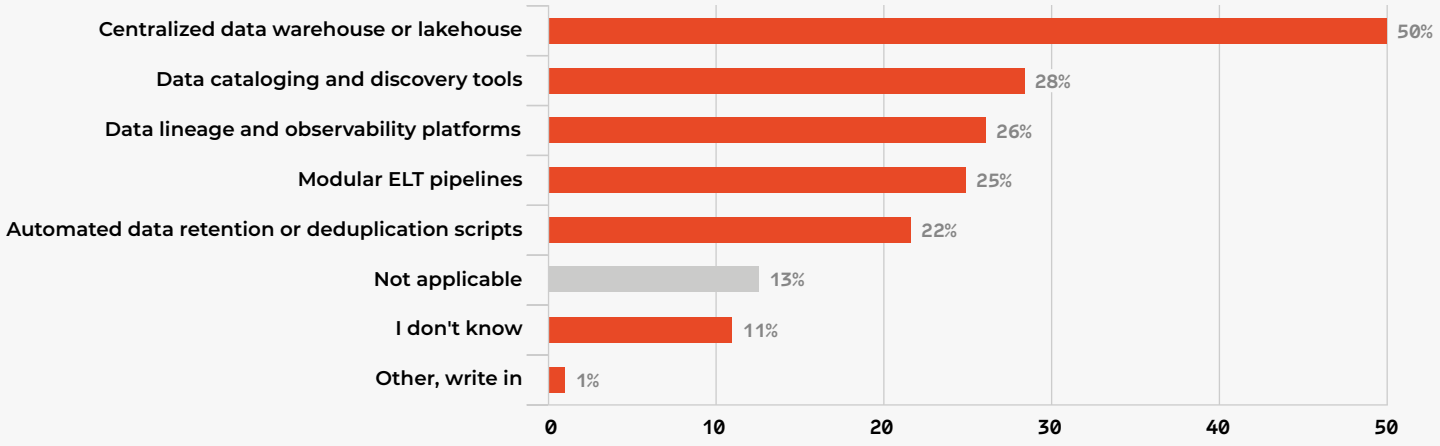


Figure 6. Strategies/tools for managing data [*n*=107]



OBSERVATIONS

About two-thirds of respondents (66%) said their organization uses one or more of the data observability tools we listed — a significant decrease from 81% last year. However, the relative number of respondents who selected two or more tools (51%) was nearly the same as in 2024 (52%). The only specific tools with significantly different response rates YOY were "Grafana" and "Datadog," both of which fell by 12 percentage points (additional details available on request).

By organization size, we observed the following trends (see Table 10 for details):

- Respondents at **large organizations** were more likely than others to say their organization uses the top three data observability tools: "Prometheus," "Grafana," and "Splunk."
- Respondents at **mid-sized organizations** were more likely than others to say their organization uses "Datadog."
- Respondents at **small organizations** were more likely than others to say their organization does not use any data observability tools. They were less likely than others to select "Prometheus" and "New Relic."

Over three-quarters of respondents (76%) said they use at least one of the five listed strategies/tools for managing data duplication, fragmentation, or sprawl in their data architecture, and nearly half (45%) selected two or more. The most commonly selected strategy for this kind of data management, by a large margin, was the use of a "Centralized data warehouse or lakehouse." We found that respondents at mid-sized organizations were more likely than others to say their organization uses "Data lineage and observability platforms." Respondents at small organizations were more likely to say their organization uses "Automated data retention or deduplication scripts," and less likely to say their organization uses "Data lineage and observability platforms." Additional data segmented by organization size can be found in Table 11.

ADDITIONAL TABLES

Table 10. Use of data observability tools by organization size*

Tool	Organization Size			Overall
	1-99	100-999	1,000+	
Prometheus	14%	32%	46%	31%
Grafana	20%	26%	40%	29%
Splunk	20%	16%	33%	24%
Elastic	14%	26%	23%	20%
Datadog	11%	32%	8%	13%
New Relic	3%	16%	19%	12%
Dynatrace	6%	5%	13%	8%
Jaeger	6%	5%	13%	8%
Great Expectations	11%	11%	6%	8%
AppDynamics	3%	0%	13%	7%
Thanos	6%	0%	10%	7%
Soda (Soda Cloud/Soda SQL)	11%	5%	4%	7%
OpenMetadata	11%	5%	6%	7%
InfluxDB	6%	5%	6%	6%
Monte Carlo	11%	0%	2%	5%
Datafold	6%	0%	0%	2%
Metaplane	0%	0%	2%	1%
Other, write in	9%	11%	6%	7%
I don't know	14%	16%	13%	15%
None	26%	0%	8%	15%
n=	35	19	48	107

**% of columns*

Table 11. Strategies/tools for managing data by organization size*

Strategy	Organization Size			Overall
	1-99	100-999	1,000+	
Centralized data warehouse or lakehouse	50%	50%	54%	50%
Data cataloging and discovery tools	29%	25%	31%	28%
Data lineage and observability platforms	18%	40%	29%	26%
Modular ELT pipelines	24%	20%	31%	25%
Automated data retention or deduplication scripts	38%	10%	19%	22%
Other, write in	3%	0%	0%	1%
I don't know	6%	10%	17%	11%
Not applicable	18%	10%	2%	13%
n=	34	20	48	107

**% of columns*

CONCLUSIONS

Tooling for data observability is still widely used, though total adoption has dropped — possibly due to cost constraints or platform consolidation — while large organizations continue to favor mature observability stacks with tools like Prometheus, Grafana, and Splunk. Centralized data warehouses and lakehouses remain the most common strategy for managing data duplication and sprawl, with organizations differing in their secondary approaches based on size and infrastructure maturity.

Research Target Two: Data Pipelines

Our research related to **data pipelines** centered around three key topic areas:

1. ETL workload and planning
2. Pipeline architecture and integration
3. Preparing and maintaining data for AI/ML

ETL Workload and Planning

We asked the following questions:

- *What percent of your time at work is spent on data extraction, loading, and/or transformation (ELT, ETL)?*
- *How often do you extract, transform, and load data using each of the following approaches?*
- *When are you planning to use an ETL and/or reverse ETL solution?*

Results:

Figure 7. Time spent on ETL/ELT [*n*=91]

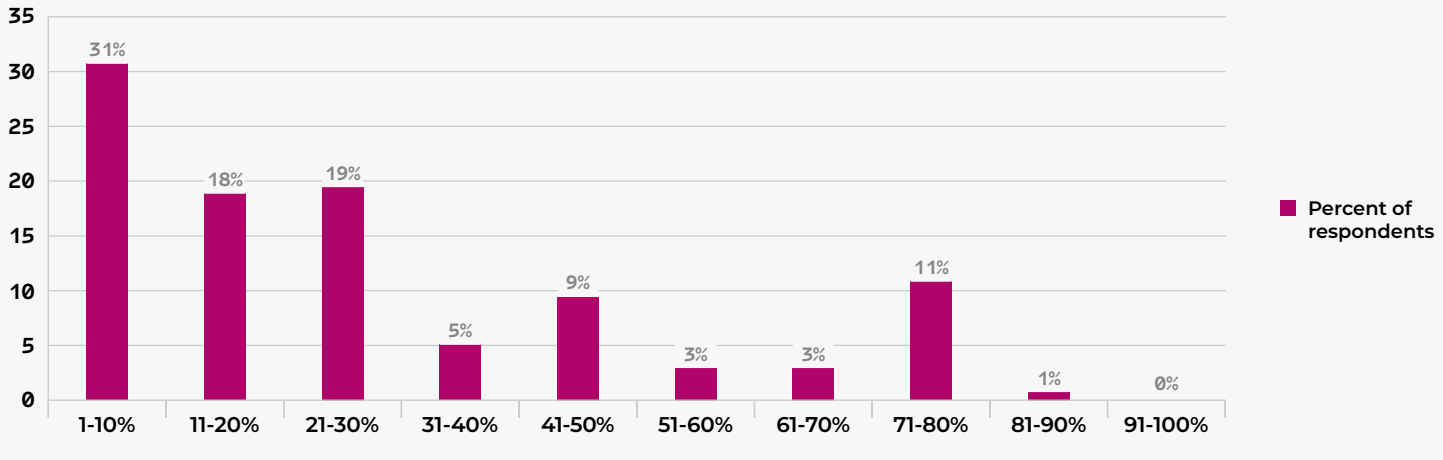


Figure 8. Plans for ETL solution use [*n*=107]

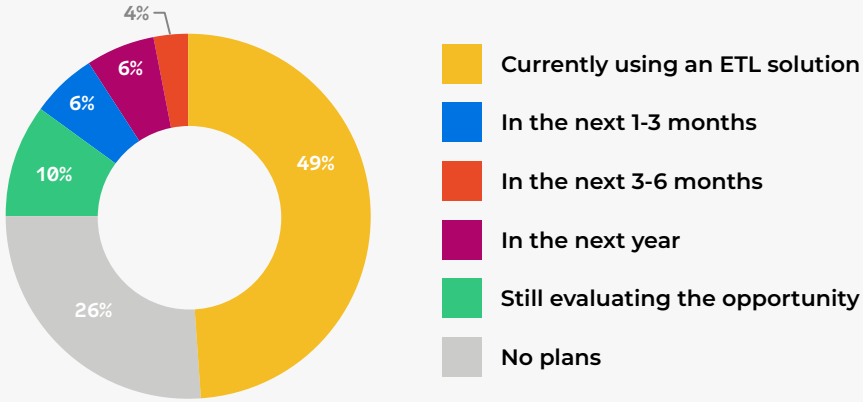
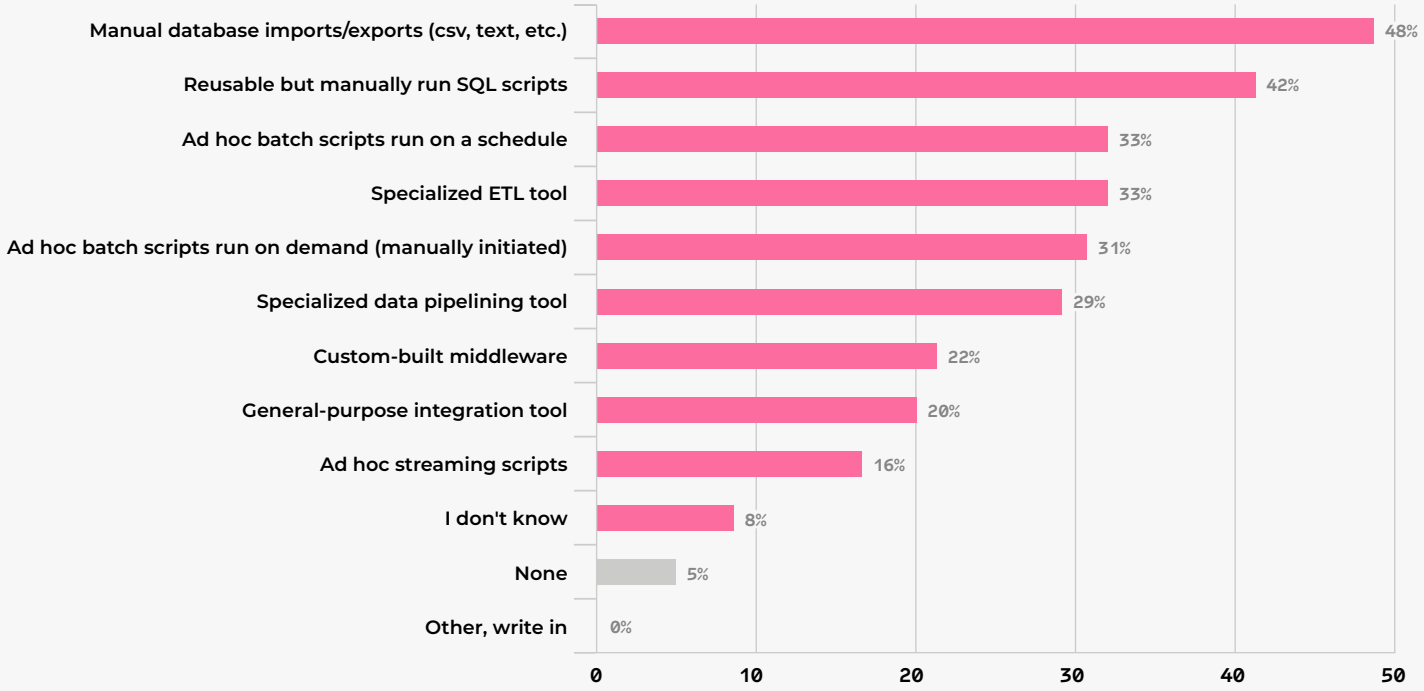


Figure 9. ETL/ELT approaches [*n*=107]



OBSERVATIONS

Three-quarters of respondents claimed to spend at least 10% of their time at work on ETL processes, and nearly half of respondents (47%) said they spent at least 30% of their time on ETL. On average, respondents said they spent 30% of their time working on ETL, with a median response of 25%. Respondents at mid-sized organizations, on average, said they spent only 20% of their time at work on data ETL compared to 33% of respondents at small organizations and 35% at large organizations.

87% of respondents selected at least one of the ETL approach options we gave, and over half of respondents (54%) selected three or more. Compared to the results of our 2024 survey, the only significant change in response rates was an 11% decline in "Specialized ETL tool" use (details available on request).

Segmented by organization size, we noted the following (additional data can be found in Table 12):

- Respondents at **large organizations** were more likely than others to say they use "Specialized ETL tool[s]."
- Respondents at **mid-sized organizations** were less likely than others to say they use "Reusable but manually run SQL scripts."

- Respondents at **small organizations** were more likely than others to say they use "Manual database imports/exports" and "Ad hoc batch scripts run on demand." They were less likely to say they use "Specialized data pipelining tool[s]."

About half of respondents said they are "Currently using an ETL solution," and another 16% said they are planning on adopting one within the next year. Around a quarter of respondents said they have no plans to use an ETL solution. There were no significant differences in response rates regarding ETL solution adoption compared to 2024 (details available on request).

Respondents at large organizations were much more likely than others to report they are currently using an ETL solution; those at small organizations were less likely than others to say they use an ETL solution and more likely to say they have no plans to do so (additional data in Table 13).

We also found that respondents with "No plans" to adopt an ETL solution spent much less of their time, on average, working on ETL than others (details in Table 14).

ADDITIONAL TABLES

Table 12. Frequency of ETL/ELT approaches by organization size*

Approach	Organization Size			Overall
	1-99	100-999	1,000+	
Manual database imports/exports (csv, text, etc.)	66%	45%	40%	48%
Reusable but manually run SQL scripts	51%	25%	47%	42%
Ad hoc batch scripts run on a schedule	29%	40%	34%	33%
Specialized ETL tool	29%	30%	40%	33%
Ad hoc batch scripts run on demand (manually initiated)	43%	25%	26%	31%
Specialized data pipelining tool	23%	35%	34%	29%
Custom-built middleware	29%	15%	23%	22%
General-purpose integration tool	29%	20%	15%	20%
Ad hoc streaming scripts	23%	15%	13%	16%
Other, write in	0%	0%	0%	0%
I don't know	6%	5%	9%	8%
None	3%	0%	4%	5%
n=	35	20	47	107

**% of columns*

Table 13. Plans for ETL solution use by organization size*

ETL Solution Plans	Organization Size			Overall
	1-99	100-999	1,000+	
Currently using an ETL solution	29%	50%	67%	49%
In the next 1-3 months	6%	10%	4%	6%
In the next 3-6 months	6%	5%	2%	4%
In the next year	9%	15%	0%	6%
Still evaluating the opportunity	18%	5%	8%	10%
No plans	32%	15%	19%	26%
n=	34	20	48	107

**% of columns*

Table 14. Average time spent on ETL based on ETL solution plans

ETL Solution Plans	Avg. Time	*n*=
Currently using an ETL solution	34%	52
In the next 1-12 months	28%	16
Still evaluating the opportunity	24%	11
No plans	10%	28

CONCLUSIONS

Most developers appear to invest a meaningful portion of their time in ETL processes, with typical workloads averaging 30% of total work time. Time spent on ETL varies by organization size, with large and small organizations reporting higher averages than mid-sized ones. This may reflect differences in staffing models, the complexity of internal data systems, or the maturity of automated workflows.

Use of diverse ETL approaches remains widespread, with most organizations combining multiple strategies. While overall usage patterns have remained relatively stable since 2024, reliance on specialized ETL tools has dropped, possibly due to cost constraints, integration challenges, or a shift toward lighter-weight, code-based solutions. Larger organizations continue to lead in adoption of specialized tools, while smaller organizations lean more heavily on manual or ad hoc methods — likely reflecting differences in scale, tooling budgets, or process maturity.

Current adoption of ETL solutions seems to be currently concentrated in larger organizations. In contrast, smaller organizations are more likely to have no plans for formal ETL adoption, and these same organizations tend to dedicate less time overall to ETL work, suggesting that for some teams, low ETL adoption is a function of both limited need and restricted resources.

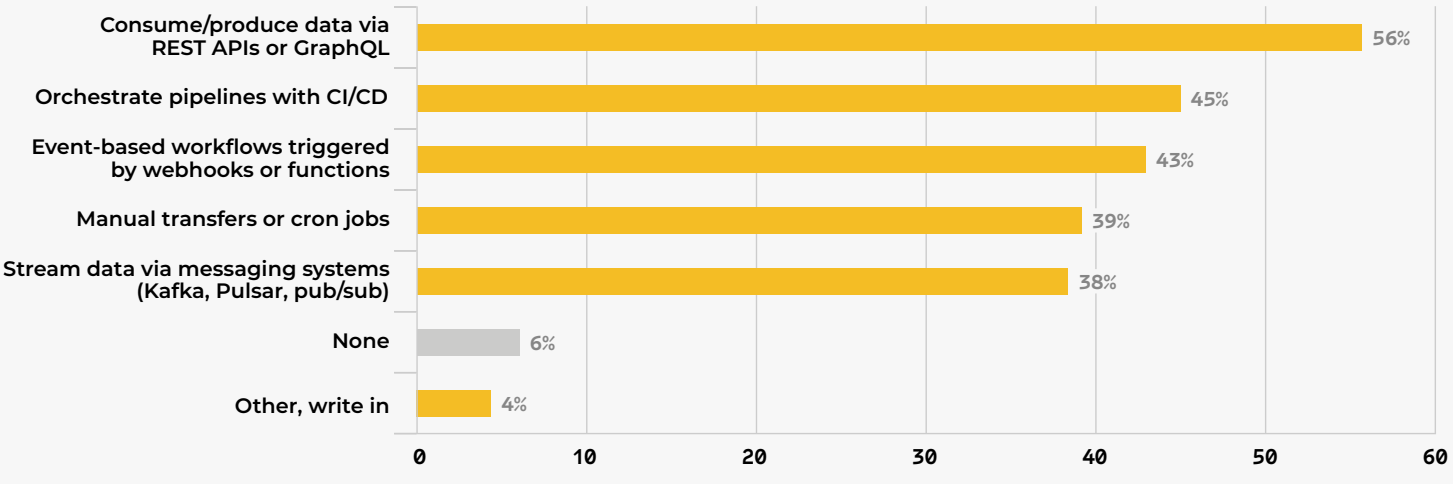
Pipeline Architecture and Integration

We asked the following questions:

- *How do you integrate or orchestrate data flows between systems in your current stack?*
- *Which DataOps practices have you implemented in your data pipelines or workflows?*

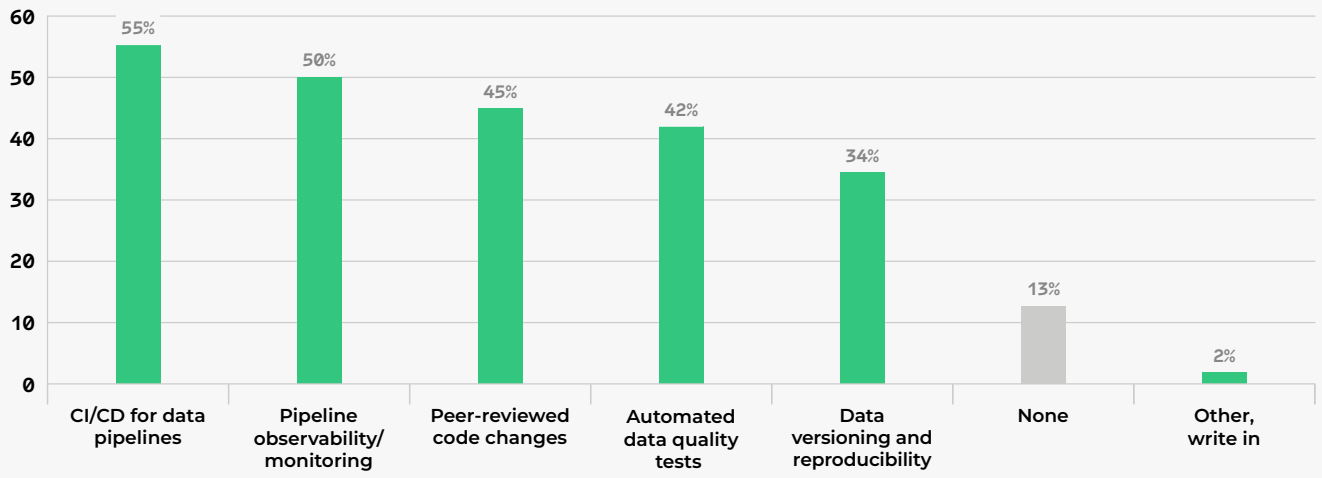
Results:

Figure 10. Methods of data flow integration/orchestration [*n*=110]



SEE FIGURE 11 ON NEXT PAGE

Figure 11. DataOps practices implemented in data pipelines/workflows [*n*=112]



OBSERVATIONS

Most respondents (91%) selected at least one of the five methods of data-flow integration/orchestration we provided, and almost two-thirds (65%) selected two or more. The most popular method of integrating/orchestrating data flows between systems was to "Consume/produce data via REST APIs or GraphQL." Segmented by organization size, we found the following (details in Table 15):

- Respondents at **large organizations** were more likely than others to say they "Consume/produce data via REST APIs or GraphQL."
- Respondents at **mid-sized organizations** were more likely than others to say they use "Event-based workflows triggered by webhooks/functions."
- Respondents at **small organizations** were less likely than others to say they use "Manual transfers or cron jobs" and "Stream data via messaging systems."

87% of respondents said they have implemented at least one of the five DataOps practices we suggested, and about two-thirds (65%) said they have implemented two or more. The most commonly selected practice was "CI/CD for data pipelines."

We found that respondents at mid-sized organizations were more likely than others to say they have implemented "Pipeline observability/monitoring," and less likely to say they have implemented "Data versioning and reproducibility." Respondents at small organizations were less likely than others to say they have implemented "CI/CD for data pipelines."

ADDITIONAL TABLES

Table 15. Methods of data flow integration/orchestration by organization size*

Method	Organization Size			Overall
	1-99	100-999	1,000+	
Consume/produce data via REST APIs or GraphQL	54%	55%	67%	56%
Orchestrate pipelines with CI/CD	34%	50%	56%	45%
Event-based workflows triggered by webhooks/functions	40%	60%	44%	43%
Manual transfers or cron jobs	49%	40%	35%	39%
Stream data via messaging systems	20%	45%	52%	38%
Other, write in	0%	5%	4%	4%
None	6%	0%	2%	6%
n=	35	20	48	110

**% of columns*

Table 16. DataOps practices implemented in data pipelines/workflows by organization size*

Practice	Organization Size			Overall
	1-99	100-999	1,000+	
CI/CD for data pipelines	40%	60%	69%	55%
Pipeline observability/monitoring	49%	65%	52%	50%
Peer-reviewed code changes	43%	50%	50%	45%
Automated data quality tests	46%	50%	44%	42%
Data versioning and reproducibility	37%	25%	40%	34%
Other, write in	0%	0%	2%	2%
None	14%	10%	6%	13%
n=	35	20	48	112

**% of columns*

CONCLUSIONS

Data-flow integration and orchestration practices appear to be well established across most organizations, with REST APIs and GraphQL unsurprisingly emerging as the most common methods for connecting systems. Use of multiple integration strategies is also common, particularly in larger organizations, which may reflect the complexity and scale of their data ecosystems. Event-based workflows seem to be favored by mid-sized organizations, while smaller organizations are less likely to rely on CI/CD pipelines and streaming approaches — possibly due to differences in technical capacity or volume of data exchange.

DataOps practices are similarly widespread: Most companies have adopted at least one, while many have implemented multiple strategies. CI/CD for data pipelines is the most prevalent practice overall, suggesting a growing emphasis on automation and reliability in data engineering workflows. Small organizations are more likely to still be evaluating or ramping up their CI/CD, likely due to differences in infrastructure maturity, staffing, or organizational priorities.

Preparing and Maintaining Data for AI/ML

We asked the following questions:

- *How do you currently supply data or context to support generative or agentic AI applications?*
- *How confident are you in your understanding of best practices for preparing clean, high-quality data for AI/ML systems?*
- *Which of the following practices do you use to ensure data quality when preparing datasets for AI or ML models?*

Results:

Figure 12. Confidence in understanding AI/ML data best practices [*n*=108]

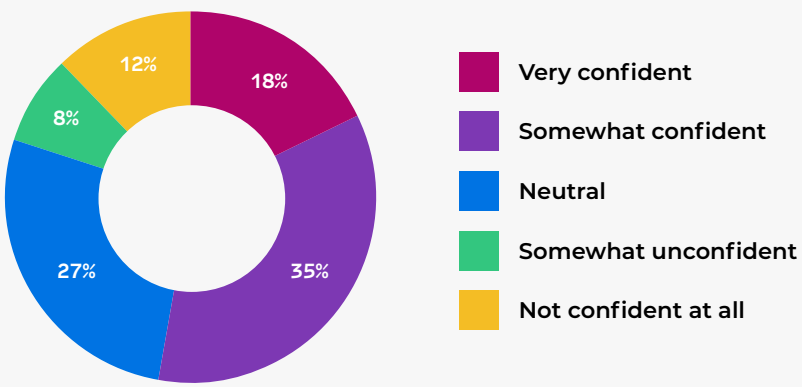


Figure 13. Methods for supplying data to generative/agentic AI apps [*n*=107]

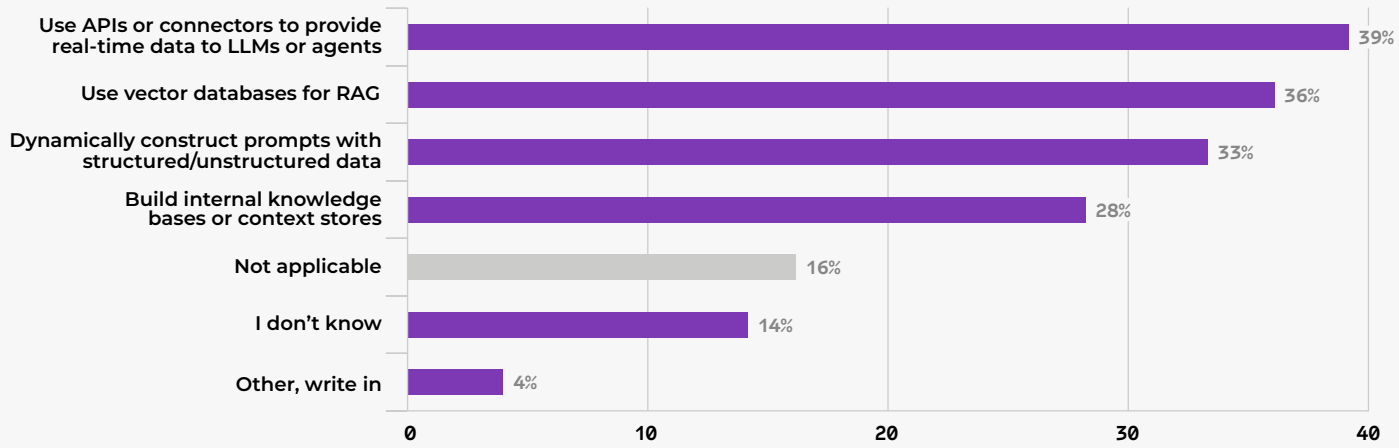
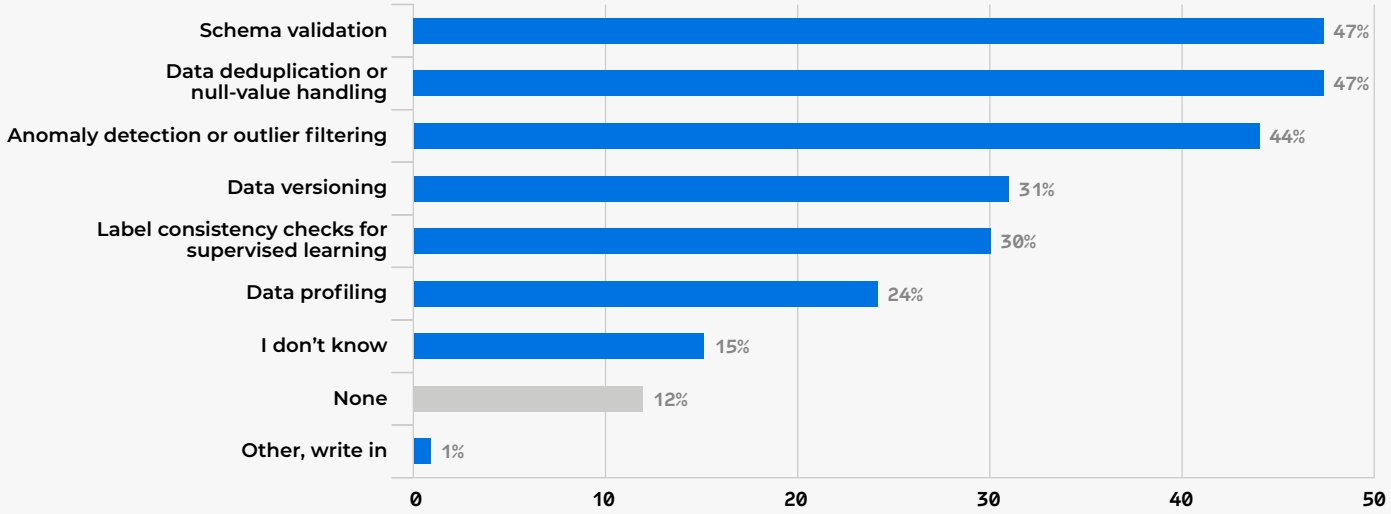


Figure 14. Practices for ensuring data quality for AI/ML models [*n*=108]



OBSERVATIONS

Just over half of respondents (53%) reported feeling at least "Somewhat confident" in their understanding of best practices for preparing high-quality data for AI/ML systems, and only one-fifth of respondents said they feel "Somewhat unconfident" or "Not confident at all." Respondents at mid-sized organizations were less likely than others to say they felt "Somewhat confident" in their understanding, while respondents at small organizations were less likely than others to say they felt "Very confident" (details available in Table 17).

Two-thirds of respondents said they use at least one of the four provided methods for supplying data to generative/agentic AI applications, the most commonly selected of which was the "Use [of] APIs or connectors to provide real-time data." Respondents at large organizations were less likely than others to say they "Dynamically construct prompts," and respondents at mid-sized organizations were less likely than others to say they "Use APIs or connectors to provide real-time data" (further details can be found in Table 18).

Nearly three-quarters of respondents (73%) said they use at least one of the six AI data quality practices we suggested, and almost half said they use three or more. "Schema validation," "Data deduplication or null-value handling," and "Anomaly detection or outlier filtering" were the most commonly selected practices.

Segmenting the results by organization size, we noted the following (details in Table 19):

- Respondents at **large organizations** were more likely than others to say they practice "Anomaly detection or outlier filtering."

- Respondents at **mid-sized organizations** were less likely than others to say they use "Data deduplication or null-value handling" and "Data versioning."
- Respondents at **small organizations** were more likely than others to say they don't use any AI-data quality practices, and less likely than others to say they "Label consistency checks for supervised learning." They were also less likely to respond "I don't know."

ADDITIONAL TABLES

Table 17. Confidence in understanding AI/ML data best practices by organization size*

Confidence Level	Organization Size			Overall
	1-99	100-999	1,000+	
Very confident	9%	25%	21%	18%
Somewhat confident	40%	25%	40%	35%
Neutral	26%	35%	25%	27%
Somewhat unconfident	11%	5%	8%	8%
Not confident at all	14%	10%	6%	12%
n=	35	20	48	108

% of columns

Table 18. Methods for supplying data to generative/agentic AI apps by organization size*

Method	Organization Size			Overall
	1-99	100-999	1,000+	
Use APIs or connectors to provide real-time data	39%	40%	25%	49%
Use vector databases for RAG	36%	40%	40%	36%
Dynamically construct prompts	33%	40%	40%	28%
Build internal knowledge bases or context stores	28%	37%	20%	28%
Other, write in	4%	0%	10%	4%
I don't know	14%	11%	10%	17%
Not applicable	16%	20%	15%	6%
n=	107	35	20	47

% of columns

Table 19. Practices for ensuring data quality for AI/ML models by organization size*

Practice	Organization Size			Overall
	1-99	100-999	1,000+	
Schema validation	51%	50%	48%	47%
Data deduplication or null-value handling	51%	35%	54%	47%
Anomaly detection or outlier filtering	40%	40%	54%	44%
Data versioning	37%	20%	35%	31%
Label consistency checks for supervised learning	23%	35%	35%	30%
Data profiling	31%	15%	25%	24%
Other, write in	3%	0%	0%	1%
I don't know	6%	20%	19%	15%
None	20%	5%	2%	12%
n=	35	20	48	108

% of columns

CONCLUSIONS

Confidence in preparing high-quality data for AI/ML systems appears to be moderate overall. While a majority of developers seem to have at least some grasp of best practices, relatively few report high confidence, suggesting a widespread need for deeper expertise or more formalized processes as AI/ML is incorporated into more and more applications. When it comes to integrating data with generative or agentic AI applications, most developers appear to be experimenting with at least one method. Real-time data delivery via APIs or connectors is the most common strategy, indicating a strong emphasis on keeping AI systems responsive and current.

AI-focused data quality practices are becoming broadly adopted, particularly those tied to schema enforcement, cleansing, and anomaly detection. These methods reflect foundational efforts to improve input data reliability for AI systems. Larger organizations appear to lead in implementing anomaly filtering, while mid-sized and smaller organizations may be slower to adopt other critical practices such as deduplication, versioning, or consistency checks. These patterns may highlight disparities in maturity or prioritization across organization types.

Future Research

Our analysis here only touched the surface of the available data, and we will look to refine and expand our data engineering research as we produce future Trend Reports. Please contact publications@dzone.com if you would like to discuss any of our findings or supplementary data.

G. Ryan Spain

@grspain
@grspain
@grspain
gryanspain.com

G. Ryan Spain lives on a beautiful two-acre farm in McCalla, Alabama with his lovely wife. He is a polyglot software engineer with an MFA in poetry, a die-hard Emacs fan and Linux user, a lover of The Legend of Zelda, a journeyman data scientist, and a home cooking enthusiast. When he isn't programming, he can often be found playing Blue Prince with a glass of red wine or a cold beer.

CONTRIBUTOR INSIGHTS

Data Lake, Warehouse, or Lakehouse?

Rethinking the Future of Data Architecture

By Miguel Garcia Lorenzo, VP of Engineering at Factorial

In the age of AI and ubiquitous data, the lines between traditional data architectures are blurring. Data lakes, warehouses, and lakehouses are no longer isolated strategies but are increasingly converging into unified, intelligent platforms. This article explores how modern data architectures are evolving to meet new demands for real-time insights, agility, and a single source of truth. Over the past five years, the shift from application-centric to data-centric thinking has transformed the technology landscape. Whether in SaaS, enterprises, or small businesses, data has become the central conversation among tech leaders. The longstanding divide between operational systems and analytical platforms is rapidly becoming obsolete.

While traditional architectures still have their place, the continued advancements in AI are accelerating the need for solutions that unify data access and interpretation. In today's world, any modern system must be a data-first system. Some industry voices even suggest that large language models (LLMs) could eventually replace many SaaS solutions. While that may be a stretch, it's clear that the next generation of technology must deeply integrate with — and be powered by — data.

Data Architectures Explained: Lake, Warehouse, and Lakehouse

Modern data architectures vary in purpose, flexibility, and technical depth. Understanding the differences between data warehouses, lakes, and lakehouses is key to choosing the right foundation for scalable and intelligent systems.

- **Data warehouses** are structured, reliable systems designed for analytics at scale. They provide strong performance and mature tooling and work best with structured data.
- **Data lakes** offer flexible, low-cost storage for raw data of all types. They are ideal for machine learning (ML), batch processing, and experimentation, but they often require significant engineering effort to become performant.
- **Data lakehouses** aim to unify the strengths of both, combining the cost efficiency of lakes with the performance and governance of warehouses. They represent the evolution toward a single, unified data platform.

Here's a quick comparison of the three:

Table 1. Data architecture overview

Aspect	Data Warehouse	Data Lake	Data Lakehouse
Data types	Structured data	Structured, semi-structured, unstructured	Structured, semi-structured, unstructured
Storage	Usually proprietary or abstracted	Object storage (e.g., S3, ADLS)	Object storage with table formats
Engines	Usually proprietary (e.g., Snowflake, BigQuery, Redshift, ClickHouse)	Spark, Flink, Hive, etc.	Spark, Trino, etc.
Out-of-the-box features	Clustering, caching, materialized views, time travel	Depends on engine and format	Depends on engine and format; some offer ACID, time travel, schema evolution, and caching
Customization effort	Low – managed experience	High – requires significant engineering effort	Medium – balance between control and out-of-the-box functionality
Real-time and query latency	Strong – built for fast, low-latency queries	Weak – not suitable for interactive or real-time needs	Improving but still a challenge depending on engine and freshness requirements
Governance and ACID	Strong	Weak	Strong (with Delta, Iceberg, Hudi)
BI integration	High	Low	Improving rapidly; depends on engine/tooling
Best use cases	Teams needing fast insights with minimal effort, in real time, and with low latency	Data scientists and engineers working with diverse raw data	Organizations looking for a single platform to serve multiple analytical needs

Modern Needs: Real-Time, AI-Driven, and Open Formats

To meet the needs of evolving data architecture, modern systems are moving toward a shared foundation: a **single, synchronized layer of truth** built on cloud object storage. Ingestion, both batch and real-time, is no longer a technical challenge. The next frontier lies in how we expose this data through flexible, efficient, and composable compute layers. That's where the real differentiation happens.

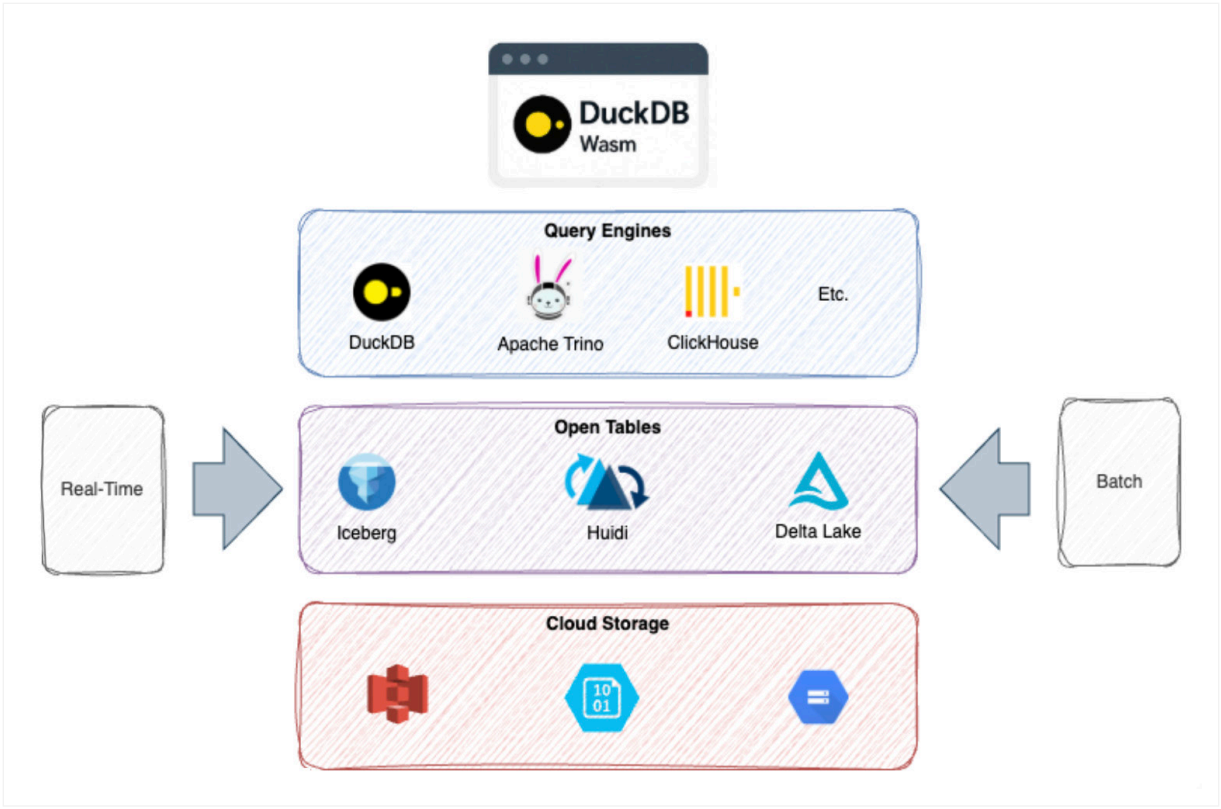
A powerful shift is underway: Instead of relying on centralized, monolithic data warehouses, modern architectures are embracing a **multi-engine, multi-modal design**, where each query engine plays a specific role — from heavy lifting in the cloud to lightweight analytics in the browser. Cloud object storage, combined with open table formats like Apache Iceberg, serves as the foundation that unifies these engines, offering transactional consistency, schema evolution, and performance optimizations.

In this section, we'll explore:

- How open table formats enable this unified foundation by decoupling data storage from compute logic
- How the emergence of distributed and embedded query engines (e.g., Trino, ClickHouse, DuckDB) is redefining where and how analytics happens — bringing data closer to the user than ever before

This model not only supports modern needs like AI and low-latency analytics but also encourages a new mindset: one where compute is no longer centralized but is composable, contextual, and optimized for the workload.

Figure 1. Data architecture



Open Table Formats as the Enabler

As data architectures evolve toward lakehouses and real-time analytics, open table formats have emerged as a critical component of scalable, maintainable system and multi-engine architecture. An open table format defines how to organize, manage, and track large datasets stored across files in cloud object storage. It adds a metadata layer on top of open files (Parquet, AVRO, or ORC), allowing query engines to treat a directory of files as a single, versioned, and transactional table.

An open file format defines how the data is stored inside a file (structure, encoding, or compression), and an open table format defines how *many* files are organized and managed together as a logical, versioned, and queryable table.

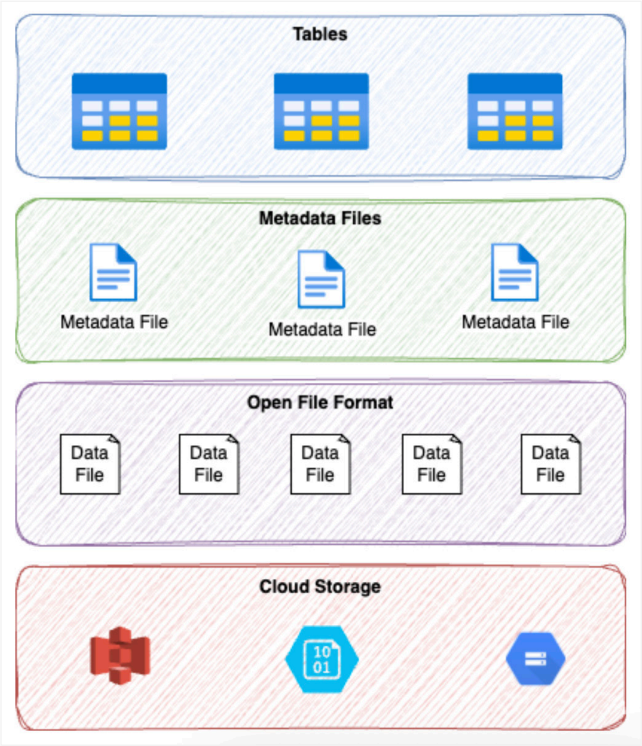
Open table formats support key features such as:

- ACID transactions – support insert/update/delete
- Schema evolution – change structure without rewriting everything
- Partitioning and pruning – enable faster queries
- Time travel – query past versions of your data
- Concurrent writes and reads – allows multiple engines to safely read/write in parallel

Rather than being tied to a specific vendor, open table formats allow data to be read and written by many engines, including Spark, Trino, Flink, Dremio, Snowflake, DuckDB, and Athena. The most widely adopted open table formats today are Apache Iceberg, Delta Lake, and Apache Hudi. Apache Iceberg is quickly becoming the standard due to its:

- **Engine neutrality** – works seamlessly with Trino, Flink, Spark, Snowflake, and more
- **Hidden partitioning** – simplifies queries and avoids user-managed partitions
- **Robust metadata management** – scales to petabyte-level datasets efficiently with snapshot isolation and manifest pruning
- **Better separation of metadata and data** – enables high performance and fewer conflicts during concurrent writes

Figure 2. Open formats



Unlike Delta Lake, Iceberg is not tied to a specific vendor, and unlike Hudi, it balances batch and streaming use cases more elegantly.

The Role of Query Engines and Single-Instance Architecture

With storage decoupled from the compute layer, organizations are no longer locked into a single analytics engine. Instead, modern architectures promote a **multi-engine strategy** where each tool plays a precise role: Trino for federation, ClickHouse for speed, and DuckDB for embedded insights. What ties them together is a common open table format — the Iceberg table — acting as the single, trusted interface for all compute layers.

Instead of centralizing all analytics in massive data warehouse clusters, modern architectures are leaning toward dynamic, on-demand compute layers. With formats like Iceberg, it's now feasible to load just a subset of the data (e.g., partition, snapshot) into a small, high-performance solution by plugging the data into different query engines depending on the use case — enabling greater flexibility, cost control, and performance tuning.

Table 2. Query engine strengths and use cases

Engine	Strength	Ideal Use Case
DuckDB	Embedded, local, ultra-fast	In-browser analytics, notebooks, embedded dashboards
Apache Trino	Federated, distributed SQL over many sources	Cross-source queries, enterprise data lake queries
ClickHouse	Real-time OLAP, high-concurrency analytics	Product analytics, time series, operational dashboards
Spark/Flink	Batch and streaming compute at scale	Data pipelines, heavy ETL, ML training

Each of these engines can read from Iceberg tables, allowing them to share a consistent and versioned view of the data, while offering different tradeoffs in latency, interactivity, and cost. This means:

- You don't have to move the entire dataset, just the slice that matters
- Query engines can be spun up instantly in containers or local environments
- Data engineers and analysts can perform high-speed analytics without needing massive infrastructure

Bringing Analytics to the Edge: Embedded Engines and In-Browser Compute

This evolution is driven by lightweight, high-performance engines that enable embedded databases at the edge using technologies like WebAssembly, making it possible to run SQL queries directly in the browser or on client devices. Engines like DuckDB bring OLAP-style analytics to the edge, while SQLite remains a widely used solution for OLTP workloads in mobile and embedded environments.

This shift is not just about conceptually bringing data closer to users — it's about running analytics *physically* in their environment. With embeddable engines, we're entering an era where queries are executed locally, unlocking powerful new patterns such as:

- Running analytics in the browser (WASM)
- Offline exploration of datasets
- Instant, local dashboards with zero back-end latency
- Embedding analytics into desktop apps or mobile tools

By minimizing infrastructure and maximizing locality, embedded analytics is redefining what's possible at the edge and bringing responsiveness, resilience, and autonomy to the forefront of modern data experiences.

Evolving Authorization: Data-Aware and Fine-Grained Access Control

As modern data architectures evolve into distributed and federated ecosystems — with multiple engines, domains, and access points — authorization becomes a critical, yet often overlooked, challenge. To truly democratize data, access control must evolve into something more context aware, fine-grained, and decoupled from infrastructure and traditional RBAC models.

While RBAC works in centralized environments, it quickly breaks down in data mesh or multi-engine lakehouse architectures, where access must be managed across domains, services, and tools.

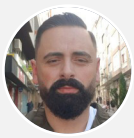
To address this, a new class of tools is emerging, focused on fine-grained, distributed, and technology-agnostic authorization. Solutions like OpenFGA and Permit.io offer:

- Centralized policy management decoupled from application logic
- Attribute-based and relationship-based access control (ABAC/ReBAC)
- APIs to enforce consistent rules across services and platforms
- Native support for hierarchical and contextual rules (e.g., row-level access, data sensitivity levels)

These systems treat authorization as a first-class architectural concern, not just a checkbox. In the future, we may see access control policies become as modular and composable as the data stacks they govern — embedded across services, versioned alongside data definitions, and enforced consistently from the data lake to the browser.

Conclusion

As the boundaries between batch and real time blur, and as AI drives demand for both scale and interactivity, the future lies in architectures that are open, modular, and context aware. A lakehouse backed by Iceberg, powered by multiple engines, and extended all the way to the browser is no longer a future ideal but rather the blueprint for modern data platforms. The opportunity ahead is to embrace openness, modularity, and locality, putting data to work wherever it's needed, from the cloud to the edge.



CORE

Miguel Garcia Lorenzo

@miguelglor

@mgarlorenzo

Miguel is vice president of engineering at Factorial. He has more than 10 years of experience in the data space, leading teams and building high-performance solutions. He is a book lover and advocate of platform design as a service and data as a product.

PARTNER OPINION

The Shift to AI-Native Architecture: What Tech Leaders Must Know

By Kiyu Gabriel, GM, Search & AI at DataStax, an IBM company

As GenAI moves from experimentation to enterprise adoption, technology leaders face a new reality: AI success depends less on model choice and more on architecture, governance, and data infrastructure. Without these, even the best models won't deliver value at scale.

Why Traditional Architectures Fall Short

Legacy systems were built to be static — AI is anything but. GenAI systems need to ingest real-time feedback, adapt to new data, and evolve continuously. This calls for:

- Always-on infrastructure: real-time replication and low-latency for reads and writes
- High-throughput ingestion: AI writes as much as it reads — logs, fine-tuning data, and signals
- AI-optimized knowledge layers: vector search, hybrid retrieval, and graph-based enrichment

Failing to modernize architecture leads to stalled AI progress and poor ROI.

Cost Efficiency Starts With the Right Stack

Powerful models are expensive. But the smarter play is in model specialization and reuse:

- Choose models based on domain (e.g., conversational vs. retrieval)
- Run sustained workloads on-prem with GPUs
- Use composed model stacks — embedding, reasoning, and reranking — to minimize waste

Governance Must Be AI-Aware

AI introduces new governance complexity: bias, lineage, compliance, and behavior risk. Tech leaders need solutions that go beyond traditional security and data governance:

- WatsonX governance for runtime guardrails and auditability
- IBM Manta for full data lineage and explainability

Governance isn't just risk mitigation — it's how you scale safely.

The Rise of the AI-Powered Builder

AI-assisted development platforms are changing who builds software. With tools like Langflow, business users can launch AI workflows using visual builders — no code needed. Tech leaders must rethink team structures to empower faster delivery across more roles.

Key Takeaway

GenAI is a new paradigm changing system design, service interaction, data governance, and team staffing. Leaders who embrace this organizational transformation will achieve faster delivery, smarter systems, and a greater competitive advantage. To succeed in AI, enterprises must:

- Build adaptive, AI-ready data architectures
- Manage costs through model specialization
- Bake governance and lineage into pipelines
- Empower nontraditional builders with AI tooling

DataStax, an IBM company, powers generative AI applications with real-time, scalable data, production-ready vector data tools that generative AI applications need, and seamless integration with developers' stacks of choice.

DATASTAX
an IBM Company

The Fastest Way to Build & Deploy AI Apps

Unleash the full potential of generative AI with Astra DB, the cloud-native vector and NoSQL database designed for building real-time AI applications. Combine it with Langflow, the drag-and-drop visual interface for creating and managing LLM workflows - no complex coding required.

Together, they give developers and data teams the tools to go from prototype to production - fast.

Built for Real-Time AI
Powered by Apache Cassandra®, Astra DB is optimized for low-latency data retrieval and processing.

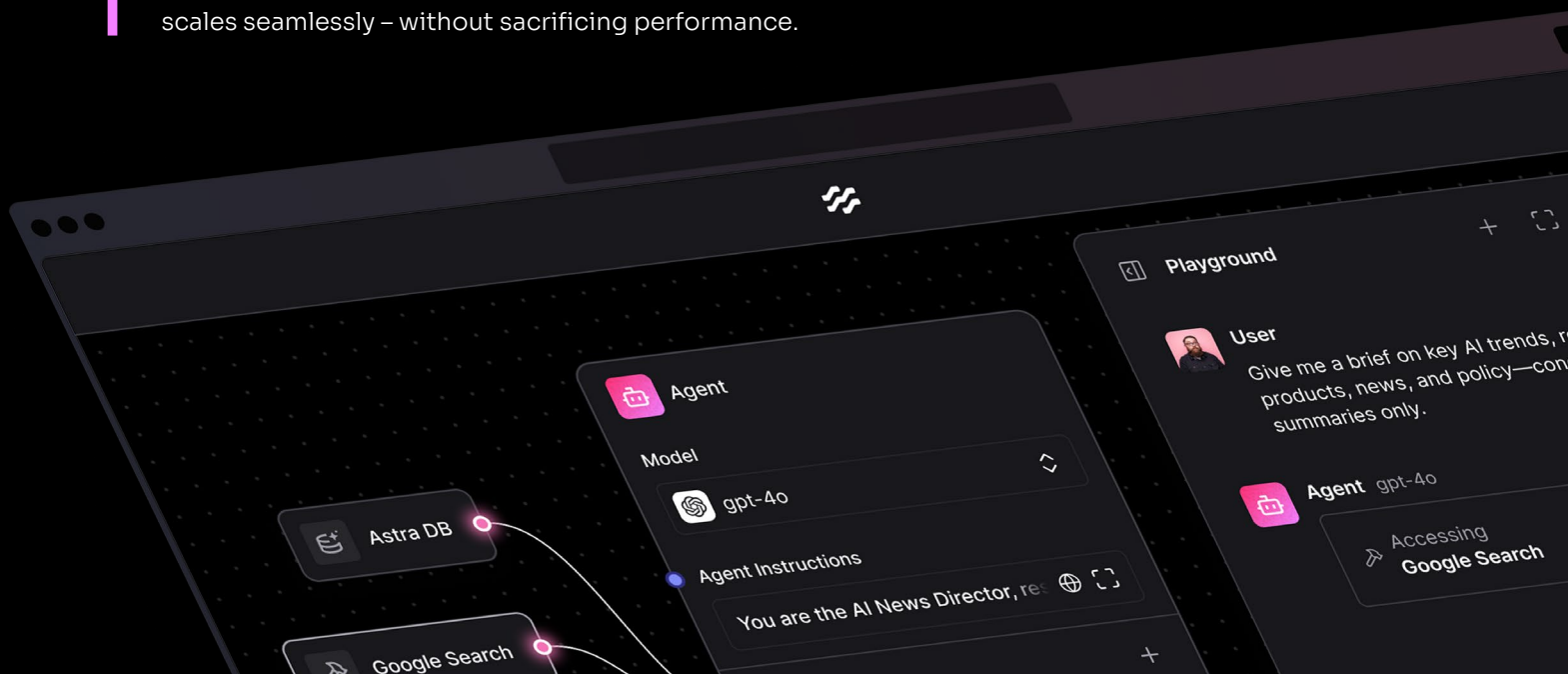
Accelerate Time-to-Value
Cut development time by up to 90% and deliver 95%+ RAG accuracy with built-in AI-optimized infrastructure.

Enterprise-Ready & Secure
Gain full control, robust security, and compliance built for enterprise needs.

Cloud-Native & Cost-Efficient
Lower your TCO with fully managed infrastructure that scales seamlessly – without sacrificing performance.

Try for Free Today
dtsx.io/3INAjSd

Talk to a Data Architect
dtsx.io/3IrRvgh



CONTRIBUTOR INSIGHTS

Data Engineering for AI-Native Architectures

Designing Scalable, Cost-Optimized Data Pipelines to Power GenAI, Agentic AI, and Real-Time Insights

By Abhishek Gupta, Product Manager at Microsoft

The data engineering landscape has undergone a fundamental transformation with a complete reimagining of how data flows through organizations. Traditional business intelligence (BI) pipelines were built for looking backward, answering questions like "How did we perform last quarter?" Today's AI-native architectures demand systems that can feed real-time insights to recommendation engines, provide context to large language models, and maintain the massive vector stores that power retrieval-augmented generation (RAG).

For data engineers, this evolution has created both opportunity and complexity. The core principles of data quality, observability, and scalable architecture remain as important as ever, but the technical requirements have expanded dramatically. What makes this particularly challenging is that most organizations can't simply rip out their existing data infrastructure and start fresh.

A successful approach involves building AI-native capabilities on top of traditional analytics systems. Organizations that successfully build scalable, cost-optimized data platforms capable of supporting both traditional analytics and cutting-edge AI applications will find themselves with a significant competitive advantage. Those that don't risk being left behind as AI capabilities become table stakes across every industry vertical.

This article explores how data engineering teams can navigate this transformation, from redesigning core architecture patterns to implementing the new toolsets and practices required for AI-native data platforms.

The Transformation of Data Architecture for AI

Traditional data architectures were built around the assumption that data would be cleaned, structured, and processed in predictable batches. Your nightly ETL job could take six hours because the BI dashboard only refreshes once a day. AI systems, particularly modern generative and agentic AI applications, operate under completely different assumptions.

If you're trying to feed a transformer model with terabytes of training data or build a RAG system that needs to retrieve semantically similar documents in under 100ms, your traditional ETL pipeline and star schema aren't going to cut it. The companies getting AI right aren't just bolting machine learning onto their existing data warehouse. Rather, *they're building AI-native architectures from the ground up*.

Table 1. Traditional vs. AI-native data architectures

Aspect	Traditional Data Architecture	AI-Native Data Architecture
Data types	Structured data only (CSV, JSON, SQL tables)	Multimodal data (text, images, audio, video, sensor data)
Processing model	Batch processing (ETL pipelines)	Real-time streaming + batch hybrid
Latency requirements	Hours to days acceptable	Milliseconds to seconds required
Processing schedule	Scheduled jobs (nightly, weekly)	Event driven, continuous processing
Data storage	Data warehouses with star schema	Data lakes + vector databases + feature stores
Storage format	Relational tables, normalized data	Object storage (Parquet, Delta Lake, Iceberg)
Query patterns	SQL-based analytics queries	Vector similarity search + traditional queries
Infrastructure	Cloud native, serverless, GPU-accelerated	Cloud native, serverless, GPU accelerated
Consumption	BI dashboards, reports	AI applications, recommendation engines, chatbots

From Batch to Streaming: Real-Time Data for AI Systems

Modern AI applications, especially those involving recommendation systems, fraud detection, and conversational AI, require continuous learning from fresh data streams. When a user interacts with a ChatGPT-style application, the system needs to understand context, not just from the current conversation but also from real-time signals about user behavior, system performance, and external data sources. This creates a fundamentally different set of requirements than traditional analytics workloads.

Streaming platforms like Apache Kafka and Apache Pulsar provide distributed event streaming capabilities for real-time data processing:

- **Kafka** uses a log-based architecture with topics partitioned across brokers, supporting configurable consistency levels from at-least-once to exactly-once delivery semantics.
- **Pulsar** implements a multi-layered design, separating serving from storage through Apache BookKeeper.
  - **BookKeeper** enables infinite retention without performance degradation and native schema registry support for enforcing data contracts across different services and versions.

These are key parts of streaming architectures for generative AI that implement AI-specific patterns. Real-time streams capture user interactions, model outputs, and feedback signals, feeding them immediately to online learning systems and safety monitoring. Meanwhile, batch processes handle compute-intensive operations like embedding generation for RAG systems and large-scale model fine-tuning on accumulated conversation data.

Managing Unstructured Data Workflows

Unstructured data ranging from text and images to audio and video is a first-class citizen in the world of data engineering. Unlike structured data, which fits neatly into rows and columns, unstructured data is inherently messy and diverse. Yet it is precisely this richness that makes it indispensable for training large language models, building multimodal AI systems, and powering next-gen user experiences. Organizations are increasingly investing in data pipelines that can ingest, store, transform, and serve unstructured data reliably.

Modern orchestration tools like Apache Airflow, Dagster, and Prefect are evolving to support these needs, often in combination with cloud-native object storage solutions such as Azure Blob Storage. Processing unstructured data at scale typically involves distributed compute frameworks like Apache Spark or Dask, containerized workloads running on Kubernetes, or GPU-accelerated tasks for specialized transformations. Pipelines are increasingly modular, chaining together tasks such as format normalization, metadata extraction, vectorization, and enrichment. These stages must be resilient, parallelizable, and auditable, thereby capable of handling large volumes while preserving the fidelity and lineage of the original content.

Also bear in mind that the challenge isn't just processing unstructured data. It's also about maintaining quality and provenance throughout the pipeline. When the AI model starts producing biased or incorrect outputs, you need to be able to trace those problems back to specific data sources, processing steps, and transformation logic. This requires a level of observability and metadata management that goes far beyond traditional data engineering practices.

Building Scalable AI Data Infrastructure

The shift toward AI-native architectures demands infrastructure that can simultaneously serve a business analyst running quarterly reports and a research team fine-tuning a foundation model on terabytes of unstructured data. What's particularly challenging is that AI workloads are inherently unpredictable. A traditional ETL pipeline processes the same data volumes at scheduled intervals, but AI training runs can consume massive compute resources for days before completion, while inference workloads might spike suddenly when a new product feature launches.

This variability requires infrastructure that can scale both horizontally and vertically without manual intervention, often leveraging cloud-native services that can provision resources on demand.

High-Volume Dataset Strategies for AI Training

Training foundation models and agentic AI systems requires infrastructure that can handle massive, heterogeneous datasets. Object storage systems are favored for their scalability and cost efficiency, supporting large-scale ingestion through batch or streaming pipelines. Columnar formats such as Apache Parquet, combined with partitioning, reduce I/O load and improve access speed for downstream processing.

Data lakes serve as the standard architecture for managing heterogeneous AI datasets, accommodating raw, unstructured data while maintaining essential metadata for governance. Apache Iceberg and Delta Lake provide atomicity, consistency, isolation, and durability (ACID) transactions and time-travel capabilities, enabling dataset versioning and rollback functionality when model performance degrades. Integration with distributed frameworks like Spark and Dask creates sophisticated ETL pipelines that transform raw data into training-ready formats while maintaining provenance tracking for enterprise compliance requirements.

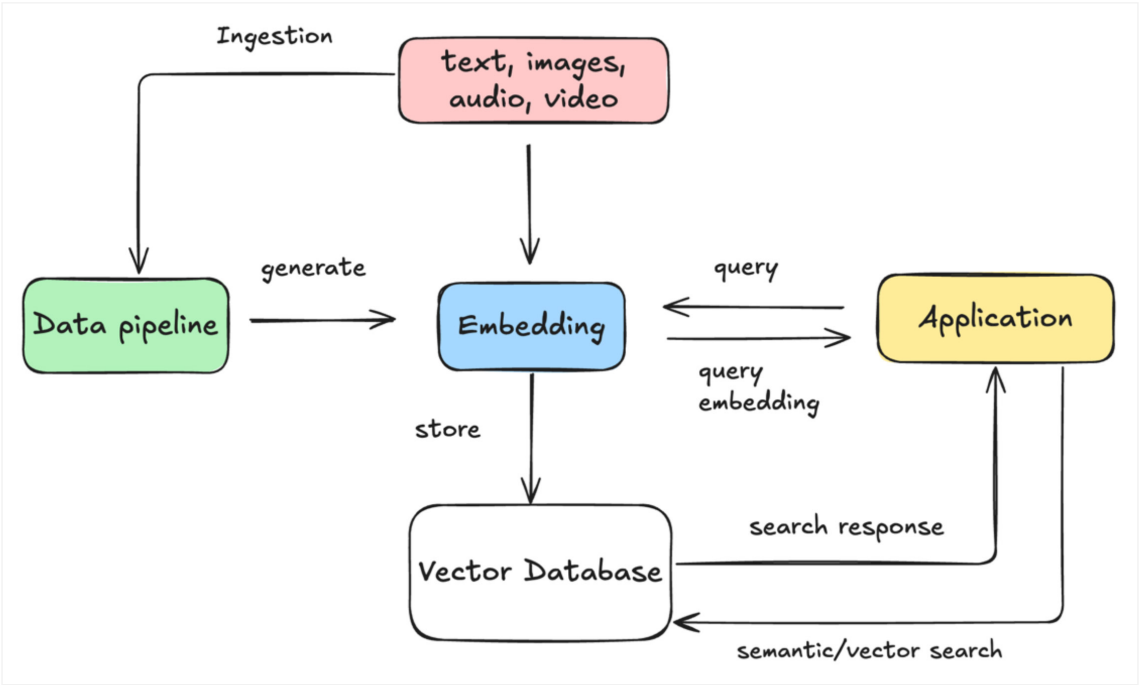
Distributed computing frameworks form the computational backbone for large-scale AI operations, with Ray and Spark dominating based on workload characteristics. Ray excels in fine-grained parallelism and stateful computations for reinforcement learning and agentic systems, while Spark remains preferred for batch processing and supervised learning pipelines. GPU-aware schedulers and CUDA-enabled processing libraries accelerate throughput but require careful memory management and data locality optimization to minimize movement between storage and compute layers.

Vector Databases and Embedding Management

Vector embeddings are numerical representations of data (e.g., text, images, audio) that capture semantic meaning in high-dimensional space, enabling generative AI apps to understand context, perform similarity searches, and retrieve relevant information for tasks like RAG and semantic search.

Vector databases like Pinecone, Weaviate, and Chroma solve a fundamental problem: how to efficiently store, index, and query high-dimensional embeddings that represent the semantic meaning of text, images, audio, or any other data types. These databases use specialized indexing algorithms like hierarchical navigable small world (HNSW) or inverted file (IVF) that enable approximate nearest neighbor searches across millions of vectors with low latency.

Figure 1. RAG with vector databases



Managing embeddings across diverse models and use cases presents significant complexity. A single organization might use OpenAI's `text-embedding-ada-002` for document search, custom-trained image models for visual similarity, and specialized models for code search, each producing vectors with different dimensionalities and semantic characteristics. This needs careful namespace management, version control, and compatibility planning.

During model upgrades or migrations, new or fine-tuned models generate embeddings in different semantic spaces, making existing vectors non-interchangeable. To address this, teams typically version their embedding collections and implement migration strategies such as dual-serving or fallback mechanisms, allowing multiple versions to coexist during phased transitions.

Operationally, embedding management extends beyond storage into the design of efficient generation pipelines. For large datasets, especially with multimodal inputs like images or video, embedding computation is resource intensive. Successful systems often combine batch generation for historical data with real-time streaming for new content, enabling both large-scale reprocessing during model changes and incremental updates as data evolves.

Implementation and Optimization Strategies

The gap between AI ambitions and production reality often comes down to implementation choices that seem minor but compound into major operational headaches. You can architect the most elegant AI-native data platform on paper, but if you can't deploy it reliably, scale it cost effectively, or maintain it without a team of specialists, you've built an expensive science project.

Smart organizations are learning that the key isn't choosing between open-source and commercial solutions. It's understanding where each fits in your specific context. The companies succeeding with AI at scale aren't necessarily the ones with the biggest budgets; they're the ones that have figured out how to balance flexibility, cost, and operational complexity in ways that actually work for their teams.

The most effective AI data platforms share a common trait: They're designed for evolution, not perfection. They anticipate that your AI workloads will change, your data volumes will grow unpredictably, and your team will need to adapt to technologies that don't exist yet. Building for this reality requires a fundamentally different approach to implementation and optimization than traditional data infrastructure.

DataOps for AI: Automation and Orchestration

When your data pipeline breaks at 2 AM because someone changed a schema upstream, you quickly realize that manual processes don't scale with AI workloads. DataOps for AI isn't just about applying DevOps principles to data; it's about recognizing that AI systems have fundamentally different reliability requirements than traditional analytics.

Unlike traditional ETL jobs that follow predictable patterns, AI pipelines often involve retraining models based on data quality thresholds, triggering inference updates when embedding models change, or coordinating between multiple agentic systems that need to share state.

The most successful implementations treat AI pipeline orchestration as a first-class engineering discipline. This means version controlling not just your code but also your pipeline definitions, model artifacts, and even the configurations that determine when models get retrained. Companies like Netflix and Uber have shown that you can achieve remarkable reliability by building orchestration systems that understand the relationships between data, models, and business outcomes.

Testing becomes particularly crucial when your pipelines feed AI systems that make autonomous decisions. In addition to unit testing for individual transformations, you need:

- Integration tests that verify end-to-end behavior
- Schema evolution tests that catch breaking changes before they propagate
- Monitoring that can detect subtle data drift that might degrade model performance over time

Your team should be able to deploy changes confidently, understand why failures happen, and recover quickly when things go wrong. This requires building orchestration systems that provide clear visibility into dependencies, explicit control over rollback scenarios, and detailed logging that helps you debug complex interactions between multiple AI systems.

Cross-Functional Collaboration and Governance

The biggest technical challenge in AI data engineering often isn't the technology; it's the organizational complexity that emerges when data engineers, data scientists, ML engineers, and business stakeholders all need different things from the same underlying infrastructure:

- Data scientists want flexibility to experiment with new models and datasets.
- Business stakeholders want reliable and auditable systems that meet compliance requirements.
- Data engineers want maintainable architectures that don't require constant firefighting.

Effective governance starts with establishing clear ownership boundaries through technical mechanisms rather than relying solely on process documentation. This includes implementing automated data quality checks, feature stores that enforce consistent transformations, and metadata systems that track lineage from raw data through model training to business outcomes.

Model governance becomes particularly complex in agentic AI systems where multiple models interact autonomously, requiring frameworks for understanding how changes in one model might affect dependent systems and monitoring that can detect emergent behaviors. Privacy and compliance add another layer of complexity as GDPR right-to-deletion requests become nightmarish when you need to remove specific data points from trained models, embedding stores, and vector databases.

The goal is creating collaboration patterns that scale with your AI initiatives by treating governance as an engineering problem that can be solved with the right combination of tooling, automation, and organizational design, enabling teams to move quickly without creating systemic risks.

Conclusion

We're witnessing the emergence of an architectural paradigm where the distinction between "analytics data" and "AI data" is dissolving. Systems like recommendation engines, fraud detection, and chatbots increasingly rely on shared data infrastructure but with vastly different performance and consistency requirements. The path forward is to *design data platforms with AI as the primary consumer*, retrofitting for traditional workloads rather than the other way around. This shift enables support for evolving needs like large context windows, vector-based retrieval, and complex agentic workflows.

The shift also introduces specific technical demands:

- Cost management must include storage format efficiency, cross-region data movement, and duplication across batch and real-time paths.
- Performance monitoring needs to capture embedding drift, vector index quality, and inference latency under load.
- Orchestration has to coordinate interdependent steps across ingestion, transformation, training, and serving.

These aren't speculative problems; they already surface in production environments that support retrieval-augmented generation, fine-tuned models, and agent-based systems. The organizations that get this right will move beyond efficiency gains to build data architectures that unlock durable AI capabilities.

Additional resources:

- *Getting Started With Large Language Models* by Tuhin Chattopadhyay, DZone Refcard
- *Getting Started With Agentic AI* by Lahiru Fernando, DZone Refcard
- *Getting Started With Vector Databases* by Miguel Garcia Lorenzo, DZone Refcard
- *Getting Started With Data Quality* by Miguel Garcia Lorenzo, DZone Refcard
- *Apache Kafka Patterns and Anti-Patterns* by Abhishek Gupta, DZone Refcard

CORE

Abhishek Gupta

@abhirockzz

@abhirockzz

Abhishek is a Product Manager at Microsoft. Over the course of his career, he has worn multiple hats including engineering, consulting, and product management. Most of his work has revolved around open-source technologies including distributed data systems and cloud-native app dev platforms. He is also an open-source contributor, speaker, and author.

CONTRIBUTOR INSIGHTS

Scaling Real-Time Data Systems With DataOps: Principles, Practices, and Use Cases

By Tulika Bhatt, Senior Software Engineer at Netflix

Real-time decision making is no longer a competitive advantage; it's becoming a baseline expectation. From fraud detection to personalized recommendations, modern systems are expected to process and respond to user activity within milliseconds. But while demand for real-time data has exploded, many engineering teams are still struggling with brittle pipelines, silent failures, and fragile deployments.

In this article, we explore how DataOps brings much-needed discipline to real-time architectures. We'll dive into principles like CI/CD, schema versioning, observability, and environment parity, and walk through a fully open-source clickstream pipeline that puts these ideas into action.

DataOps Principles for Streaming Architectures

One of the foundational principles of DataOps is treating **data as a product**. This means going beyond the one-off script or ad hoc transformation logic to view data pipelines as long-lived assets. A real-time data pipeline, much like an API or service, should have version control, proper documentation, automated tests, and clearly defined contracts with its consumers. This enables pipelines to evolve safely over time, much like APIs.

Another fundamental pillar of DataOps is enabling the **continuous delivery of both data and metadata**. While batch workflows often align deployment cycles with scheduled reports or model refreshes, real-time systems demand a different approach. Because they operate in an ongoing, always-on mode, the supporting delivery mechanisms must also be continuous, automated, and resilient.

In this setting, CI/CD pipelines take on expanded responsibilities. They not only deploy new code but also track schema changes, propagate metadata like lineage and freshness, and ensure consistent configuration across development, staging, and production environments. Whether it's a transformation update, a configuration change, or a schema evolution, every modification must be treated with the same engineering discipline and testing rigor as a production software release.

Reproducibility and **environment parity** are equally vital in streaming architectures. The cost of "it works on my machine" can be disastrous when data is flowing in real time. Consistent configuration across environments helps prevent subtle bugs that emerge only in production. This requires a strong discipline of managing infrastructure and pipeline definitions as code.

Together, these principles form the backbone of DataOps for real-time systems. But to put them into practice, we need supporting strategies around schema management, orchestration, and monitoring. In the next few sections, we'll explore each of these dimensions in detail and learn about ways we could incorporate them in our systems.

Managing Schemas and Sources in Streaming Architectures

One of the most important aspects of building real-time pipelines is schema management. Before you can monitor or deploy pipelines safely, you need to manage the structure of your data. Schemas should be version controlled and validated before being promoted to production. For instance, using Apache Avro or Protobuf, you can define a record structure that producers and consumers agree on. These schemas are stored in a registry and validated during pull requests, ensuring backward and forward compatibility.

```
json

//Avro schema for a sample 'ClickEvent'
```

CODE CONTINUES ON NEXT PAGE

```
{
  "type": "record",
  "name": "ClickEvent",
  "namespace": "com.example.analytics",
  "fields": [
    { "name": "user_id", "type": "int" },
    { "name": "event_type", "type": "string" },
    { "name": "timestamp", "type": "long" }
  ]
}
```

In addition to managing schemas, teams should apply the same rigor to the logic and configuration that power their pipelines. Streaming SQL, transformation logic, and configuration files should all live in Git alongside test harnesses and example inputs. A modular repo structure where business logic, test data, and environment configs are separated makes it easier to review changes and trace errors when incidents arise. This way, teams can build pipelines that are testable and reproducible by treating both schemas and code as source-controlled assets.

```
// A modular project structure for storing configs, code and tests

clickstream-pipeline/
├── README.md
├── ci/
│   └── validate-schema.sh             # CI scripts for schema compatibility, linting
├── config/
│   ├── dev/
│   │   └── flink-config.yaml          # Dev-specific job config
│   ├── staging/
│   │   └── flink-config.yaml
│   └── prod/
│       └── flink-config.yaml          # Production-specific parameters (e.g., Kafka topics)
├── jobs/
│   └── clickstream-sessionizer/
│       ├── src/
│       │   └── MainFlinkJob.java      # Main Flink job logic
│       ├── resources/
│       │   └── application.conf       # Job-level config
│       └── build.gradle               # Build file for the job
├── schemas/
│   ├── v1/
│   │   └── click_event.avsc           # Initial schema version
│   └── v2/
│       └── click_event.avsc           # Updated schema version
├── test/
│   ├── integration/
│   │   ├── sample-click-events.json   # Test data for end-to-end validation
│   │   └── expected-output.json
│   └── unit/
│       └── ClickstreamJobTest.java    # Unit tests for transformation logic
├── docker/
│   └── Dockerfile                     # Container definition for deployment
└── .github/
    └── workflows/
        └── ci.yaml                    # GitHub Actions workflow for CI/CD
```

Automating Deployments and Validating Pipeline Changes

Schema management lays the foundation, but safe deployments require robust orchestration and automation. Streaming pipelines should be deployed declaratively, using versioned job definitions and Infrastructure-as-Code principles. For example, rather than manually submitting jobs or modifying runtime parameters, teams can define their Flink or Airflow jobs as code, store them in Git, and apply changes via pull requests.

```
name: CI/CD for Clickstream Pipeline

on: [push]

jobs:
  build-test-deploy:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout
      uses: actions/checkout@v3

    - name: Build JAR
      run: ./gradlew shadowJar

    - name: Validate Schema
      run: ./ci/validate-schema.sh

    - name: Run Integration Tests
      run: ./test/run-integration-tests.sh

    - name: Deploy to Staging
      run: ./scripts/deploy.sh staging

    - name: Canary Deploy to Prod
      run: ./scripts/canary-deploy.sh
```

This kind of automation removes the human error from critical deployments, encourages frequent iteration, and creates a paper trail for audits or incident response. It also supports true environment parity, where staging and production differ only by templated variables and not code.

Observability and On-Call Practices for Streaming Pipelines

Even with robust deployments, real-time systems will eventually fail. Metrics and observability are what turn these failures into solvable problems rather than open-ended mysteries. Every layer of a streaming pipeline emits important operational signals that help engineers monitor health, performance, and failures. The table below highlights key metrics from the most common components:

Table 1. Key metrics for most common components

Pipeline Layer	Key Operational Signals
Apache Kafka	Consumer lag, message throughput, partition skew
Apache Flink	Checkpoint duration, processing latency, error rates
Apache Airflow	DAG success/failure rates, task duration, scheduling delays

These metrics should be exposed through tools like Prometheus and visualized with dashboards (e.g., Grafana). But the real value comes from creating business-aware alerts. Instead of paging on CPU spikes, alert when event throughput drops unexpectedly or when schema violations spike.

Validation checks can also be embedded directly in job logic. A Flink job, for example, might verify that a `user_id` field is never `null` or that a specific distribution of event types remains within an expected range. These checks help surface data quality issues before they reach consumers. On-call readiness also means having clear runbooks.

When an alert fires, responders should know: what changed, who owns it, and what the rollback plan is. Post-incident reviews help refine these processes over time, improving resilience across the board.

Use Case: Real-Time Clickstream Aggregation

To see DataOps in action, let's walk through a simplified clickstream pipeline. We will use real data, simulate real-time ingestion, and process events using streaming tools, all while applying versioning, testing, and observability best practices.

Setting Up the Infrastructure Stack

Before actually writing code, we need to set up the infrastructure stack for our DataOps-enabled project. This involves installing Kafka and Apache Zookeeper for data ingestion, Flink for stream processing, Apicurio Schema Registry for schema management, and Prometheus and Grafana for observability. We can use this sample file for installation:

```
docker-compose.yml
# Docker Compose setup for Kafka, Flink, Apicurio, Prometheus, Grafana

version: '3.8'

services:
  zookeeper:
    image: confluentinc/cp-zookeeper:7.0.1
    environment:
      ZOOKEEPER_CLIENT_PORT: 2181
      ZOOKEEPER_TICK_TIME: 2000
    ports:
      - "2181:2181"

  kafka:
    image: confluentinc/cp-kafka:7.0.1
    depends_on:
      - zookeeper
    ports:
      - "9092:9092"
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://localhost:9092
      KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

  apicurio:
    image: apicurio/apicurio-registry-mem:2.3.2.Final
    ports:
      - "8081:8080"
    environment:
      QUARKUS_PROFILE: prod
      REGISTRY_DATASOURCE_URL: jdbc:h2:mem:registry
      REGISTRY_UI_CONFIG_URL: http://localhost:8081

  jobmanager:
    image: flink:1.17.1
    ports:
      - "8082:8081"
    command: jobmanager
```

CODE CONTINUES ON NEXT PAGE

```
    environment:
      - JOB_MANAGER_RPC_ADDRESS=jobmanager

  taskmanager:
    image: flink:1.17.1
    depends_on:
      - jobmanager
    command: taskmanager
    environment:
      - JOB_MANAGER_RPC_ADDRESS=jobmanager

  prometheus:
    image: prom/prometheus
    volumes:
      - ./prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - "9090:9090"

  grafana:
    image: grafana/grafana
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_PASSWORD=admin
```

Ingesting Events

The next phase focuses on data ingestion. We start by exploring the dataset and defining a schema that represents the event structure accurately. This schema is registered to enforce compatibility and enable downstream validation. Real-time events are then produced into Apache Kafka, and Apache Flink is used to ingest and process the event stream as it flows through the system.

REGISTERING SCHEMA

Once we have performed the above, the next step is to explore the dataset, finalize the schema, and register it. We will be using the E-Commerce Behavior Data From Multi-Category Store for our walkthrough, which includes user events such as page views and purchases with timestamps. Registering schema in a registry is essential as it validates the incoming events and enforces backward compatibility across versions. This ensures producers don't accidentally introduce breaking changes that affect downstream jobs. You can use the CLI or cURL to register your Avro schema:

```
cat <<EOF > click_event.avsc
{
  "type": "record",
  "name": "ClickEvent",
  "fields": [
    {"name": "user_id", "type": "long"},
    {"name": "event_type", "type": "string"},
    {"name": "timestamp", "type": "long"}
  ]
}
EOF

curl -X POST http://localhost:8081/apis/registry/v2/groups/default/artifacts   -H "Content-Type:
application/json"   -H "X-Registry-ArtifactId: click-event"   --data-binary @schemas/click_event.
avsc
```

PRODUCE TO KAFKA (SIMULATED REAL TIME)

Next, we will simulate real-time events by reading records of the e-commerce behavior dataset, pausing for 50ms between each record and sending it to Kafka:

```
python
import time
import json
import pandas as pd
from fastavro import schemaless_writer
from io import BytesIO
from confluent_kafka import Producer

# Load schema
with open("schemas/click_event.avsc", "r") as f:
    schema = json.load(f)

# Set up Kafka producer
producer = Producer({'bootstrap.servers': 'localhost:9092'})

# Load the CSV (you can limit for testing)
df = pd.read_csv("test/data/2019-Nov.csv", nrows=10000)

# Filter and rename
df = df[["user_id", "event_type", "event_time"]]
df = df.dropna()

# Convert datetime to epoch ms
df["event_time"] = pd.to_datetime(df["event_time"], utc=True)
df["event_time"] = df["event_time"].astype(int) // 10**6  # to milliseconds

# Publish to Kafka
for _, row in df.iterrows():
    event = {
        "user_id": int(row['user_id']),
        "event_type": str(row['event_type']),
        "timestamp": int(row['event_time'])
    }

    buffer = BytesIO()
    schemaless_writer(buffer, schema, event)

    producer.produce("clickstream", value=buffer.getvalue())
    print(f"Sent: {event}")
    time.sleep(0.05)  # simulate real-time

producer.flush()
```

STREAM PROCESSING IN REAL TIME

We'll use a Flink job to handle these events as they arrive, grouping them by user session, tracking user actions through conversion funnels, and counting event types within tumbling time windows, by using either SQL or the DataStream API.

SEE CODE ON NEXT PAGE

```
-- Define source table using Avro and schema registry
CREATE TABLE clickstream (
  user_id BIGINT,
  event_type STRING,
  timestamp TIMESTAMP(3),
  WATERMARK FOR timestamp AS timestamp - INTERVAL '5' SECOND
) WITH (
  'connector' = 'kafka',
  'topic' = 'clickstream',
  'properties.bootstrap.servers' = 'localhost:9092',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'avro-confluent',
  'avro-confluent.schema-registry.url' = 'http://apicurio:8080/apis/registry/v2',
  'avro-confluent.subject' = 'click-event'
);

-- Define sink table (printing for now)
CREATE TABLE event_summary (
  event_type STRING,
  event_count BIGINT,
  window_start TIMESTAMP(3),
  window_end TIMESTAMP(3)
) WITH (
  'connector' = 'print'
);

-- Aggregate click events by type and time window
INSERT INTO event_summary
SELECT
  event_type,
  COUNT(*) AS event_count,
  TUMBLE_START(timestamp, INTERVAL '10' MINUTE),
  TUMBLE_END(timestamp, INTERVAL '10' MINUTE)
FROM clickstream
GROUP BY
  TUMBLE(timestamp, INTERVAL '10' MINUTE),
  event_type;
```

CI/CD Validation

The job's logic, configuration, and test cases are version controlled in Git. When a change is proposed, continuous integration runs schema validations and unit tests before deploying to a staging environment. After successful verification, a canary deployment gradually rolls out the update to production.

```
name: CI for Clickstream Pipeline
on: [push]
jobs:
  build-test-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3
```

CODE CONTINUES ON NEXT PAGE

```
      - name: Build pipeline logic
        run: ./gradlew shadowJar

      - name: Validate Avro Schema
        run: ./ci/validate-schema.sh

      - name: Run integration tests
        run: ./test/integration/run-tests.sh

      - name: Deploy to staging
        run: ./deploy/deploy.sh staging

      - name: Canary deploy to prod
        run: ./deploy/deploy.sh production --canary
```

The following table lists different configs for different environments.

Table 2. Configs for various environments

Config Category	What to Customize	Examples
Kafka topics	Use separate topics per env to isolate data	clickstream-dev, clickstream-staging, clickstream-prod
Schema registry	Point to isolated schema registry or use group namespace	http://localhost:8081/groups/dev
Checkpoint storage	Use different file system paths	/tmp/flink/dev-checkpoints/, /mnt/prod-checkpoints/
State backends	Enable RocksDB in prod, memory in dev	rocksdb vs heap
Parallelism	Lower parallelism for dev/test	parallelism.default: 1 (dev), 8 (prod)
Job timeouts	Use shorter timeouts for rapid iteration in dev	execution.checkpointing.interval: 2m (dev) vs 10m (prod)
Logging levels	Verbose logging in dev, warnings only in prod	INFO (dev), WARN or ERROR (prod)
Dead letter queues	Use isolated DLQ Kafka topics	dlq-dev, dlq-prod
Validation flags	Enable strict checks in staging/prod	`validate.schema: true`
Alert thresholds	More tolerant in dev	Alert on lag > 10k (dev) vs > 1k (prod)

Observing the Pipeline

Flink exports metrics like processing lag and checkpoint duration to Prometheus. Dashboards in Grafana visualize these metrics alongside custom alerts for event drops or schema anomalies. For example, we can update the config file with following metrics and restart the flink job with these configs to export metrics to Prometheus:

```
flink-conf.yaml

metrics.reporter.prom.class: org.apache.flink.metrics.prometheus.PrometheusReporter
metrics.reporter.prom.port: 9250
```

Prometheus will scrape metrics at: http://flink-jobmanager:9250/metrics. To set up Prometheus, we need to edit the file as shown below:

```
prometheus.yml

global:
  scrape_interval: 15s
```

CODE CONTINUES ON NEXT PAGE

```
scrape_configs:
  - job_name: 'flink'
    static_configs:
      - targets: ['jobmanager:9250']

  - job_name: 'kafka'
    static_configs:
      - targets: ['kafka:9101']
```

Similarly, we can visualize these metrics in Grafana by these steps:

1. Login: http://localhost:3000 (default user: admin/admin)
2. Add Prometheus as a data source
3. Create dashboards:
    - Kafka consumer lag
    - Flink job latency and checkpoint time
    - Data freshness or throughput by event type

To setup alerting, use these configs in the Grafana rules:

```
alert:
  - alert: HighLag
    expr: kafka_consumer_lag > 5000
    for: 2m
    labels:
      severity: critical
    annotations:
      summary: "Kafka consumer lag too high"
```

Conclusion

As the industry accelerates toward real-time decisioning, DataOps is no longer a nice-to-have; it's becoming essential infrastructure. The fragility of streaming systems makes traditional batch-era practices obsolete, and teams need new ways to scale trust, resilience, and speed. By applying software engineering principles to the data layer, organizations can tame the chaos of streaming pipelines. This includes versioning schemas, automating deployments, validating logic before release, and building observability into every stage of the data pipeline.

The journey doesn't require a full overhaul. Teams can start small, with schema validation or simple CI pipelines, and evolve toward full automation and on-call readiness. Those who do will reduce downtime and unlock faster iteration, better data quality, and a system that's built to last.

Tulika Bhatt

@tb2134

Tulika Bhatt is a senior software engineer at Netflix, specializing in real-time data systems and personalization infrastructure. She writes and speaks on DataOps, stream processing, and scalable data platforms, and is passionate about making complex systems observable, reliable, and easy to evolve.

CONTRIBUTOR INSIGHTS

How Healthy Is Your Data in the Age of AI?

An In-Depth Checklist to Assess Data Accuracy, Governance, and AI Readiness

By Sukanya Konatam, Sr. Manager, Data Governance & Data Science at Vialto Partners

Data has evolved from a byproduct of business processes to a vital asset for innovation and strategic decision making, and even more so as AI's capabilities continue to advance and are integrated further into the fabric of software development. The effectiveness of AI relies heavily on high-quality, reliable data; without it, even the most advanced AI tools can fail. Therefore, organizations must ask: *How healthy is our data?*

Whether initiating a new AI project or refining existing data pipelines, this checklist provides a structured framework that will not only guarantee the success of your AI initiatives but also cultivate a culture of data responsibility and long-term digital resiliency.

Ensuring Data Quality Across Architectures, Models, and Monitoring Systems

Data quality is the backbone of an AI system's integrity and performance. As AI applications become ubiquitous across diverse industries, the reliability of the data that our AI model learns from and runs on is crucial. Even the most advanced algorithms may fail to deliver appropriate and unbiased results when fed with low-quality data, consequences that can be costly in many ways. Moreover, biased data may extend or strengthen existing societal and economic disparities and, consequently, make unjustified decisions.

1. Assess the Core Dimensions of Data Quality

Evaluating the health of your data should cover the core dimensions of data quality: accuracy, completeness, consistency, timeliness, and validity. These dimensions play a critical role in realizing a robust, ethical, and trustworthy AI solution that will be reliable and succeed in meeting its potential:

ACCURACY

- ☐ Confirm that data values are correct and error free
- ☐ Enforce validation checks (e.g., dropdowns, input masks) at data entry
- ☐ Automatically and regularly cross-check data against trusted sources and known standards (e.g., via address validation APIs)
- ☐ Implement mechanisms to tag abnormalities in real time

COMPLETENESS

- ☐ Ensure all required fields in forms and ingestion pipelines are populated
- ☐ Trace missing values to specific sources or systems
- ☐ Identify recurring gaps in critical data using profiling tools
- ☐ Track completeness over time to determine data gaps or failed integration

CONSISTENCY

- ☐ Implement single naming standards, codified code lists, and standard data types on ETL processes
- ☐ Create and maintain a data dictionary that each team uses when field mapping
- ☐ Reconcile redundant datasets regularly to identify and eliminate discrepancies

UNIQUENESS

- ☐ Detect duplicate records (e.g., customer profiles)
- ☐ Ensure that primary keys are unique and enforced strictly

CHECKLIST CONTINUES ON NEXT PAGE

TIMELINESS

- ☐ Identify requirements of your use case (e.g., monthly reports with a batch load)
- ☐ Ensure that data is up to date and available when needed
- ☐ Monitor latency between data generation and delivery, and send a warning if SLAs are at risk
- ☐ Align ingestion frequencies (hourly, daily, real time) with stakeholder requirements

VALIDITY

- ☐ Perform schema validation automatically on ingestion against a metadata registry (e.g., data type, structure, and format)
- ☐ Use automated validators to flag, quarantine, or discard outliers and invalid records
- ☐ Confirm that deduplication logic is embedded in ETL jobs
- ☐ Check and monitor validity regulations regularly as business needs change

INTEGRITY

- ☐ Enforce database constraints (e.g., primary keys, foreign keys) to maintain referential integrity
- ☐ Execute cross-table validation scripts to detect inconsistencies and reference violations across related tables
- ☐ Track data lineage metadata to verify that derived tables accurately map back to their source systems
- ☐ Verify parent-child relationships between related tables during routine data quality audits

2. Monitor Data Quality Continuously

As systems evolve, data should be monitored continuously to maintain reliability. Putting the right checks in place (e.g., automated alerts, performance metrics) makes it easier to catch problems early without relying on manual reviews. When these tools are integrated into daily workflows, teams can respond faster to issues, reduce risk, and build trust in the data that powers their analytics and AI systems across the organization:

- ☐ Implement automated tools to detect anomalies (e.g., nulls, schema drift)
- ☐ Automate profiling and integrate into pipelines before production deployment
- ☐ Profile datasets regularly and align frequency with data volatility (e.g., daily, weekly)
- ☐ Integrate checks into ETL workflows with alerts and custom rules for batch/streaming data
- ☐ Eliminate manual checks using threshold logic and statistical anomaly detection
- ☐ Create dashboards that display key metrics; use targets and color indicators to highlight issues and track trends
- ☐ Enable drill-down views to trace problems to their source
- ☐ Assign data quality ownership across teams with defined KPIs
- ☐ Promote shared accountability through visibility and ongoing reporting

3. Strengthen Data Governance and Ownership

Strong data governance and clearly assigned data ownership are the foundation of high-quality data. **Governance** defines how data is *accessed*, *secured*, and *used* across an organization, while **ownership** ensures *accountability* for the data's *accuracy* and *proper use*. Together, they reduce risk, improve consistency, and turn data into a reliable business asset. With clear roles, well-documented policies, and proactive oversight, organizations can build trust in their data and meet regulatory demands without slowing innovation:

- ☐ Assign data owners to oversee dataset strategy, access, and quality for key datasets
- ☐ Designate data stewards to enforce governance standards and monitor data quality
- ☐ Establish core policies for access control, retention, sharing, and privacy
- ☐ Create and maintain a data catalog to centralize metadata and improve data discoverability

CHECKLIST CONTINUES ON NEXT PAGE

- ☐ Document and distribute governance policies covering usage, compliance, and security expectations
- ☐ Integrate governance controls into existing workflows and tools for enforcement
- ☐ Track compliance metrics to measure policy adherence and identify gaps
- ☐ Review and update governance practices regularly to keep pace with organizational and legal changes
- ☐ Promote a culture of responsibility around data through visibility and training

4. Track Data Lineage and Traceability

Understanding where data comes from, how it's transformed, and where it flows is crucial for debugging issues, meeting compliance requirements, and building trust. Data **lineage** provides that visibility, capturing the full history of every dataset across your ecosystem. From initial ingestion to final output, **traceability** helps ensure accuracy, enable audits, and support reproducibility. Implementing solid lineage practices with change tracking and version control creates transparency across both technical and business users:

- ☐ Map data origins and transformations across pipelines, including API sources, transactional systems, and flat files
- ☐ Capture lineage metadata to log merges, filters, and transformations for full processing visibility
- ☐ Integrate lineage tools with ETL processes to track changes from ingestion to output
- ☐ Log schema changes and dataset updates with metadata on *who* changed *what*, *when*, and *why*
- ☐ Maintain a version history for key datasets to support rollback and auditability
- ☐ Use version control tools to manage schema evolution and prevent conflicting updates in collaborative environments
- ☐ Retain historical lineage and transformation records to ensure reproducibility of results
- ☐ Trace anomalies to their source with minimal friction to support audits and investigations
- ☐ Link lineage insights with change logs and data dependencies to facilitate impact analysis

5. Validate Readiness for AI and Machine Learning

Preparing data for AI and machine learning requires thoughtful structuring and labeling, plus mitigating bias and ensuring the richness needed for deeper, more accurate predictions. Whether you're building a classification model or a real-time recommendation engine, upfront investment in data quality pays off in model performance, trust, and fairness:

- ☐ Label datasets with clear, granular, and compliant tags that match AI/ML model objectives
- ☐ Organize data into feature stores or structured tables with consistent formats, column names, and types
- ☐ Include essential metadata (e.g., timestamps, data source origins)
- ☐ Remove duplicates, fill or impute missing values, and standardize formats to reduce training errors
- ☐ Validate column consistency to prevent schema mismatches during modeling
- ☐ Document preprocessing steps to support reproducibility and troubleshooting
- ☐ Detect bias in features and outcomes using statistical tests (e.g., disparate impact ratio)
- ☐ Visualize demographic and feature distributions to surface imbalance or overrepresentation
- ☐ Apply mitigation techniques (e.g., re-sampling, synthetic data generation)
- ☐ Track audit results and interventions to maintain transparency and meet regulatory standards
- ☐ Include fine-grained data (e.g., geolocation, user logs) for deeper modeling
- ☐ Augment with external sources (e.g., economic indicators, demographics) where relevant
- ☐ Ensure datasets are dense enough to support pattern recognition and generalization without noise or sparsity

6. Ensure Data Security and Compliance

As industry and global regulations evolve and data volumes grow, ensuring privacy and protecting sensitive information is essential. Compliance frameworks like GDPR, CCPA, and HIPAA set legal expectations, but it's the combination of policy, process, and technical safeguards that keeps data protected and organizations accountable. Meeting these requirements, which can be done through the following steps, builds trust and reduces the risk of costly violations:

- ☐ Map datasets that include personal or regulated information across systems
- ☐ Audit consent management, user rights (access, correction, deletion), and breach notification procedures
- ☐ Review data residency requirements and ensure processing aligns with legal boundaries
- ☐ Document processing activities to support audits and demonstrate accountability
- ☐ Partner with legal, privacy, and security teams to track regulation changes
- ☐ Mask sensitive fields when using data in non-production or analytic environments

- ☐ Encrypt data at rest and in transit using TLS/SSL and secure encryption standards
- ☐ Apply field-level encryption for high-risk values (e.g., payment data)
- ☐ Enforce RBAC to restrict data access based on job function
- ☐ Implement key management and rotation policies to protect decryption credentials
- ☐ Combine masking and encryption to reduce the impact of any potential data breach

7. Invest in Culture and Continuous Improvement

Data quality requires sustained effort, clear processes, and a culture that values accuracy. By building structured review cycles and open feedback loops, and investing in data literacy, organizations can improve the reliability of their data while remaining aligned with their evolving AI and analytics needs. A consistent commitment to improvement ensures long-term value and trust in your data assets:

- ☐ Schedule regular data quality reviews (monthly, by delivery cycle)
- ☐ Evaluate core quality dimensions against historical benchmarks
- ☐ Document issues, trends, and resolutions to create a living archive of quality progress
- ☐ Integrate assessments into governance workflows to ensure accountability
- ☐ Set up clear communication channels between data producers and consumers
- ☐ Troubleshoot collaboratively to resolve issues quickly and define new data needs

- ☐ Highlight how upstream actions affect downstream outcomes to promote shared ownership
- ☐ Invest in data training programs to improve awareness of quality and responsible AI use
- ☐ Establish stewardship roles within each department to lead local quality efforts
- ☐ Celebrate quality improvements to reinforce positive behaviors

Conclusion

The impact of any AI or analytics initiative depends on the quality of the data behind it. Inaccurate, incomplete, or outdated data can erode trust, produce misleading results, waste valuable resources, and cause costly consequences. To avoid these pitfalls, organizations must take a well-rounded and comprehensive approach: assess data quality across the key dimensions, perform ongoing monitoring, adhere to governance and compliance practices, establish continuous feedback loops, and take action where gaps exist.

As regulations evolve and data demands grow, building a culture that values quality will set your organization apart. Ultimately, this entails regular reviews, targeted training, and investing in tools that embed data quality into

everyday practices. Using this checklist as a guide, you can take practical, proactive steps to strengthen your data and lay the foundation for responsible, high-impact AI. The payoff is clear: better decisions, greater trust, and a durable competitive advantage in a data-driven world.

Additional resources and related reading:

- "Data Governance Essentials: Policies and Procedures (Part 6)" by Sukanya Konatam
- "AI Governance: Building Ethical and Transparent Systems for the Future" by Sukanya Konatam
- *Getting Started With Data Quality* by Miguel Garcia Lorenzo, DZone Refcard
- *Data Pipeline Essentials* by Sudip Sengupta, DZone Refcard
- *Open-Source Data Management Practices and Patterns* by Abhishek Gupta, DZone Refcard
- *Machine Learning Patterns and Anti-Patterns* by Tuhin Chattopadhyay, DZone Refcard
- *AI Automation Essentials* by Tuhin Chattopadhyay, DZone Refcard
- *Getting Started With Agentic AI* by Lahiru Fernando, DZone Refcard
- AI Policy Labs

Sukanya Konatam
@sukanyakonatam
@sukanya-konatam

AI governance and data strategy leader, keynote speaker, and author driving ethical AI adoption, data governance, and innovation across industries. Passionate about shaping responsible technology and empowering communities through education and leadership.

ADDITIONAL RESOURCES

Solutions Directory

This directory contains data and BI tools for end-to-end data management. It provides pricing data and product category information gathered from vendor websites and project pages. Solutions are selected for inclusion based on several impartial criteria, including solution maturity, technical innovativeness, relevance, and data availability.

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY

2025 PARTNERS

Product	Purpose	Availability	Website
DataStax Astra DB	Built for AI: fast, scalable, cloud-native NoSQL database	Free tier	datastax.com/products/astra
Langflow	Build AI agents fast with a visual, production-ready tool	Free tier	langflow.org
Company	**Purpose**	**Availability**	**Website**
1010data Insights Platform	Analytics platform for big data analysis and data insights	By request	1010data.com
Accenture Data Services	Data and AI services	By request	accenture.com/us-en/services/data-ai
Actian DataConnect	Low-code data integration platform	By request	actian.com/data-integration/dataconnect
Actian DataFlow	Data extraction, analysis, transformation, and loading	By request	actian.com/data-integration/dataflow
Airbyte Cloud	Fully managed cloud data integration	Trial period	airbyte.com/product/airbyte-cloud
AirByte Open Source	Data integration for ETL/ELT data pipelines	Open source	airbyte.com/product/airbyte-open-source
Alation Data Intelligence Platform	Manage, discover, and govern data	Trial period	alation.com/product/data-intelligence-platform
Alluxio	Data platform for AI and analytics	Free tier	alluxio.io
Altair Grid Engine	Distributed resource management and optimization	Trial period	altair.com/grid-engine
Altair Monarch	Self-service data preparation	Trial period	altair.com/monarch
Altair Panopticon	Data visualization and streaming analytics	By request	altair.com/panopticon
Altair RapidMiner Platform	Data analytics and AI platform	By request	altair.com/altair-rapidminer
Altair Unlimited	Data analytics appliance	By request	altair.com/altair-unlimited-data-analytics
Alteryx One Platform	Unified analytics, data prep, and AI automation	Trial period	alteryx.com/products/alteryx-platform
Amazon Athena	Analyze petabyte-scale data	By request	aws.amazon.com/athena
Amazon EMR	Big data platform	By request	aws.amazon.com/emr
Amazon Kinesis	Process and analyze real-time streaming data	By request	aws.amazon.com/kinesis
Amazon Redshift	Cloud data warehouse	Trial period	aws.amazon.com/redshift
AWS Glue	Serverless data integration	Trial period	aws.amazon.com/glue
AWS Lake Formation	Secure data lake	Trial period	aws.amazon.com/lake-formation
Analytics8	Data analytics consulting services	By request	analytics8.com

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY			
Product	**Purpose**	**Availability**	**Website**
Apache Beam	Unified batch and streaming data processing	Open source	beam.apache.org
Apache Hudi	Incrementally process, manage, and query large data lakes	Open source	hudi.apache.org
Apache Kafka	Distributed event streaming platform	Open source	kafka.apache.org
Apache Kudu	Distributed data storage engine	Open source	kudu.apache.org
Apache NiFi	Automate data flow management, transformation, and system integration	Open source	nifi.apache.org
Apache Ozone	Scalable, distributed storage for analytics, big data, and cloud-native apps	Open source	ozone.apache.org
Apache Paimon	Streaming data lake platform for real-time data management	Open source	paimon.apache.org
Apache Pinot	Real-time analytics platform	Open source	pinot.apache.org
Apache Pulsar	Cloud-native distributed messaging and streaming platform	Open source	pulsar.apache.org
Apache Spark	Unified analytics engine for large-scale data processing	Open source	spark.apache.org
Apache Storm	Distributed stream processing computation framework	Open source	storm.apache.org
Apache Superset	Modern data exploration and visualization platform	Open source	superset.apache.org
Argo Workflows	Kubernetes-native workflow engine to orchestrate parallel jobs	Open source	argoproj.github.io/workflows
Ascend	Automated DataOps for end-to-end pipeline orchestration	By request	ascend.io
Astronomer Astro	Orchestration-first DataOps platform	Trial period	astronomer.io
AtScale	Universal semantic layer tool	Free tier	atscale.com
BMC Control-M	Simplify application and data workflow orchestration	By request	bmc.com/it-solutions/control-m.html
CelerData Cloud	Data lakehouse analytics	Trial period	celerdata.com/celerdata-cloud
Census Universal Data Platform	Data transformation, activation, and governance	Free tier	getcensus.com
Civis Analytics	Managed data warehouse and AI platform	By request	civisanalytics.com
ClickHouse	Open columnar OLAP SQL database	Open source	clickhouse.com/clickhouse
Cloudera Data Platform	Hybrid data management and analytics	Trial period	cloudera.com/products/cloudera-data-platform.html
Coginiti	AI-enabled collaborative data operations platform	Trial period	coginiti.co
Confluent Cloud	Fully managed Kafka as a service	Free tier	confluent.io/confluent-cloud
Confluent Platform	Enterprise-grade Apache Kafka distribution	Trial period	confluent.io/product/confluent-platform
Dagster Labs Dagster+	Data orchestrator for managing and deploying data pipelines	Trial period	dagster.io/plus
DAS42	End-to-end data consulting and implementation	By request	das42.com
Databricks	Data intelligence platform	Trial period	databricks.com
Dataddo	Fully managed, no-code data integration platform	Free tier	dataddo.com
Dataiku	Build, deploy, and manage data, analytics, and AI	Trial period	dataiku.com

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY			
Product	**Purpose**	**Availability**	**Website**
Datameer Cloud	Data integration, transformation, and analytics	By request	datameer.com/cloud
Datameer X	Data analytics platform for data lakes	By request	datameer.com/dmx
dbt Labs dbt (Data Build Tool)	SQL-based transformation tool for analytics engineering	Trial period	getdbt.com/product/dbt
Delta Lake	Storage framework for building lakehouse architecture	Open source	delta.io
Digdag	Build, run, schedule, and monitor complex pipelines of tasks	Open source	digdag.io
Domo Data Experience Platform	AI-powered data integration, automation, and visualization	Trial period	domo.com
Dremio Cloud	Fully managed unified data lakehouse platform	Trial period	dremio.com/cloud
Dremio Community	Self-managed data lakehouse	Free	dremio.com/community-edition
DuckDB	Analytical in-process SQL DBMS	Open source	github.com/duckdb/duckdb
DuckDB DuckLake	Integrated data lake and catalog format	Open source	github.com/duckdb/ducklake
Elasticsearch	Distributed search and analytics engine, scalable data store, vector DB	Open source	github.com/elastic/elasticsearch
Embulk	Pluggable bulk data loader	Open source	embulk.org
Exaptive Cognitive City	Collaborative data exploration and insights	Trial period	exaptive.com/cognitive-city
Exaptive Studio	Data visualization software	Trial period	exaptive.com/studio
Exasol Analytics Engine	AI-integrated analytics database	Free tier	exasol.com/analytics-engine
FICO Platform	Applied intelligence platform	By request	fico.com/en/fico-platform
Fivetran	Automated data movement platform	Free tier	fivetran.com
GoodData	AI-powered analytics	Trial period	gooddata.com
Google Cloud Big Query	Enterprise data warehouse	Free tier	cloud.google.com/bigquery
Google Cloud CDAP	Data analytics platform	Free tier	cdap.io
Google Cloud Dataflow	Fully managed unified batch and stream pipeline service	Trial period	cloud.google.com/products/dataflow
Google Cloud Looker	Business intelligence platform	Trial period	cloud.google.com/looker
Great Expectations GX Cloud	End-to-end data quality management	Free tier	greatexpectations.io/cloud
Great Expectations GX Core	Open framework for data quality testing and validation	Open source	greatexpectations.io/gx-core
Hazelcast Platform	Distributed compute engine and fast data store	Free tier	hazelcast.com/products/hazelcast-platform
Hevo Pipeline	End-to-end ELT with built-in transformations	Free tier	hevodata.com/pipeline
Hightouch	Composable customer data platform and reverse ETL	Free tier	hightouch.com
Hitachi Vantara Virtual Storage Platform One	Hybrid cloud data platform	By request	hitachivantara.com/en-us/products/storage-platforms/data-platform

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY

Product	Purpose	Availability	Website
HPE GreenLake for Block Storage	AI-managed, self-service block storage	By request	hpe.com/us/en/hpe-greenlake-block-storage.html
HPE GreenLake for File Storage	Enterprise-grade, scale-out file storage	By request	hpe.com/us/en/hpe-greenlake-file-storage.html
IBM Cloud Pak for Data	Data analysis, organization, and management	Trial period	ibm.com/products/cloud-pak-for-data
IBM InfoSphere Information Server	Data integration platform	By request	ibm.com/information-server
IBM StreamSets	Develop and manage data pipelines	Trial period	ibm.com/products/streamsets
Incorta	Streamline data acquisition, integration, analysis, and visualization	By request	incorta.com
Indium Software ibriX	Databricks AI platform	By request	indium.tech/ibrix
Indium Software iDAF	Data assurance framework	By request	indium.tech/idaf
Infor Data Lake	Enterprise data storage	By request	infor.com/products/data-lake
Informatica Intelligent Data Management Cloud	AI-powered data management	Free tier	informatica.com/platform.html
insightsoftware Logi Symphony	Simplify data visualization	By request	insightsoftware.com/logi-symphony
InterSystems Data Fabric Studio	Build and manage a unified, governed, real-time data fabric	By request	intersystems.com/products/intersystems-data-fabric-studio
InterSystems IRIS	Cloud-first data platform for high-performance applications	Trial period	intersystems.com/products/intersystems-iris
InterSystems Supply Chain Orchestrator	High-performance data management for supply chains	By request	intersystems.com/products/supply-chain-orchestrator
Jellyfish Engineering Management Platform	Data insights on engineering team performance and management	By request	jellyfish.co/platform/engineering-management-platform
Keboola	Unify, automate, and manage end-to-end data workflows	Free tier	keboola.com
Keebo Warehouse Optimization	Fully automated Snowflake cost and performance optimization	Trial period	keebo.ai
Kyligence Enterprise	Simplify multidimensional data analysis	By request	kyligence.io/kyligence-enterprise
Kyligence Zen	Decision intelligence platform	Trial period	kyligence.io/zen
lakeFS Enterprise	Git-like version control for data lakes	Trial period	lakefs.io/enterprise
lakeFS Open Source	Open Git-like version control for data lakes	Open source	github.com/treeverse/lakeFS
Lightbend Akka	Toolkit for building APIs, microservices, and data streams	Free tier	lightbend.com/akka
Luigi (by Spotify)	Python module to help build complex pipelines of batch jobs	Open source	github.com/spotify/luigi
Mammoth Analytics	No-code data management	By request	mammoth.io
Matallion Data Productivity Cloud	Build and manage pipelines and deliver data for AI and analytics	Trial period	matillion.com
Melissa Data Quality Suite	Verify and correct contact data at point of entry	By request	melissa.com/data-quality-suite
Melissa Unison	Data profiling, cleansing, matching/deduping, and monitoring	By request	melissa.com/customer-data-validation-platform

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY			
Product	**Purpose**	**Availability**	**Website**
Melissa Web APIs	Scalable and flexible data validation APIs	By request	melissa.com/developer
Meltano by Arch	Orchestrate and version control ELT pipelines via code-first CLI	Open source	meltano.com/product
Microsoft Azure Data Factory	Cloud-based data integration service on Azure	Trial period	azure.microsoft.com/en-us/products/data-factory
Microsoft Azure Databricks	Open data lakehouse	Trial period	azure.microsoft.com/en-us/products/databricks
Microsoft Azure Stream Analytics	Serverless real-time analytics from cloud to edge	Trial period	azure.microsoft.com/en-us/products/stream-analytics
Microsoft Azure Synapse Analytics	Enterprise analytics service to accelerate time to insight	Trial period	azure.microsoft.com/en-us/products/synapse-analytics
Microsoft Power BI	Data visualization	Free tier	powerbi.microsoft.com/en-us
MinIO AIStor Enterprise	Hyperscale, exascale-ready S3-compatible object storage	Trial period	min.io/product/aistor-overview
MinIO AIStor Open Source	Open hyperscale, exascale-ready S3-compatible object storage	Open source	github.com/minio
MongoDB Atlas	Multi-cloud developer data platform	Trial period	mongodb.com/atlas
MongoDB Compass	Query, optimize, and analyze MongoDB data	Free	mongodb.com/products/tools/compass
Monte Carlo	Data and AI observability	By request	montecarlodata.com
MoreSteam EngineRoom	Analysis, statistical tools, and process improvement insights	Trial period	moresteam.com/engineroom
Neo4j Aura Graph Analytics	Ad hoc graph processing without ETL	Trial period	neo4j.com/product/aura-graph-analytics
Neo4j AuraDB	Fully managed graph database	Free tier	neo4j.com/product/auradb
Neo4j Bloom	Visual data exploration	Free tier	neo4j.com/product/bloom
Neo4j Graph Data Science	Analytics and ML solution for improved data-based predictions	Free tier	neo4j.com/product/graph-data-science
Neo4j Graph Database	Self-managed high-speed, scalable graph database	Free tier	neo4j.com/product/neo4j-graph-database
Neo4j GraphQL	GraphQL API library	Open source	neo4j.com/product/graphql-library
Octolis	Headless customer data platform	By request	octolis.com
Omnata Connect	Live-query data warehouse integration for Salesforce	Trial period	omnata.com/products/connect
Omnata Sync	Snowflake-native app for bi-directional SaaS and database syncing	Trial period	omnata.com/products/sync
OpenLineage	Open framework for data lineage collection and analysis	Open source	openlineage.io
OpenText Analytics Cloud	AI-powered, real-time analytics platform	By request	opentext.com/products/ai-and-analytics
Oracle Analytics Platform	Analytics cloud and server platform	By request	oracle.com/business-analytics/analytics-platform
Panoply by SQream	Cloud data warehouse platform	Trial period	panoply.io
Pepperdata Capacity Optimizer	Autonomous cost optimization	By request	pepperdata.com/product/capacity-optimizer-cost-optimization

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY

Product	Purpose	Availability	Website
Perforce Delphix DevOps Data Platform	Multi-cloud data automation, compliance, and governance	By request	perforce.com/products/delphix
PK Protect Data Protection Platform	Automated data discovery and protection	By request	pkware.com/products
Polytomic	Bi-directional ETL and data syncing	By request	polytomic.com/overview
Polytomic Connect	API for two-way ETL and data syncing with customers	By request	polytomic.com/connect
Prefect Cloud	Orchestrate, monitor, and automate data workflows	Free tier	prefect.io/cloud
Prefect Open Source	Modern data workflow orchestration	Open source	prefect.io/opensource
Presto	SQL query engine	Open source	prestodb.io
Progress Data Platform	End-to-end data and AI integration suite	Trial period	progress.com/data-platform
Promethium	Self-service analytics and data management	Trial period	promethium.ai/product-overview
Prophecy Data Copilot	AI assistant for data transformation and visualization	Trial period	prophecy.io/prophecy-data-copilot
Prophecy Data Integration Platform	Visual interface, AI, and compiler for data transformation	Trial period	prophecy.io/data-integration-platform-overview
PyPI Petl	Lightweight Python ETL framework	Open source	pypi.org/project/petl
Pyramid Decision Intelligence Platform	Frictionless, integrated, no-code, AI-driven data platform	By request	pyramidanalytics.com/decision-intelligence-platform
Qlik Cloud Analytics	Explore, analyze, and share data insights with AI-powered analytics	Trial period	qlik.com/us/products/qlik-cloud-analytics
Qlik Talend Data Integration and Quality	Unify, clean, and govern data with real-time, AI-driven pipelines	By request	qlik.com/us/products/qlik-talend-data-integration-and-quality
Qubole	Open and secure multi-cloud data lake platform	Trial period	qubole.com
Redpanda	Scalable, Kafka-compatible streaming data platform	Trial period	redpanda.com
Redpoint CDP	Composable customer data platform	By request	redpointglobal.com/cdp
Rivery	Cloud-native ETL and DataOps platform	Trial period	rivery.io/product
Rudderstack	Warehouse-native customer data platform	Free tier	rudderstack.com
Salesforce Data Cloud	Unify customer data in real time	By request	salesforce.com/data
Salesforce Tableau	Flexible, AI-powered analytics platform	Trial period	tableau.com/products/tableau
SAP HANA Cloud	Build and deploy intelligent data applications at scale	Trial period	sap.com/products/technology-platform/hana.html
SAP SAS Viya	AI and analytics platform	Trial period	sas.com/en_us/software/viya.html
Secoda	AI-powered data catalog, observability, and governance	Free tier	secoda.co
Siemens Xcelerator	Open digital platform with suite of industrial and IoT data solutions	Free	xcelerator.siemens.com/global/en.html
Singer	Open ETL standard using modular taps and targets	Open source	singer.io
Sisense Platform	AI analytics platform	By request	sisense.com/ai-analytics-platform
Snowflake	Cloud data platform	Trial period	snowflake.com/en

DZONE'S 2025 DATA ENGINEERING SOLUTIONS DIRECTORY

Product	Purpose	Availability	Website
Snowplow Customer Data Infrastructure	Real-time, governed customer behavioral data platform	Free tier	snowplow.io
Soda	Data quality platform for testing, monitoring, and fixing pipelines	Free tier	soda.io
SQream Blue	Data preparation lakehouse	Trial period	sqream.com/product/blue
Starburst Galaxy	Fully managed data lakehouse	Free tier	starburst.io/platform/starburst-galaxy
StarRocks	High-performance analytical database	Open source	starrocks.io
StarTree Cloud	Real-time analytics platform powered by Apache Pinot	By request	startree.ai/products/startree-cloud
Strategy One	AI and BI analytics platform	Trial period	strategysoftware.com/strategyone
StreamNative ONE Platform	Kafka/Pulsar-compatible engine for real-time streaming and analytics	Free tier	streamnative.io/products/one-streamnative-platform
Talend Data Fabric	Data integration pattern	Trial period	talend.com/products/data-fabric
Talend Stitch by Qlik	Cloud-first ETL service for data piplines	Trial period	stitchdata.com
TARGIT Decision Suite	BI and analytics tool	By request	targit.com/targit-decision-suite
Tecton	Feature store for scalable real-time ML	Trial period	tecton.ai
Teradata VantageCloud	Cloud analytics and data platform	Trial period	teradata.com/platform/vantagecloud
teX.ai	Text analytics accelerator	By request	tex-ai.com
ThoughtSpot Analytics	AI-powered analytics for modern data stacks	Trial period	thoughtspot.com/product/analytics
ThoughtSpot Embedded	Design and embed AI-powered analytics experiences	Trial period	thoughtspot.com/product/embedded
TIBCO Platform	Real-time, composable data platform	By request	tibco.com/platform
Treasure Data	Intelligent customer data platform	By request	treasuredata.com/product
Trino	Distributed SQL query engine for big data analytics	Open source	trino.io
Trocco	Fully managed no-code ELT and reverse-ETL	Free tier	global.trocco.io/product-overview
Twilio Segment	AI-powered customer data platform	By request	segment.com/customer-data-platform
Unravel	Data observability platform	Free tier	unraveldata.com
VMWare Tanzu Gemfire	High-speed in-memory data and compute grid	By request	tanzu.vmware.com/gemfire
VMWare Tanzu Greenplum	Data warehouse, analytics, and AI platform for unifying data	By request	vmware.com/products/app-platform/tanzu-greenplum
Volt Active Data	Real-time data processing platform	Free tier	voltactivedata.com
Whaly	Self-service AI-powered business intelligence	By request	whaly.io
Yellowbrick	SQL data platform	Trial period	yellowbrick.com
Yellowfin Analytics Platform	Enterprise and embedded analytics platform	By request	yellowfinbi.com/suite

DZone

DATA ENGINEERING

AI/ML | Databases | Big Data Data | Internet of Things

As you determine the first steps for new systems or reevaluate existing ones, you're going to require tools and resources to gather, store, and analyze data.

The Zones within our Data Engineering category contain resources that will help you expertly navigate through the SDLC Analysis stage.

VISIT THE ZONE

TUTORIALS

TECHNIQUES

CODE SNIPPETS

RESEARCH

DZone

3343 Perimeter Hill Dr, Suite 100
Nashville, TN 37211
888.678.0399 | 919.678.0300

At DZone, we foster a collaborative environment that empowers developers and tech professionals to share knowledge, build skills, and solve problems through content, code, and community. We thoughtfully — and with intention — challenge the status quo and value diverse perspectives so that, as one, we can inspire positive change through technology.