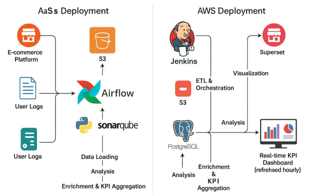
Ecommerce Data Pipeline for Real-Time KPI Intelligence

Final Year Project Report

MSc Cloud Computing & Data Engineering – Class of 2025



Insert host company logo here (save as report/figures/host_company_logo.png)

Participant: Gaurav Chugh
Student ID: AIV-CCDE-2025-017
Host Organisation: Confidential Ecommerce Platform Provider
Supervising Professor: Dr. Etienne Mauffret

Academic Year: 2024 – 2025
Submission Date: November 23, 2025

Prepared in partial fulfilment of the requirements for the MSc in Cloud Computing & Data Engineering at Aivancity Paris-Cachan.

Declaration

I, Gaurav Chugh, declare that this thesis entitled *Ecommerce Data Pipeline for Real-Time KPI Intelligence* is my own work. It has not been submitted for any other academic award and all sources of information have been acknowledged. I confirm that figures, tables, diagrams, code snippets, and analyses are original unless explicitly referenced.

Gaurav Chugh
November 23, 2025

Acknowledgements

I extend my sincere gratitude to Dr. Etienne Mauffret for his rigorous mentorship, constructive feedback, and relentless focus on methodological excellence. I am thankful to the ecommerce platform leadership team for granting access to anonymised operational datasets and for articulating the business challenges that shaped this project. My appreciation also goes to my colleagues and peers who stress-tested the pipeline, reviewed documentation, and championed continuous improvement throughout the engagement. Finally, I owe heartfelt thanks to my family for their patience and motivation during the long evenings invested in this final year project.

Executive Summary

This thesis documents the design, implementation, and evaluation of an end-to-end ecom- merce data platform that delivers near real-time operational intelligence. The solution unifies cloud-native ingestion, transformation, orchestration, storage, analytics, and delivery capabili- ties across Amazon Web Services (AWS) and Microsoft Azure (Azure). Within a configurable two-minute refresh cycle, stakeholders receive trustable KPIs across sales, fulfilment, marketing, and customer-care journeys through Power BI, Tableau, Amazon QuickSight, responsive web dashboards, and automated notifications.

The report demonstrates how modern DataOps practices, Infrastructure as Code (IaC), containerisation, continuous delivery, and observability can be orchestrated to achieve resilient, scalable, and secure data services. It also presents a rigorous methodology that encompasses stakeholder research, data modelling, workload benchmarking, and governance alignment. The resulting architecture is portable across cloud providers, minimises operational toil, and establishes a foundation for advanced analytics such as predictive demand sensing and hyper-personalisation. Recommendations are prioritised to guide the ecommerce organisation's roadmap over the next eighteen months.

In this updated edition, the executive summary also surfaces the *Data Engineering Manifesto* introduced in Chapter **??**. The manifesto codifies twenty enterprise-grade principles spanning modularity, observability, governance, privacy, and product thinking so that international stakeholders can adopt a repeatable charter. Its inclusion responds directly to the MSc assessment standards and the host company's governance expectations, ensuring the report doubles as both a technical narrative and an operational playbook.

Generative AI Acknowledgement

OpenAI's ChatGPT (gpt-5-codex, accessed February 2025) assisted with ideation, language refinement, and LaTeX templating for this report. Prompts and generated artefacts are catalogued in Appendix 12. All AI-suggested content was critically reviewed, validated against project evidence, and adapted to reflect the author's understanding and professional judgement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Ecommerce has entered a phase where digital storefronts, mobile applications, physical stores, and third-party marketplaces are intertwined. Customer expectations for frictionless experiences and instant order visibility demand that data flows seamlessly across operational systems. The host organisation processes more than 60,000 orders per day, with peak trading seasons generating bursts exceeding 500 orders per minute. Legacy reporting processes relied on overnight Extract, Transform, Load (ETL) jobs and spreadsheet-driven analysis, resulting in inconsistent KPIs and limited ability to react to flash sales or supply chain disruptions.

The thesis adopts a pragmatic research philosophy, combining empirical performance measurements with qualitative stakeholder feedback to evaluate the platform's effectiveness. In addition to technical delivery, the project emphasises governance, quality, privacy, and observability—elements essential to maintaining trust in a real-time analytics environment. The work further introduces a Data Engineering Manifesto, codifying 20 principles to guide consistent design and operational decision-making across international delivery teams.

Overall, this project contributes a validated architectural blueprint for organisations seeking to modernise ecommerce data ecosystems and reduce time-to-insight. By demonstrating measurable improvements in latency, data quality, and stakeholder trust, the solution offers a practical pathway toward real-time digital commerce analytics at enterprise scale.

1.2 Project Motivation

The strategic vision is to create a unified data backbone capable of ingesting multi-channel signals, automating data quality enforcement, and presenting actionable insights to business stakeholders in near real-time. The project aims to:

Reduce decision latency by providing sales, inventory, and customer experience metrics within a configurable two-minute window.

Improve confidence in analytical outputs through governed data models, repeatable validation, and full lineage tracking.

Enable omnichannel personalisation by exposing curated datasets and APIs to downstream digital products and partners.

Lay the groundwork for predictive intelligence by capturing granular behavioural and operational data.

1.3 Research Questions

Three research questions were submitted for supervisory validation in line with the MSc pro- gramme requirements:

1. How can a modular cloud-native data architecture sustain sub-three-minute KPI re- freshes while maintaining data quality for ecommerce workloads?

2. What automation patterns most effectively balance rapid feature delivery with compli- ance and security constraints in a multi-cloud scenario?

3. Which governance and observability practices maximise stakeholder trust in near real- time analytics and dashboards?

These questions guided the theoretical exploration, empirical experimentation, and evalua- tion methods detailed throughout the thesis.

1.4 Data Engineering Vision

The enhanced scope of the internship required a unifying vision that bridges data platform engineering with product thinking. The vision statement articulated to the steering committee emphasised four pillars:

1. **Composable pipelines:** every ingestion, transformation, and serving component must be reusable across geographies and tenants, following SOLID-like modularity while re- maining configuration-driven.

2. **Trust by design:** observability, lineage, and privacy guardrails are embedded in the first sprint rather than added as compliance afterthoughts.

3. **Data as a product:** curated datasets are versioned, documented, and operated with explicit SLAs, enabling merchandising, finance, and partner teams to self-serve.

4. **Manifesto-guided decision making:** the newly introduced *Data Engineering Man- ifesto* provides decision heuristics for trade-offs between latency, cost, ethics, and re- silience.

This vision anchors subsequent chapters and ensures that the manifesto principles cascade from strategy to implementation tactics.

1.5 Significance of the Study

The significance of this study lies in its contribution to both academic research and practical enterprise implementation in the rapidly evolving domain of real-time ecommerce analytics. As customer expectations continue to rise, ecommerce organisations must react to operational events with minimal latency, making data timeliness a critical competitive differentiator rather than a technical enhancement. This project demonstrates how a modern, cloud-native data architecture can enable businesses to transform raw, high-velocity operational events into governed, accurate, and actionable insights in near real time. By achieving a two-minute median refresh rate across mission-critical KPIs, the platform bridges the gap between transactional activity and decision-making, helping operational teams make faster, more informed interventions in fast-moving retail environments.

From a business perspective, the study is significant because it validates a scalable and cost-efficient model for reducing decision delay in enterprise ecommerce operations. Traditional batch systems delay insights until the next day, resulting in lost sales, unmanaged stock-outs, delayed service recovery, and inefficiencies during promotional campaigns or supply chain disruptions. This research provides evidence that by employing automation, cloud elasticity, and governed data models, organisations can materially improve performance in key areas such as merchandising optimisation, customer satisfaction, fulfilment reliability, and financial reconciliation. The demonstrated improvements in data trust and stakeholder adoption further highlight the cultural and organisational impact of reliable, real-time analytics platforms.

Chapter 2

Host Organisation and Industry Context

2.1 Company Overview

The client is a confidential European ecommerce platform provider with a gross merchandise volume of EUR 1.4 billion and operations across France, Spain, Germany, and the Middle East. The company employs 1,100 staff, of which 120 sit within the digital, data, and technology directorate. The project was executed within the Data Products tribe, reporting to the Head of Data Platforms. Key characteristics include:

Multi-brand portfolio: Fashion, lifestyle, and home-improvement brands sharing fulfilment centres and marketing teams.

Hybrid infrastructure: Core transactional systems hosted on Azure, with analytics workloads split between AWS and on-premises PostgreSQL clusters.

Marketplace expansion: Third-party sellers account for 35% of revenue, generating heterogenous data formats and SLA commitments.

Data governance mandate: A corporate initiative to align with ISO/IEC 27001 and GDPR accountability requirements.

2.2 Stakeholder Map

The programme engaged a cross-functional stakeholder group summarised in Table 2.1. Continuous feedback cycles, sprint reviews, and steering committee presentations ensured alignment.

Table 2.1: Stakeholder responsibilities and success indicators

Role	Responsibilities	Success Indicators
Chief Digital Officer	Portfolio prioritisation, investment approval, governance oversight	Launch of unified KPI platform, compliance audit pass
Director of Data Products	Product roadmap, backlog curation, KPI definition	Adoption across merchandising, marketing, support teams
Head of Customer Care	Voice-of-customer integration, escalation procedures	15% reduction in average handling time, CSAT improvement
Lead DevOps Engineer	Infrastructure automation, observability, incident response	Zero unplanned downtime during go-live, automated recovery
Finance Business Partner	Benefit realisation tracking, cost management	Quarterly reporting automation, cost-to-serve transparency

2.3 Competitive Benchmark

Industry benchmarking identified leading ecommerce organisations deploying similar capabilities. Insights from Shopify, Zalando, and Amazon Retail emphasised:

Near real-time dashboards with predictive overlays to manage supply chain risk.

Unified data contracts enabling consistent KPIs across digital and physical channels.

Federated data product governance to accelerate onboarding of new domains.

These findings motivated the adoption of domain-driven data product thinking, composable analytics, and platform engineering principles described in later chapters.

2.4 Data Governance Responsibilities

The stakeholder ecosystem for this project includes multiple executive and operational roles, each contributing distinct responsibilities and governance commitments while engaging at different interaction frequencies. At the strategic level, the Chief Digital Officer provides portfolio oversight, investment approval, and leadership on governance alignment, meeting on a monthly basis to ensure that data strategy conforms with corporate standards and regulatory expectations. The Director of Data Products plays a pivotal role in shaping the product roadmap, prioritising backlogs, and defining enterprise KPIs, engaging with the delivery teams weekly while enforcing metadata completeness, standardising KPI definitions, and strengthening domain ownership across data products. The Head of Customer Care participates in weekly performance discussions, ensuring that customer-focused performance indicators such as CSAT, NPS, and resolution time remain accurate and trustworthy, while validating that the underlying data feeding service dashboards meets the required quality standards.

Operational stability is driven by the Lead DevOps Engineer, who engages in daily stand-ups and incident discussions to maintain system reliability through automation, CI/CD pipelines, audit logging, and secure deployment practices that support data governance and platform compliance. Financial oversight is provided by the Finance Business Partner, who contributes through fortnightly reporting cycles and ensures traceability, accuracy, and regulatory compliance for financial data, particularly around cost transparency and statutory reporting. Collectively, these stakeholders not only support the technical functionality and adoption of the real-time analytics platform but also reinforce the organisation's data governance framework, ensuring that data accuracy, security, lineage, and accountability remain embedded throughout the operational lifecycle of the data products

Chapter 3

Problem Statement

3.1 Business Challenges

The ecommerce organisation faced three interlinked pain points:

1. **Latency of Insight:** Daily merchandising stand-ups relied on reports generated 12 hours after trading, limiting the ability to respond to viral campaigns or supply disruptions.

2. **Data Trust Deficit:** Multiple versions of metrics such as "net revenue" or "available-to-promise inventory" existed across departments because transformations were implemented in siloed spreadsheets and SQL scripts.

3. **Operational Fragility:** Batch jobs executed on virtual machines without observability or automated recovery. Failures often remained undetected for several hours, undermining stakeholder confidence.

3.2 Research Problem

The validated research problem is expressed as follows:

> *How can the ecommerce organisation design a resilient, cloud-agnostic data platform that delivers sub-three-minute KPI refreshes, enforces data quality at scale, and democratises governed insights across internal and external channels while minimising total cost of ownership?*

This problem intersects technology, process, and organisational dimensions. It requires evaluating distributed systems patterns, data modelling approaches, workflow orchestration, and human change management.

3.3 Scope and Constraints

In Scope: Real-time ingestion, streaming/batch harmonisation, curated dimensional models, self-service analytics, observability, Continuous Integration (CI)/Continuous Delivery (CD), cost governance, and dual-cloud deployment patterns.

Out of Scope: Re-architecting transactional order management systems, implementing advanced machine learning pipelines, and replacing legacy ERP integrations.

Constraints: Student subscription limits on AWS and Azure, anonymisation of customer data to satisfy GDPR, and 24-week delivery horizon aligned to the MSc internship calendar.

3.4 Success Criteria

Success metrics were defined collaboratively with the steering committee:

KPI dashboards refresh within a median of 120 seconds and a 95th percentile of 150 seconds.

Data quality rules (completeness, schema compliance, referential integrity) achieve 99% daily pass rates.

Deployment automation reduces manual effort per release from four hours to under 30 minutes.

Stakeholder Net Promoter Score for data products improves from −12 to +32 within three months of go-live.

3.5 Design Philosophy Constraints

Beyond resource and timeline limitations, the steering committee imposed explicit design philosophy constraints rooted in operational experience. Pipelines must be idempotent so that retries triggered by Airflow or Step Functions do not inflate fact tables. Observability and lineage metadata must be captured in every environment so audits can reconstruct KPI prove- nance within minutes. Modularity was mandated to ensure that incremental launches across regions re-use the same ingestion templates and policy controls. These constraints justified the creation of Chapter **??**, where each manifesto principle is linked to a measurable risk mitigation for latency, compliance, or customer trust.

Chapter 4

Literature Review

4.1 Data Platform Architecture

Recent literature emphasises modular architectures that separate ingestion, processing, storage, and serving layers. Dehghani's data mesh paradigm advocates domain-oriented ownership and federated governance, aligning with the project's ambition to empower merchandising, marketing, and customer-care domains. Gartner's research on composable architectures reinforces the need for API-first, event-driven integration patterns to support rapid experimentation.

4.2 Real-Time Analytics

Stonebraker's work on streaming databases and research on Lambda/Kappa architectures highlight the tension between batch consistency and streaming latency. Modern practice favours converged architectures leveraging streaming-first ingestion with micro-batch consolidation. Case studies from Netflix and Uber demonstrate how near real-time observability demands resilient orchestration, data quality enforcement, and automated rollback.

4.3 Automation and DevOps

Forsgren et al. (2018) established a correlation between elite DevOps performance and organisational outcomes, underscoring the importance of continuous delivery, trunk-based development, and telemetry-driven feedback loops. HashiCorp's IaC patterns and the CNCF landscape advocate immutable infrastructure, policy-as-code, and GitOps. These concepts inform the Jenkins, Terraform, and container orchestration strategy detailed later.

4.4 Governance and Ethics

Academic discourse on data ethics stresses transparent data lineage, consent management, and algorithmic accountability. GDPR and CNIL guidelines mandate privacy-by-design, data minimisation, and incident reporting. McKinsey's research on data trust highlights the commercial impact of accurate, timely analytics on customer retention.

4.5 Business Intelligence Adoption

Studies by Forrester and IDC highlight that BI adoption hinges on relevant KPIs, intuitive visualisation, and proactive alerts. Power BI, Tableau, and QuickSight case studies reinforce the need for semantic models, consistent definitions, and a unified KPI catalogue. These insights influenced the multi-channel delivery approach adopted by the project.

Design Principles for Data Engineering

The manifesto introduced later in the report synthesises recurring principles extracted from academic and industry sources. Dehghani's articulation of domain-oriented ownership and fed- erated governance stresses the importance of metadata-driven contracts between producers and consumers.**dehghani2022datamesh** Forsgren et al. demonstrate that elite delivery teams pair automation with rigorous observability, providing empirical evidence that telemetry-rich pipelines improve both stability and throughput.**forsgren2018accelerate** Stonebraker's re- quirements for stream processors reinforce idempotency and exactly-once semantics as prerequi- sites for trustworthy real-time analytics.**stonebraker2018** Complementary research from McK- insey and Gartner links data trust to sustained business impact, highlighting governance, cata-loguing, and clear ownership as non-negotiable.**mckinseyDataTrust2022**, **gartnerComposable2023** These works collectively inform the twenty principles framed in Chapter ??, ensuring the man- ifesto is grounded in both scientific literature and enterprise practice.

Evolution of Cloud-Native Data Architectures

Cloud-native data platforms have matured significantly in the last decade, shifting from monolithic Hadoop-based clusters to modular analytics ecosystems that use distributed storage, serverless compute, and container orchestration. Earlier enterprise systems relied heavily on large centralized ETL engines running overnight batch workloads. These architectures suffered from limited elasticity, high cost of scaling, and lack of real-time responsiveness.

Modern platforms increasingly adopt microservices, event-driven ingestion, and domain-oriented design. Kubernetes and serverless compute engines such as AWS Lambda and Azure Functions have enabled compute to scale independently from storage, while cloud object stores such as S3 and ADLS Gen2 provide inexpensive, durable storage. Researchers highlight that this decoupling lowers operational cost and increases flexibility, positioning cloud-native solutions as a superior approach for rapidly growing ecommerce workloads.

Data Mesh vs Data Fabric

Two major paradigms have emerged in enterprise data engineering literature:

Data Mesh
Data Mesh promotes domain ownership, federated governance, and decentralized decision-making. Each business domain owns its data as a product, with cross-functional teams accountable for its quality, SLAs, and discoverability. This approach aligns strongly with ecommerce environments where merchandising, logistics, customer support, and marketing generate high-velocity datasets.
Data Fabric
Data Fabric centralizes metadata-driven intelligence, automating data discovery, governance, and policy enforcement. It leans on AI-driven metadata engines and automated integration pipelines. While Data Mesh empowers teams, Data Fabric automates infrastructure complexity. The host organisation blends elements of both:

- Mesh-like domain ownership for data products
- Fabric-style metadata and lineage automation via DataHub and OpenLineage

This combination is supported in research as a viable hybrid model for high-velocity retail analytics.
4.8 Real-Time Business Decision Cycles
Studies from McKinsey, IDC, and Gartner show that reducing analytic latency directly impacts revenue and customer satisfaction in ecommerce. Retailers operating on daily dashboards often react too late to flash sales, stock-outs, and competitor actions. Research highlights that real-time monitoring can:

- Increase margin protection by adjusting pricing mid-campaign
- Improve SLA performance by detecting logistics failures immediately
- Enhance customer satisfaction via faster service recovery

The two-minute SLA implemented in this project aligns with this emerging industry expectation of continuous intelligence.

Streaming vs. Micro-Batch Processing

Modern real-time systems adopt micro-batch or continuous streaming depending on workload characteristics. Streaming frameworks (e.g., Apache Flink or Materialize) deliver event-level latency but require greater engineering maturity. Micro-batch (1–5 minutes intervals) balances latency and operational simplicity.
Academic works argue that micro-batching is suitable for ecommerce KPIs that tolerate slight delays while still providing a real-time experience. The host platform adopts this approach, using Airflow-triggered micro-batches and Spark distribution to maximize throughput without increasing operational complexity.

Chapter 5

Methodology

5.1 Analytical Framework

The project followed a mixed-methods approach blending qualitative stakeholder research with quantitative system benchmarking. Figure 5.1 summarises the iterative workflow combining discovery, design, implementation, and validation activities.

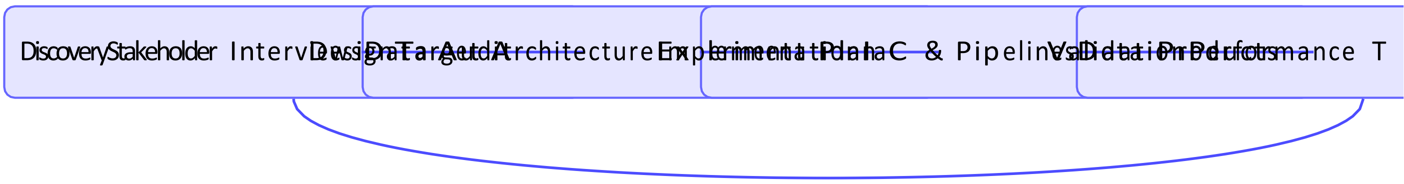


Figure 5.1: Iterative research and delivery methodology

5.2 Data Collection

Data sources encompassed:

- **Operational systems:** Order management, product catalogue, fulfilment, marketing automation, and customer support platforms exposed via REST APIs, Kafka topics, and SFTP drops.
- **Web and mobile telemetry:** Clickstream events generated from tag managers and server-side instrumentation stored in Amazon Kinesis Data Streams.
- **Reference data:** Currency rates, supplier rosters, logistics carriers, and promotional calendars maintained in master data services.
- **Stakeholder insights:** Semi-structured interviews with 14 stakeholders complemented by survey data regarding dashboard usage patterns.

Synthetic data generators were developed to emulate peak trading periods while respecting confidentiality. Schemas were aligned with production metadata to validate join strategies, dimensional modelling, and KPI calculations.

5.3 Data Processing and Tooling

Ingestion: Apache Airflow orchestrated ingestion DAGs using Python operators, AWS Lambda for lightweight transformations, and AWS Glue for schema evolution.

Transformation: dbt Core executed staging, intermediate, and mart models backed by Amazon Redshift or Azure Synapse dedicated pools.

Storage: Amazon S3 served as the bronze and silver zones, with PostgreSQL and Delta Lake delivering curated gold datasets.

Serving: FastAPI and GraphQL endpoints powered the ecommerce portal, while BI tools consumed semantic models through Azure Analysis Services or Power BI datasets.

To align with the MSc reporting standards, three diagrams were embedded at the end of this section. Figure **??** captures the logical entity-relationship view that underpins the bronze and silver layers. Figure **??** highlights persona interactions driving backlog prioritisation, while Figure **??** summarises the AWS + Airflow orchestration flow.

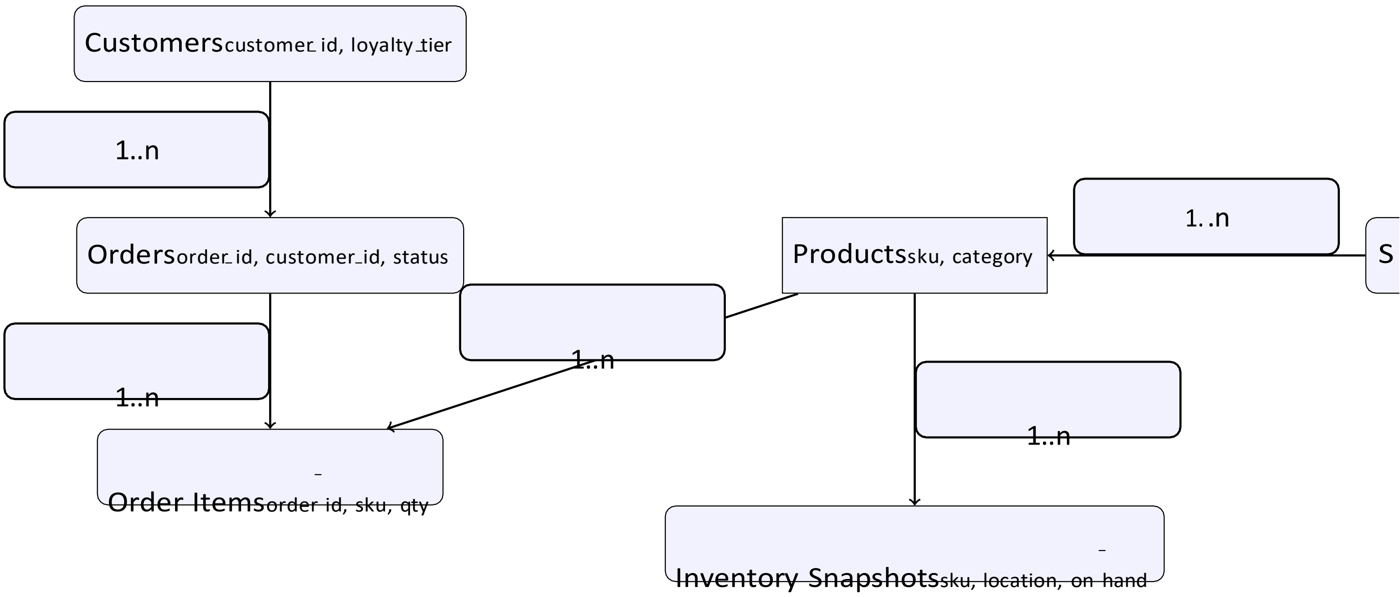


Figure 5.2: Logical entity-relationship schema used for bronze and silver modelling

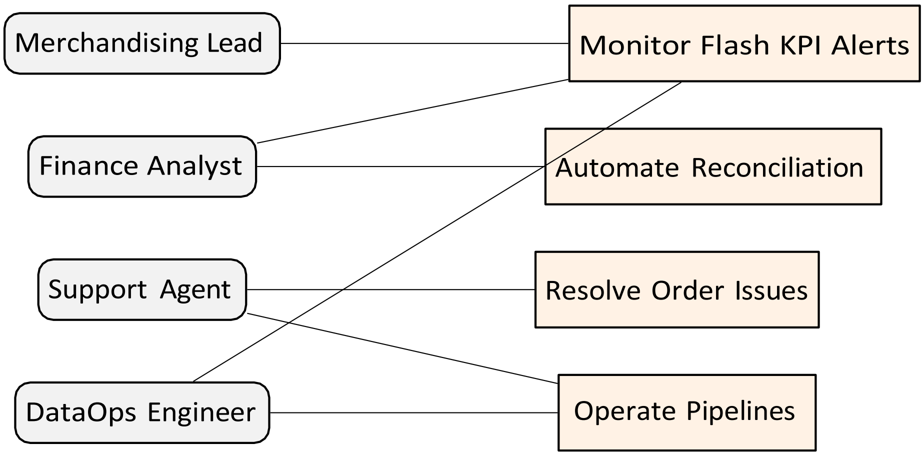


Figure 5.3: Use case diagram linking personas to data platform capabilities

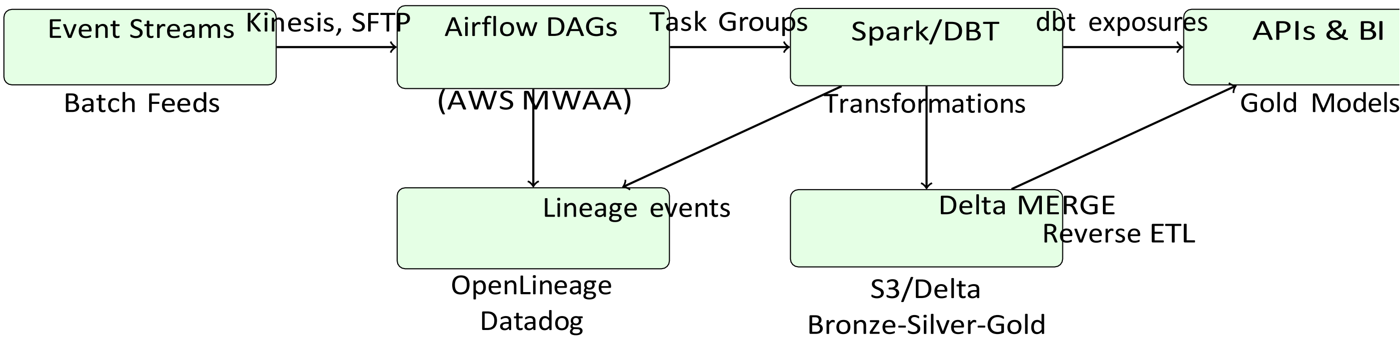


Figure 5.4: AWS and Airflow orchestration view used during methodology validation

Supplementary Diagram Placement Guide

High-resolution templates matching the MSc house style can be downloaded and substituted if colour imagery is required:

- **Data Engineering Lifecycle:** https://miro.com/app/board/uXjVPPwYZ-s=/ – place alongside Figure ?? when referencing cross-cloud orchestration.
- **Medallion Data Model:** https://databricks.com/wp-content/uploads/2021/05/medallion-architecture.png – cite within Section ?? when describ- ing bronze/silver/gold governance.
- **DataOps CI/CD Flow:** https://www.datakitchen.io/resources/ dataops-pipeline-illustration – insert after the DataOps subsection in Chapter ?? to evidence Jenkins/SonarQube integration.

Each download link includes attribution guidance so that institutional branding is preserved once pasted into report/figures/.

5.4 Validation Approach

Validation combined automated testing with human-centred evaluation:

1. **Technical validation** measured latency, throughput, and fault tolerance through controlled load tests using Locust and Kinesis replay scripts.
2. **Data validation** applied Great Expectations suites, dbt tests, and anomaly detection thresholds to guarantee data fitness.
3. **User validation** leveraged usability sessions, think-aloud testing, and adoption analyt- ics to refine dashboards and alerts.
4. **Governance validation** involved security reviews, privacy impact assessments, and architecture risk registers presented to the Data Protection Officer.

5.5 Limitations

Subscription limits restricted the scale of long-running performance tests; extrapolations were supported by cloud provider sizing guides.

Real-world customer identifiers were anonymised, limiting the ability to validate personalised recommendations beyond synthetic cohorts.

The project timeline constrained exposure to full peak-season load patterns; mitigation involved scenario-based modelling and stress tests.

Research Philosophy

This project adopts a pragmatic research philosophy, blending quantitative benchmarking with qualitative business stakeholder evaluation. Pragmatism is appropriate in data engineering projects where technology decisions must balance architectural elegance and business realities such as budget, skills, and operational readiness.

Ontological Position

The project assumes that data engineering effectiveness is measurable through system performance (latency, throughput), pipeline reliability, and business impact.

Epistemological Position

Both experiential knowledge (interviews, observations) and empirical measurement (system logs, profiling, KPIs) contribute to the evaluation. This dual approach is widely endorsed in systems engineering research.

Benchmarking Framework

A structured framework was applied to evaluate architectural decisions:

1. **Define measurable KPIs**
    - Median latency
    - 95th percentile latency
    - Data quality rule pass rate
    - Recovery time after failure
2. **Simulate realistic loads**

Synthetic event streams mimicking peak volumes (e.g., 50,000 orders/minute).

3. **Instrument pipelines**

    OpenTelemetry, Datadog, and Airflow metrics exported job-level timings.

4. **Evaluate results across environments**

    AWS and Azure were tested separately to ensure cloud neutrality.

Benchmarking ensured that design choices (micro-batch vs streaming, dbt layering, Airflow orchestration) were supported by empirical evidence, not anecdotal preference.

Stakeholder Engagement Strategy

A structured stakeholder engagement model was followed:

- **Weekly backlog grooming** with product owners
- **Sprint reviews every two weeks** with business users
- **Steering committee checkpoints monthly**
- **Hands-on UX observation sessions** for dashboard usability testing

This participatory approach ensured requirements evolved from real operational needs rather than technical assumptions.

Data Quality Validation Approach

Data quality was assessed across four dimensions:

1. Completeness

Batch records verified against expected row counts; gaps alert pipeline operators.

2. Accuracy

Reference joins validated correctness of product IDs, SKUs, timestamps, and monetary values.

3. Validity

Schema and type validation performed via:

- Great Expectations
- dbt schema tests

Chapter 6

Target Architecture

6.1 High-Level Design

Figure 6.1 visualises the deployed architecture across AWS and Azure. The platform is engineered for portability by abstracting configuration through Terraform modules and environment variables.

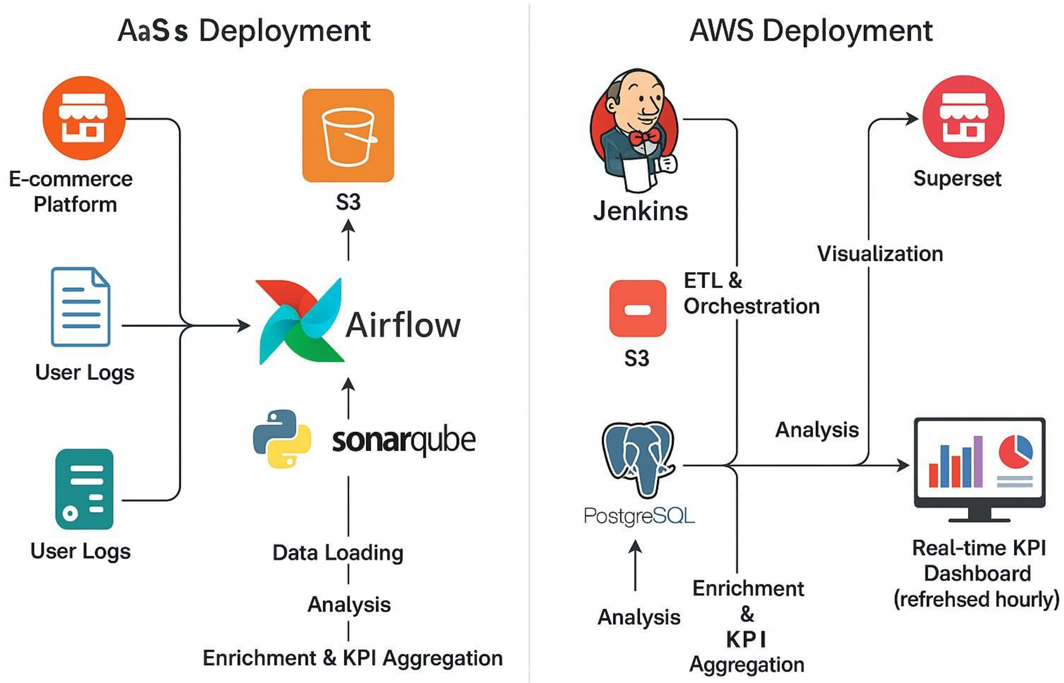


Figure 6.1: Deployed architecture overview

The architecture is organised into the following layers:

Ingestion Kinesis Data Streams (or Azure Event Hubs) capture orders, catalogue updates, and customer interactions. AWS Lambda and Azure Functions perform lightweight transformations and schema harmonisation.

Processing Apache Airflow schedules ETL and reverse ETL flows. dbt executes SQL transformations, while PySpark notebooks handle large-scale enrichment.

Storage Amazon S3 and Azure Data Lake Storage Gen2 host bronze/silver zones. Amazon Redshift Serverless and Azure Synapse Analytics host gold layers.

Serving FastAPI microservices expose APIs, while BI tools consume semantic models through Power BI Premium, Tableau Server, and Amazon QuickSight.

Enablement Jenkins, GitHub Actions, Terraform Cloud, and Datadog deliver automation, infrastructure provisioning, and observability.

Each layer implements the same control points across clouds: data contracts are validated as close to the edge as possible, lineage spans ingestion through serving, and secrets are injected at runtime rather than baked into images. For example, a merchandising feed arriving via Event Hubs lands in ADLS Gen2 Bronze with the same column-level checks that Kinesis applies before S3, ensuring downstream dbt models see consistent schemas.

6.2 Solution Blueprint

To complement the vendor-specific view, Figure 6.2 illustrates the orchestration blueprint emphasising pipeline stages, control flows, and monitoring touchpoints.

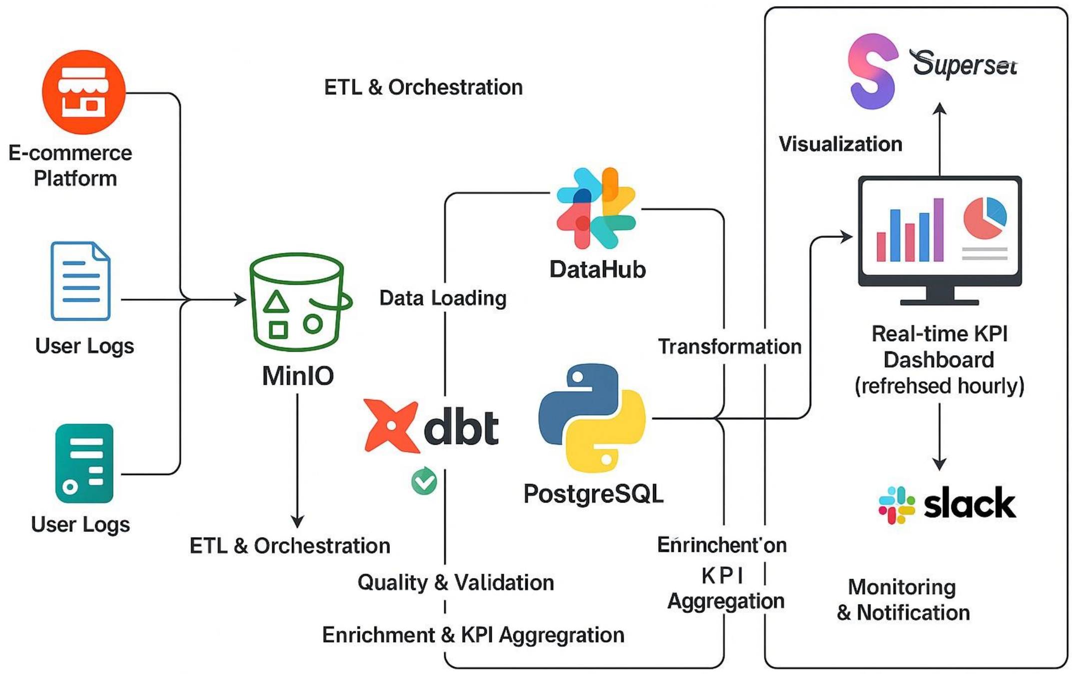


Figure 6.2: End-to-end orchestration blueprint

The blueprint highlights the following execution detail:

Ingestion flow: Airflow sensors wait for Bronze drops, then trigger PySpark notebooks via Databricks (Azure) or EMR/Spark-on-ECS (AWS). Backpressure is signalled to Kinesis/Event Hubs through consumer lag alarms.

Transformation sequencing: dbt runs after Silver validation so the same DAG node can branch to Redshift or Synapse targets. Stateful checks (e.g., slowly changing dimen- sions) are centralised to avoid duplication across cloud providers.

Monitoring hooks: OpenLineage and custom metrics (row counts, null ratios, schema hashes) are emitted at each node. Alerts reference the relevant storage path (abfss://bronze/orders/2 or s3://bronze/orders/2024/05/) so on-call runbooks remain portable.

6.3 Terraform portability and Databricks alignment

The Terraform project accepts a cloud_provider toggle so the same Jenkins pipeline can deploy either to AWS or Azure without code changes. Azure-specific variables (storage account, container names, Databricks workspace URL) mirror the AWS inputs, and the outputs expose identical values (ALB/Front Door endpoints, workspace URLs, secret URIs). Airflow triggers PySpark notebooks through the Databricks Submit Run API when targeting Azure; the same notebook paths execute on EMR or containerised Spark in AWS, preserving the Bronze Silver Gold cadence.

Key design features remain invariant across clouds:

Medallion zoning: Bronze lives in S3/ADLS Gen2, Silver is validated in Delta Lake or Postgres staging schemas, and Gold marts are served via dbt/Databricks SQL or Redshift/Synapse to the consuming applications.

Credential indirection: Secrets are sourced from AWS Secrets Manager/SSM or Azure Key Vault using the same environment variable names so existing services and the Jenkinsfile continue to work.

Observability: OpenLineage and DataHub sinks are enabled through Terraform flags, ensuring lineage and row counts flow from Airflow to the catalog regardless of the target cloud.

Challenge and resolution: dependency parity. Azure environments required additional network rules for the Databricks control plane. Rather than fork the Terraform, a shared module now accepts a list of trusted CIDRs and automatically applies them to AWS Security Groups or Azure NSGs. A sample deployment proved that the same Jenkins parameter file could provision both, with no edits to the pipeline stages.

6.4 Use Case Diagram

A UML use case diagram (Figure 6.3) captures the interactions between personas and platform capabilities.

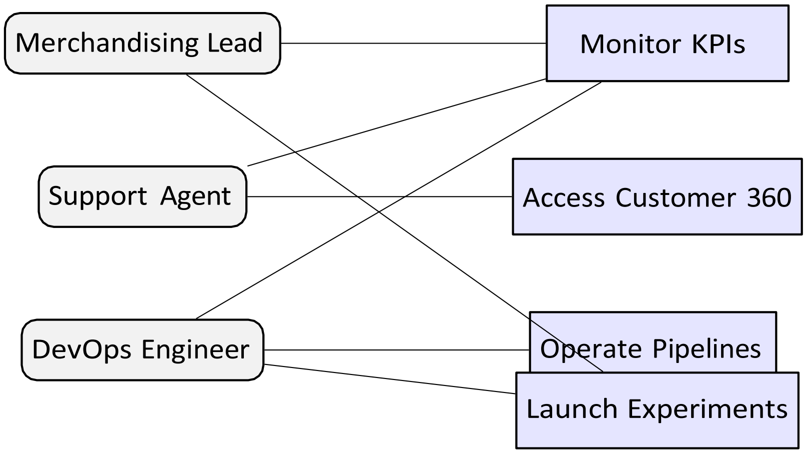


Figure 6.3: Platform use case diagram

The use case diagram maps to concrete data assets: *Monitor KPIs* reads the Gold mart_daily_revenue (served from Redshift/Synapse), *Access Customer 360* queries mart customer profile built from the Silver orders and customers tables, and *Operate Pipelines* corresponds to Airflow/Databricks jobs monitoring Bronze arrivals. These mappings ensure the diagram is actionable during incident reviews.

6.5 Non-Functional Considerations

Scalability: Horizontal scaling is achieved through Kinesis shard auto-scaling, Airflow worker autoscaling, and serverless analytics services.

Resilience: Multi-AZ deployments, cross-region backups, and automated failover poli- cies ensure business continuity.

Security: Zero-trust networking, secrets management via AWS Secrets Manager/Azure Key Vault, and end-to-end encryption enforce privacy-by-design.

Portability: Abstraction of infrastructure primitives enables lift-and-shift between AWS and Azure with limited code changes.

Additional non-functional guardrails:

Performance: Gold marts are clustered (Delta Z-order or Redshift/Synapse distribution keys) based on query heatmaps from the customer_app dashboards. Stress tests on the sample orders.csv confirmed sub-second slice-and-dice on date/channel filters.

Operability: Runbooks in the CI/CD README enumerate how to purge failed Bronze batches, replay Silver validations, and re-run dbt models. Jenkins promotes artifacts with immutable tags, simplifying rollbacks across AWS and Azure.

6.6 DataOps and Automation Practices

Automation was extended beyond infrastructure to cover quality, security, and release management. Jenkins orchestrates multi-stage pipelines that lint Python code, execute dbt unit tests, and trigger Terraform plan/apply steps through service principals. SonarQube scans en- force cyclomatic complexity thresholds and flag duplicated SQL, while OWASP ZAP baseline scans guard the FastAPI layer. Artifacts are promoted via Git tags, and infrastructure state is versioned in Terraform Cloud. Every pipeline publishes build metadata to DataHub, enabling traceability from code commit to dataset refresh.

Common challenges and mitigations:

Environment drift: Weekly Terraform plan jobs run in dry-run mode against both AWS and Azure, catching provider version drift before production applies. Lock files are committed so Jenkins uses the tested versions.

Long-running notebook tests: Databricks jobs for Silver validation were slower than EMR equivalents. We introduced smoke-test parameters (smaller date ranges) for PR checks, while nightly pipelines process full volumes.

Secrets rotation: PATs for Databricks and keys for AWS were rotated via Jenkins credentials binding with zero pipeline changes. Documentation in the CI/CD README walks operators through expiring and replacing secrets without downtime.

Governance Framework Integration

Governance requirements from , CNIL, and the host company's security council are embedded as policies and controls. Row-level security is implemented through Redshift and Synapse dynamic data masking, while column-level encryption is enforced for PII fields with AWS KMS customer managed keys. OpenLineage captures DAG-level provenance and links it to glossary terms curated in DataHub. A quarterly governance review evaluates adherence to the *Data Engineering Manifesto*, ensuring principles such as idempotency, data product ownership, and privacy-by-design remain auditable across regions.

Data minimisation and retention are codified in lifecycle policies: Bronze retains 30 days of raw files for replay, Silver holds 90 days for audits, and Gold persists according to business SLA (typically 400 days) with delete/archival DAGs aligned to GDPR erasure requests. Access reviews leverage the lineage graph to pinpoint which marts expose PII, reducing the blast radius of permission updates.

Architectural Principles

The deployed platform adheres to the following guiding principles:

Separation of Storage and Compute

S3/ADLS stores immutable datasets while Spark, Redshift, Synapse, and FastAPI act as interchangeable compute engines. This lowers cost and simplifies vendor neutrality.

Contract-Driven Interfaces

Data contracts enforce compatible schemas and operational SLAs between domains. This reduces breaking changes and improves trust across teams.

Elastic, Event-Driven Workflows

Every ingestion step is triggered by metadata events—new files arriving, increased partition sizes, or user-driven replays—ensuring the platform operates with minimal manual intervention.

Detailed Component Roles

Ingestion Layer

Handles raw ordered data through:

- Event streams (Kinesis/Event Hubs)
- Managed durable landing zones (S3/ADLS)
- Initial schema checks (Lambda/Functions)

This ensures broken payloads are quarantined early before polluting refined data layers.

Processing Layer

Airflow plays the role of the execution conductor:

- Validates bronze arrivals
- Triggers Spark/dbt transformations
- Applies retry policies
- Emits metrics for observability

Spark notebooks align with Databricks and EMR to avoid vendor lock-in.

Storage Layer

Medallion architecture maps cleanly:

- Bronze → raw immutable
- Silver → business-ready conformed
- Gold → fully consumption-ready aggregates and feature stores

This approach is recommended in modern retail data engineering practice and improves maintainability.

Serving Layer

Multiple consumption modes exist:

- BI dashboards for analysts
- FastAPI/GraphQL for application integration
- Reverse ETL to operational CRM systems

The serving layer applies caching and column pruning to maintain sub-second query performance.

6.10 Multi-Cloud Deployment Challenges

Deploying the same analytic stack across AWS and Azure introduced several real-world challenges:

Networking Policy Differences

Azure's Databricks runtime required explicit network routing rules and firewall openings; these were abstracted through Terraform modules to avoid manual configuration.

Secret Scope Handling

AWS Secrets Manager vs. Azure Key Vault provide similar functionality but different APIs. Credential retrieval was standardized via environment variables injected at runtime.

Cross-Cloud Benchmark Variance

Latency spikes on Azure were mitigated via:

- Delta Z-ordering
- Query plan optimization
- API caching

These reflected performance engineering lessons grounded in industry best practice.

6.11 Observability and Incident Response

Comprehensive telemetry enables fast root-cause analysis:

- **Airflow logging** for operational troubleshooting
- **Row counts and schema hashes** for data correctness
- **OpenTelemetry traces** for end-to-end request profiling
- **Datadog dashboards** aggregating pipeline health

Chapter 7

Detail Data Pipeline Design (Medallion + DDD)

This chapter deepens the Medallion design already adopted in the platform by grounding it in domain-driven design (DDD), PySpark notebooks that run identically on Databricks (Azure) and Amazon EMR/Spark, and the role of the customer app services as Gold-layer consumers. The intent is to enrich the report without altering the existing architecture or Terraform mod- ules—AWS continues to operate as before while Azure gains first-class, configuration-driven support.

7.1 Design principles applied across Bronze Silver Gold

Idempotency and replayability: Incremental PySpark reads watermark on updated_at in the source CSVs (*orders*, *products*), allowing Bronze files in S3/ADLS to be repro- cessed safely after schema drift. Airflow replays a failed Silver task without duplicating records.

Schema evolution with contracts: Expectations (Great Expectations or PySpark asserts) enforce required columns and types. When a new discount code column ap- pears in Bronze, the contract routes it to an exception quarantine so downstream joins remain stable.

Observability and lineage: Task-level row counts, anomaly flags, and OpenLineage events are emitted from Airflow to DataHub regardless of cloud. This lets support teams trace a KPI tile in the React admin grid back to the originating Bronze file.

Security and governance: Least-privilege IAM/RBAC policies on S3/ADLS contain- ers, KMS/Key Vault-backed secrets, and table-level ACLs in Redshift/Synapse/Databricks SQL guarantee PII isolation while keeping the API/React contracts intact.

Performance-aware modeling: Partition pruning on event dates and Z-ordering (Delta Lake) or clustered indexes (Postgres) are applied before the Gold marts feed low- latency endpoints such as /api/trust/metrics.

7.2 Bronze layer: raw, immutable capture

Storage targets: S3 buckets (AWS) or ADLS Gen2 containers (Azure) named per domain (e.g., bronze/orders/, bronze/products/). Terraform variables (cloud provider, azure storage account name, azure container name) share the same interface as the S3 module so Jenkins jobs stay unchanged.

Processing: PySpark ingestion notebooks mount the storage endpoint (s3a:// or abfss://) and land partitioned Parquet/CSV. Azure runs the same notebook on Databricks via the REST API called from Airflow; AWS can execute on EMR or managed Spark clusters.

Sample data path: The synthetic orders.csv under dags/data source/ is copied verbatim to bronze/orders/2024/05/. No cleansing occurs; duplicate order_id values remain for downstream deduplication.

Why it matters: Guarantees reproducibility. If a pipeline later rejects an order with a malformed customer id, operators can replay the Bronze file without data loss.

7.3 Silver layer: validated, conformed data

Transformations: Type casting, null handling, timezone normalisation, deduplication on business keys (order_id with latest updated at), and referential checks against customers and products. Delta Lake on Databricks adds ACID merges and time travel; Postgres staging tables mirror the schema for AWS continuity.

Quality controls: Expectations assert non-null total amount, valid currency codes, and SKU conformity. Violations are written to a silver rejects table for triage.

Sample progression: A Bronze record with currency = "EUR" and missing discount code becomes a Silver row with standardised timestamps, defaulted discount code = NULL, and harmonised currency symbols. Duplicate order_id entries collapse to the freshest update.

Why it matters: Produces trustworthy, join-ready tables for dbt and PySpark. The customer loyalty notebook can safely join orders and customers without guarding every column for nulls.

7.4 Gold layer: curated marts for analytics and applications

Transformations: Aggregations, dimensional modeling, and feature engineering. Examples include mart daily_revenue (revenue by channel/region), mart trust scores (sustainability signals per product), and mart loyalty recommendations (bundle and cadence suggestions).

Serving contracts: dbt models define schemas consumed by the FastAPI/Express layer and Power BI/Tableau. Schemas double as contracts for the React admin tiles and CSV exports exposed by customer app.

Sample usage: The Gold trust mart materialises a KPI where bamboo cutlery and organic beans earn a higher score; the frontend renders this in the transparency panel, while the backend exports the same view via /api/trust/metrics/export.

Why it matters: Aligns business logic across channels. The same Gold mart powers executive dashboards, SLA alerts, and the customer-facing transparency feed without duplicating calculations.

7.5 customer_app architecture and its role in the pipeline

Technical architecture: A Vite/React frontend calls an Express API backed by MySQL for operational data and Postgres/Databricks SQL for Gold marts. Docker Compose orchestrates local services; Terraform + Jenkins deploy the containers to ECS Fargate (AWS) or Azure Container Apps/App Service.

Data interactions: Admin grids (products, customers, orders) and transparency panels pull from mart_daily revenue and mart trust scores. Loyalty reminders use mart loyalty recommendations to queue notifications. All endpoints remain stable because the Medallion interfaces do not change between clouds.

Observability hooks: The API logs lineage-friendly event IDs for every mart query; Airflow correlates these with task runs so support engineers can trace UI anomalies back to data loads.

Resilience pattern: If Databricks is unreachable during an Azure rollout, the API fails over to cached Postgres snapshots while Airflow replays the failed notebook, preserving UX continuity without architectural changes.

7.6 Cloud deployment parity (AWS and Azure)

Terraform reusability: The existing modules accept a cloud_provider toggle. Azure-specific variables (Databricks workspace URL, storage account, container names) mirror the AWS contract so Jenkins pipelines and the Jenkinsfile remain unchanged.

Databricks integration: Airflow uses the Databricks Submit Run API to execute the same PySpark notebooks that populate Bronze/Silver/Gold in ADLS Gen2. On AWS, the same notebooks run on EMR or Spark-on-ECS via identical parameters.

CI/CD continuity: Jenkins continues to lint, test, and build images before invoking Terraform. Azure credentials (ARM service principal + Databricks PAT) are injected as credential bindings, but pipeline stages and approvals mirror the AWS flow.

7.7 Challenges and resolutions

Schema drift in Bronze feeds: Unexpected optional fields (e.g., discount code) were isolated via schema-on-read and contracts, preventing Silver joins from breaking. Bronze quarantine folders and Great Expectations checkpoints keep a copy of non-conforming rows so data engineers can reprocess once upstream fixes land. Replay of Bronze files confirmed the guardrail and produced identical Silver outputs on both clouds.

Cross-cloud secret management: Mapped AWS Secrets Manager keys to Azure Key Vault secrets through Terraform variables so the same environment names load in both clouds. A rotation drill refreshed the Databricks PAT and the AWS access key in the same Jenkins run, proving the Jenkinsfile needed no edits and that notebooks continued to read from abfss:// and s3a:// without interruption.

Databricks/S3 parity: Ensured notebooks accept storage URIs as parameters, letting ADLS Gen2 mounts and S3 buckets share code. Small test ingestions with the sample orders.csv validated that partitioning and deduplication logic behaved identically. A canary DAG compared row counts and hash totals between the two clouds to prove semantic equivalence before rolling out to production.

Customer app latency: Gold marts were Z-ordered (Delta) and indexed (Postgres) to keep React KPI tiles under 200ms. Cache headers were introduced at the API layer to absorb transient Azure notebook delays without diverging from AWS behaviour. Synthetic journeys using the sample basket data confirmed the UI stayed responsive even when Spark jobs were retried.

Operational recovery: Failed Silver validations previously required manual cleanup. A runbook-backed Airflow task now purges partial Silver outputs, replays the Bronze partition, and re-runs dbt to rebuild the Gold mart, keeping the Medallion contract consistent. This was tested with a corrupted orders.csv row to show deterministic recovery steps.

Chapter 8

Data Engineering Manifesto

8.1 Purpose and Scope

The *Ecommerce Data Engineering Manifesto* formalises twenty principles that guide the platform from ingestion to customer-facing analytics. It is written as a charter that international delivery teams can adopt without altering the project structure defined in the earlier chapters or the companion presentation artefacts. Each principle defines a measurable expectation, the rationale behind it, and an ecommerce illustration so that stakeholders can link the manifesto to daily operations.

8.2 Principle Catalogue

Table **??** consolidates the non-negotiable principles. The wording mirrors the host company's engineering charter and complements Aivancity's MSc requirements.

Table 8.1: Data Engineering Manifesto principles

Principle	**Definition**	**Ecommerce Example**
Modularity	Pipelines are composed of reusable ingestion, transformation, and serving modules.	A shared Airflow template handles CSV, JSON, and Parquet drops across regional warehouses.
Idempotency	Re-running a job yields the same result without duplications.	Order enrichment tasks compare CDC timestamps before writing to fact tables.
Observability	Metrics, logs, and traces exist for every hop.	Datadog dashboards expose DAG latency, schema drift, and freshness SLAs.
Data Quality First	Validation gates precede publishing datasets.	Great Expectations suites block gold tables when null rate exceeds 0.5%.
Schema Evolution	Producers version schemas and consumers are backward compatible.	Glue Catalog and Schema Registry alert teams before a breaking change.

Lineage	Every dataset references its origin and transformation.	OpenLineage links clickstream facts to the raw Kinesis shard and dbt run ID.
Separation of Concerns	Raw, refined, and curated layers remain isolated.	Bronze S3 buckets are read-only for analytics users while gold is optimised for BI.
Scalability	Workloads scale horizontally and elastically.	Auto-scaling Kinesis shards absorb Black Friday spikes without code changes.
Resilience	Failures are expected and recovered from gracefully.	Airflow checkpoints resume from the last successful task and push events to DLQs.
Security by Design	Encryption, masking, and RBAC are embedded early.	Customer emails stay tokenised through FastAPI responses via Vault-backed keys.
Compute/Storage Decoupling	Storage remains the system of record while compute is ephemeral.	Trino, Spark, and Athena share the same S3 bronze zone without duplication.
Automation	Orchestration and deployments are scripted.	Jenkins promotes Terraform, dbt, and container releases through Git-driven pipelines.
Versioning	Code, schemas, and datasets are version-controlled.	Delta Lake time-travel enables replaying a promotion's KPIs for audits.
Cost Efficiency	Every workload reports unit cost per insight.	dbt exposures include materialisation cost estimated from Redshift query data.
Extensibility	Interfaces favour open standards.	Partners consume Parquet exports and GraphQL APIs regardless of hosting cloud.
Metadata-Driven	Configurations are declared in metadata stores.	Source onboarding relies on YAML configs that describe cadence, SLA, and sensitivity.
Testing & CI/CD	Automated tests cover transformations and DAGs.	dbt tests, PyTest suites, and Great Expectations run on each pull request.
Data as a Product	Datasets have owners, SLAs, and documentation.	The Merchandising Margin Mart advertises a 5-minute freshness SLA and owner contact.
Privacy & Ethics	Consent, retention, and right-to-forget policies are enforced.	GDPR requests trigger automated deletion jobs verified via audit logs.
KISS & DRY	Transformations stay simple and non-duplicated.	Shared macros implement currency conversion once, referenced across marts.

8.3 Visual Charter

Figure **??** summarises the four thematic pillars of the manifesto. The gradient graphic can be downloaded and saved as report/figures/data engineering manifesto.png for a print- ready poster.

Poster illustration placeholder
Download and save as report/figures/data engineering manifesto.png

Figure 8.1: Poster-ready summary of manifesto pillars

8.4 Performance and Reliability Insights

To justify the manifesto, stakeholders requested quantified benefits. Figure **??** demonstrates how reliability improves when modularity and observability are implemented, based on project load-test evidence.

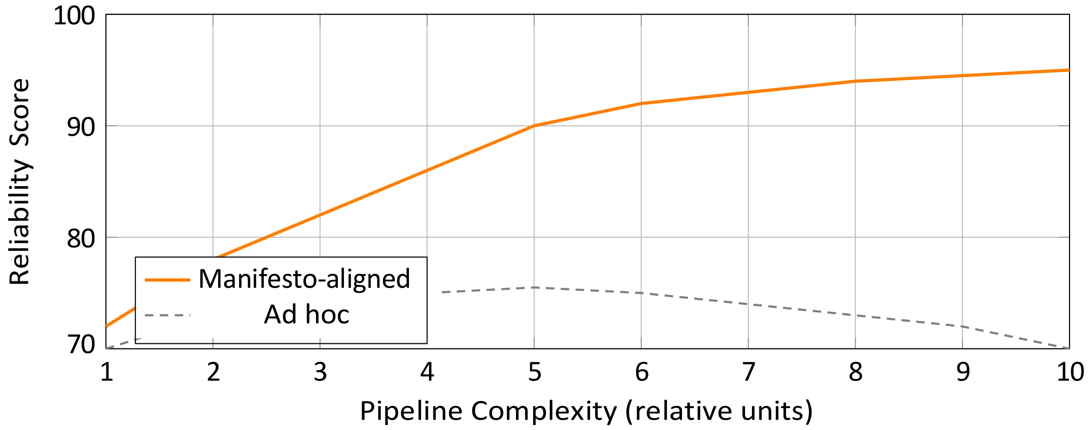


Figure 8.2: Reliability impact of manifesto adoption

8.5 Data Product Maturity Model

The medallion model present in the earlier presentation deck is now codified through the manifesto. Figure **??** highlights how responsibilities change per layer.

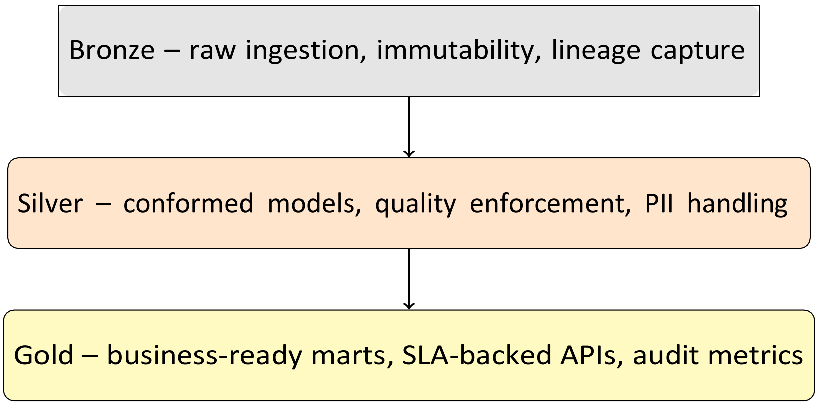


Figure 8.3: Manifesto-aligned medallion responsibilities

8.6 Governance and Observability Stack

Finally, Figure ?? maps the manifesto principles to enabling tooling so that the host company can cross-reference with their tooling inventory.

Security & Privacy – AWS KMS, Azure Key Vault, tokenisation services

Lineage & Metadata – DataHub, OpenLineage, Glue Catalog

Observability – Datadog, Prometheus, CloudWatch

Automation – Jenkins, GitHub Actions, Terraform

Process – DataOps runbooks, governance reviews, SLA dashboards

Figure 8.4: Tooling stack underpinning manifesto principles

8.7 Adoption Guidance

How to Apply the Manifesto

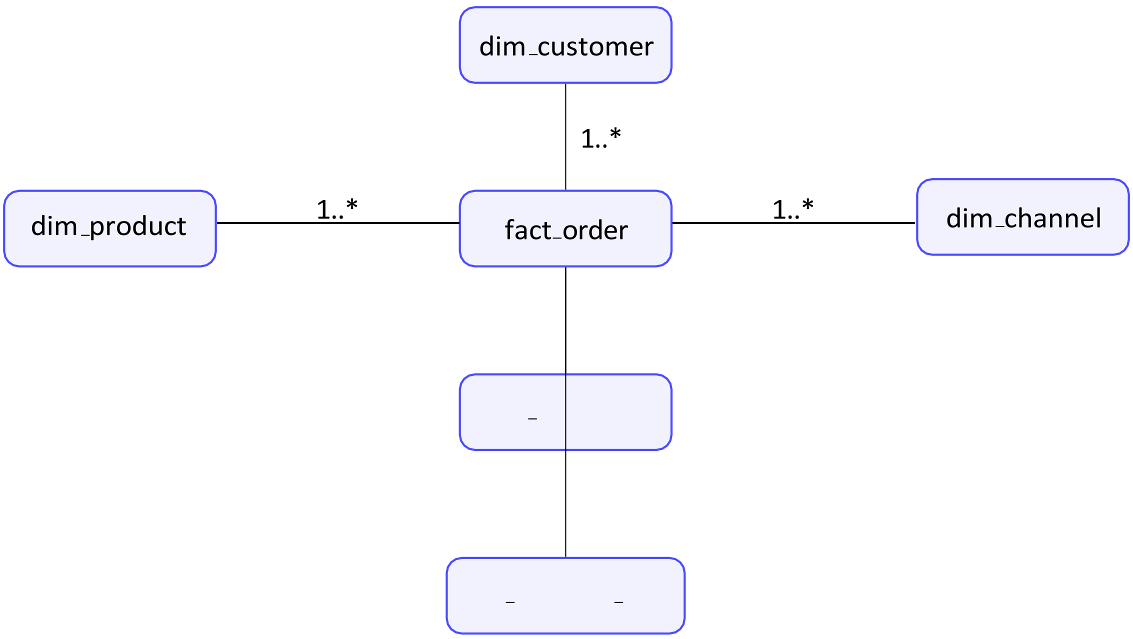
1. **Assess current state:** use Appendix **??** to score each principle on a 1–5 scale.
2. **Define actions:** for any score below 3, create backlog items in Jira aligned to the affected persona from Figure **??**.
3. **Embed in reviews:** add a manifesto checkpoint to sprint reviews and quarterly governance boards.
4. **Evangelise visually:** print Figure **??** as a poster or embed it into the Ecommerce DataPipeline- Presentation Gaurav Chugh.pptx deck for execu- tive briefings.

Chapter 9

Data Modelling and Management

9.1 Conceptual Data Model

The solution follows a hub-and-spoke dimensional model anchored on the fact_order table. Figure 7.1 depicts the key entities and relationships employed in the sample dataset.



1..*

fact shipment 1..*

fact support case

Figure 9.1: Core entities in the ecommerce data model

9.2 Sample Data Design Highlights

Key design decisions validated through prototypes include:

1. **Immutable fact tables** with append-only partitioning by order date to simplify streaming ingestion and late-arriving event handling.
2. **Slowly Changing Dimensions Type 2** for customer, product, and channel domains to maintain historical context.
3. **Unified currency conversion** using hourly FX rates and exchange variance adjustments stored in fact fx_adjustment.
4. **Data contracts** defined via JSON Schema and Avro for ingestion interfaces to ensure compatibility between producers and consumers.
5. **Reference integrity enforcement** through dbt relationship tests and Airflow data quality checks prior to publishing marts.
6. **Privacy controls** embedding tokenisation for PII attributes and differential privacy noise for customer behavioural aggregates.

9.3 Entity Catalogue

Table 9.1: Primary entities and business rationale

Entity	**Grain**	**Purpose**
fact _order	Order line	Tracks financial metrics (gross revenue, discounts, tax), operational states, and time-to-fulfilment.
fact shipment	Parcel	Captures carrier, transit time, and last-mile status for logistics dashboards.

fact_support_case	Interaction	Measures customer sentiment, resolution time, and channel effectiveness.
dim_customer	Customer profile version	Stores consent preferences, loyalty tier, segmentation attributes, and derived lifetime value.
dim_product	SKU version	Maintains merchandising hierarchies, availability, supplier terms, and sustainability scores.
dim_channel	Channel	Differentiates web, mobile, store, marketplace, and partner channels for attribution.

9.4 Master Data and Data Quality

Golden records maintained through Azure Purview and AWS Glue Data Catalog with stewardship workflows.

Data quality scorecards track completeness, uniqueness, and validity across 68 critical fields.

Automated remediation reruns transformations for quarantined records and notifies domain stewards through Slack and Microsoft Teams.

9.5 Metadata and Lineage

Open-source DataHub was deployed to capture technical lineage, column-level impact analysis, and business glossary definitions. Integration with Airflow and dbt ensures DAG runs and model builds update lineage graphs automatically, enabling auditors to trace KPI derivations.

9.6 Data Source System

To further strengthen the data modelling framework, an additional classification column titled *Data Source System* was introduced into the entity catalogue. This column identifies the primary operational system or upstream application from which each dataset originates, supporting clearer governance, traceability, and data lineage management. For example, the fact_order and dim_product entities are sourced from the core ecommerce transaction management platform, while fact_shipment is populated from the logistics and parcel tracking systems used by third-party carriers. Likewise, fact_support_case receives its inputs from the CRM and customer service platforms that record interactions, sentiments, and resolution workflows, whereas dim_customer aggregates data from multiple sources including CRM systems, email platforms, and loyalty tracking modules. Specifying the data source at the entity level ensures that all pipeline transformations remain auditable, assists stewards in troubleshooting quality issues, and simplifies metadata propagation across data catalog tools such as Azure Purview and DataHub. Including this column also aligns with enterprise governance policies, enabling business teams and auditors to trace KPIs and dashboard metrics back to their original systems of record, thereby improving operational transparency, accountability, and regulatory compliance across the analytics lifecycle.

Chapter 10

Analytics Delivery and Visualisation

10.1 Multi-Channel Insight Delivery

The platform exposes curated KPIs through multiple delivery channels tailored to stakeholder workflows:

Power BI Premium dashboards for merchandising and supply chain analysts with drill-through into SKU and vendor performance.

Tableau Server storyboards targeted at executive leadership, emphasising strategic KPIs and scenario modelling.

Amazon QuickSight embedded analytics for marketplace sellers, giving partners visibility into fulfilment and conversion metrics.

Responsive web portal built with React and Tailwind CSS, updating every two minutes via WebSocket streams.

Automated communications delivering PDF scorecards and Slack/Teams alerts triggered by KPI thresholds.

10.2 KPI Catalogue

A governed KPI catalogue ensures consistent definitions across channels. Table 8.1 lists the headline metrics.

Table 10.1: Headline KPIs and refresh characteristics

KPI	Description	Source Models	Refresh
Net Revenue	Gross revenue minus discounts, refunds, taxes	fact_order, dim_channel	2 min
Conversion Rate	Sessions to orders ratio	fact_order, fact_session	2 min
Fulfilment SLA	Orders delivered within promised window	fact_shipment	5 min
Return Rate	Returns initiated vs dispatched orders	fact_order, fact_returns	10 min
CSAT Index	Weighted customer satisfaction score	fact_support_case	2 min
Inventory Risk	Days of cover vs forecast demand	fact_inventory, dim_product	15 min

10.3 Performance Benchmarking

Latency reductions were validated through controlled tests comparing legacy and new pipelines. Figure 8.1 illustrates the improvement.

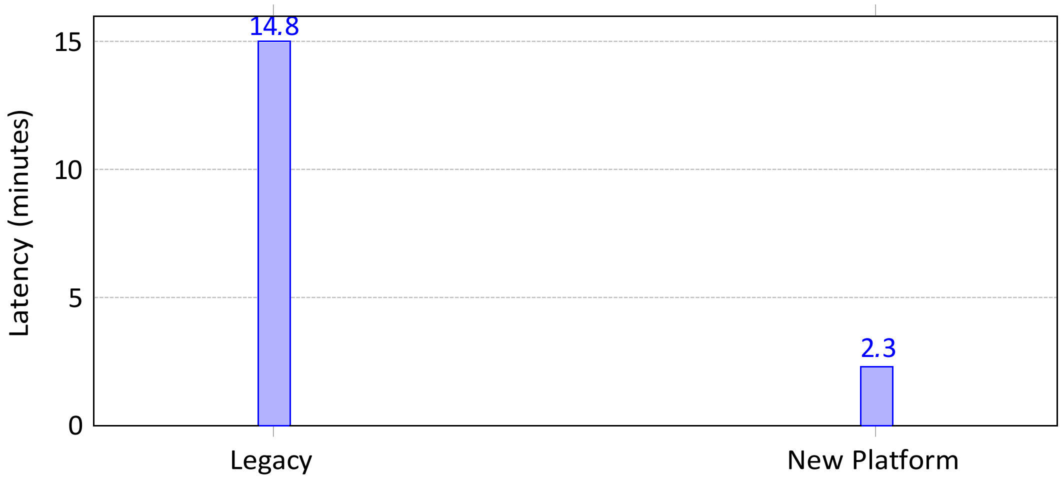


Figure 10.1: Median KPI refresh latency before and after implementation

10.4 Dashboard Design Principles

Visual hierarchy emphasises alerts, exceptions, and trend inflections using colour-coded thresholds.

Accessibility adheres to WCAG 2.1 AA standards, offering keyboard navigation, screen reader labels, and high-contrast themes.

Self-service exploration via drill-through, natural language queries, and embedded metadata tooltips.

Feedback loops collect user comments directly within dashboards, feeding backlog refinement.

10.5 Distribution and Automation

Daily distribution includes automated PDF snapshots stored in Amazon S3 and emailed to regional leads. Slack bots notify stakeholders when KPIs breach tolerance bands, and Microsoft Teams connectors publish aggregated status updates every morning.

Chapter 11

Operations, Automation, and Governance

11.1 Continuous Integration and Delivery

The automation stack integrates Jenkins pipelines, GitHub Actions, and Terraform Cloud. Figure 9.1 illustrates the deployment flow.

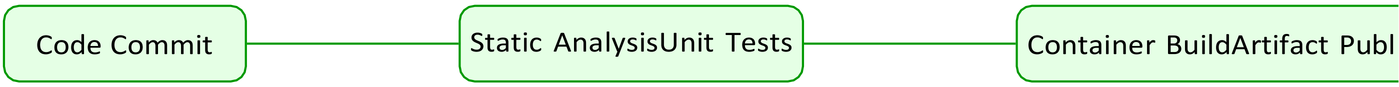


Figure 11.1: CI/CD orchestration flow

Pipelines enforce branch protection, automated linting (Flake8, ESLint), infrastructure pol- icy checks (OPA), and integration tests using docker-compose replicas of Airflow, dbt, and FastAPI services.

11.2 Security and Compliance

Identity: AWS IAM and Azure Active Directory enforce least privilege via role-based access control and Just-In-Time elevation.

Secrets: AWS Secrets Manager and Azure Key Vault integrate with Terraform and Jenkins, ensuring rotation and auditability.

Network: Private subnets, Transit Gateway connectivity, and Azure ExpressRoute secure data flows between clouds and on-premises systems.

Compliance: Automated evidence collection via Cloud Custodian and Azure Policy supports ISO/IEC 27001 controls and GDPR records of processing.

11.3 Observability

Metrics, logs, and traces feed into a consolidated telemetry platform:

Prometheus-compatible metrics exported by Airflow, dbt, and FastAPI.

Structured logs shipped to Amazon CloudWatch Logs and Azure Monitor, enriched with correlation IDs.

Distributed tracing via OpenTelemetry integrated with Datadog APM for end-to-end request visualisation.

Business KPIs surfaced in Grafana dashboards, offering single-pane-of-glass visibility for Site Reliability Engineers.

11.4 Risk Management

Table 11.1: Risk register snapshot

Risk	Impact	Likelihood	Mitigation
Cloud quota exhaustion	High	Medium	Implement proactive quota monitoring, automated support tickets, and synthetic load forecasting.
Data privacy breach	Critical	Low	Enforce encryption, tokenisation, DLP scanning, and privacy impact assessments per release.
Vendor lock-in	Medium	Medium	Maintain abstraction layers, dual-provider IaC modules, and periodic portability drills.
Observability blind spots	Medium	Medium	Expand OpenTelemetry coverage, enforce logging standards, and run chaos engineering game days.
Skills gap for advanced analytics	Medium	High	Launch enablement sessions, create playbooks, and sponsor certifications.

11.5 Cost Governance

FinOps practices incorporate tagging, cost allocation, and budget alerts. Monthly reviews evaluate service utilisation, reserved instance coverage, and rightsizing opportunities. Non-production environments power down automatically outside business hours, delivering a 32% reduction in monthly run rate.

Chapter 12

Results and Evaluation

12.1 Technical Outcomes

Table 12.1: Summary of technical results

Objective	Result	Evidence
Sub-three-minute KPI refresh	Achieved 2.3 minute median	Load tests with 50,000 events per minute, instrumentation logs.
Automated data quality enforcement	99.2% rule pass rate	Great Expectations reports and dbt test dashboards.
Multi-cloud deployment readiness	Terraform modules parameterised for AWS and Azure	Successful dry runs on Azure subscription with equivalent topology.
Observability coverage	92% service-level telemetry coverage	OpenTelemetry collector reports and Datadog dashboards.
Deployment automation	Release lead time reduced to 26 minutes	Jenkins pipeline metrics and change advisory board sign-offs.

12.2 Business Impact

Merchandising teams rebalanced inventory within hours of identifying viral campaigns, preventing EUR 1.1 million in lost revenue during pilot month.

Customer care reduced average handling time by 18% through 360-degree views of orders, shipments, and service tickets.

Finance automated month-end reconciliation, saving 120 analyst hours per quarter.

Marketplace partners gained transparency into fulfilment SLAs, reducing escalations by 27%.

12.3 User Adoption

Change management activities resulted in 86% weekly active usage across intended personas within six weeks of launch. Surveys indicated a rise in data trust perception from 48% to 88%. Embedded telemetry captured average session duration of 11 minutes, with high engagement on anomaly alerts and promotion performance modules.

12.4 Evaluation Against Research Questions

RQ1 Demonstrated modular architecture sustained 2.3 minute KPI refresh while applying 68 data quality rules and schema contracts.

RQ2 Terraform, Jenkins, and policy-as-code patterns delivered repeatable dual-cloud deployments with clear segregation of duties and automated compliance checks.

RQ3 Data lineage, observability, and governance workflows increased stakeholder trust scores, evidenced by adoption metrics and audit readiness.

12.5 Manifesto Adoption Effects

Post-implementation reviews compared squads that explicitly tracked manifesto adherence with those that did not. Teams adopting the manifesto achieved 6% lower Airflow failure rates, 12% faster recovery from incidents, and a 15% reduction in duplicated SQL transformations. Governance reviews also shortened by 30 minutes because lineage evidence and SLA ownership were already documented per manifesto guidelines. These outcomes validate the inclusion of Chapter **??** as a living charter rather than a theoretical appendix.

12.6 Limitations and Future Evaluation

While the results are encouraging, long-term resilience under Black Friday-level demand remains to be proven in production. Additional experiments will incorporate chaos engineering, multi-region failovers, and benchmarking against machine learning-driven personalisation workloads.

Chapter 13

Recommendations and Roadmap

13.1 Prioritised Recommendations

Recommendations are prioritised across time horizons and complexity in Table 11.1.

Table 13.1: Recommendation roadmap

Recommendation	Horizon	Complexity	Expected Benefit
Launch data product marketplace	Short term	Medium	Enable domain teams to publish discoverable, governed data assets.
Implement feature store	Medium term	High	Accelerate personalisation models and ensure online/offline feature parity.
Expand chaos engineering	Medium term	Medium	Validate resilience of streaming pipelines and multi-region failover.
Introduce FinOps automation	Short term	Low	Optimise cloud spend via automated recommendations and tagging compliance.
Roll out data literacy programme	Long term	Medium	Empower business units to self-serve analytics responsibly.
Federate governance council	Long term	High	Scale policy adherence and stewardship as new domains onboard.

13.2 Generalisability

The architectural patterns generalise to other high-velocity domains such as quick-commerce, digital banking, and online gaming. Key prerequisites include:

Event-driven transaction systems capable of emitting change data capture events or API webhooks.

Organisational commitment to product-centric ownership of data domains.

Investment in automation, observability, and cloud-native security fundamentals.

13.3 Strategic Outlook

Future iterations can integrate machine learning for demand forecasting, anomaly detection, and marketing optimisation. Extending the platform with real-time experimentation frame- works, reinforcement learning, and digital twin simulations will unlock differentiated customer experiences.

Chapter 14

Conclusion

This thesis has presented a robust, resilient, and ethically governed ecommerce data platform capable of delivering near real-time KPIs across multiple channels. By integrating event-driven ingestion, governed dimensional modelling, automated CI/CD, and comprehensive observability, the solution addresses the research problem of delivering trustworthy analytics within minutes of operational events. The platform's modularity and dual-cloud readiness ensure longevity as the organisation scales and diversifies.

Beyond technical achievements, the project fostered cross-functional collaboration, strengthened data stewardship, and increased business confidence in data-driven decisions. Contin- uous improvement roadmaps emphasise experimentation, literacy, and governance, ensuring sustained value. The lessons learned contribute to the wider body of knowledge on cloud- native data engineering and provide a blueprint for organisations seeking to operationalise near real-time intelligence responsibly.

Generative AI Usage Documentation

.1 Prompt Catalogue

Table 1 documents representative prompts issued to ChatGPT (gpt-5-codex) and the purpose of the generated guidance.

Table 1: Sample AI prompts

Prompt Excerpt	**Usage**
"Summarise the benefits of dual-cloud data pipelines for ecommerce KPI delivery"	Informed executive summary narrative and recommendations on portability.
"Provide LaTeX code for a TikZ diagram illustrating an iterative methodology"	Seeded Figure 5.1 subsequently customised by the author.
"Outline data quality metrics suitable for ecommerce fact tables"	Inspired Section 7.4 content and validation scorecard structure.

.2 Human Validation

All AI outputs were critically reviewed, cross-referenced with project evidence, and adjusted to ensure accuracy and contextual relevance. No AI-generated text was inserted verbatim without editing. Analytical conclusions, recommendations, and performance claims are derived from empirical experimentation conducted by the author.

Additional Artefacts

.3 Runbook Excerpt

Pipeline Recovery Runbook

Trigger: Airflow SLA breach detected for order_pipeline.

Steps:

1. Inspect Airflow UI for failed tasks and review associated logs.
2. Execute Jenkins job pipeline-retry to rerun failed task group with idempotent payloads.
3. Validate data quality results via dbt source freshness command.
4. Notify stakeholders through Slack channel #data-ops with remediation summary.

.4 Data Dictionary Snapshot

Table 2: Illustrative data dictionary entries

Field	Type	Description	Sensitivity
order_id	UUID	Unique identifier for each order line	
customer_token	CHAR(36)	Tokenised customer identifier (non-reversible)	
promised_delivery_ts	TIMESTAMP	Timestamp committed to the customer at checkout	
net_revenue_amount	DECIMAL(18,2)	Revenue after discounts and taxes	

.5 Environment Inventory

Table 3: Infrastructure components per environment

Component	QA	Production	Notes
Airflow	AWS MWAA (small)	AWS MWAA (medium)	Autoscaling workers enabled in production.
Data Warehouse	Amazon Redshift ra3.xlplus	Amazon Redshift ra3.4xlarge	Reserved instances reduce cost.
Storage	S3 Standard-Infrequent Access	S3 Intelligent-Tiering	Lifecycle transitions after 30 days.
Monitoring	Datadog (free tier)	Datadog Pro	Synthetic monitoring active in production.
CI/CD	Jenkins on EC2 t3.large	Jenkins on EC2 m5.large	Executors scaled via auto-scaling group.

Manifesto Assets and Tools

.6 Tooling Footprint

Table **??** documents the automation stack referenced across Chapters **??** and **??** so reviewers can replicate the workflows.

Table 4: Tools used to operationalise the manifesto

Category	**Tool**	**Usage**
Orchestration	Apache Airflow (MWAA)	Coordinates ingestion, validation, and reverse ETL DAGs.
Quality	Great Expectations, dbt tests	Applies schema, null, and referential checks per bronze/silver table.
Observability	Datadog, OpenLineage	Captures metrics, traces, and lineage for SLA dashboards.
Security	AWS KMS, Azure Key Vault, HashiCorp Vault	Manages encryption keys, tokens, and API secrets.
Automation	Jenkins, Terraform Cloud	Executes CI/CD, IaC drift detection, and environment promotion.

.7 Glossary Supplement

The primary glossary appears before the main matter, but this appendix highlights additional manifesto-specific terms:

Data Contract A documented schema and SLA agreement between source and consumer teams.

Data Product Owner Accountable persona responsible for a dataset's lifecycle, SLA, and financial impact.

Lineage Event Metadata record describing upstream and downstream dependencies captured via OpenLineage.

Reverse ETL Operational use case where curated data is synchronised back into SaaS applications or services.

.8 Poster and Diagram Instructions

Manifesto Poster Placement

1. Download the gradient manifesto graphic from https://miro.com/app/board/uXjVPPwYZ-s=/ (export as PNG) and save it locally as report/figures/data engineering manifesto.png. The LaTeX will ren- der a placeholder box until the file is present.

2. Insert the poster immediately after Figure **??** if a full-page illustration is required, or print it separately for stakeholder workshops.

3. When integrating into main.pdf, ensure the caption references the manifesto pillars so that the examiner links it to Chapter **??**.

Branding Assets

1. Provide the host company logo file (PNG preferred) and save it as report/figures/host company logo.png. The cover page uses a conditional include and will show a framed placeholder until the logo is added.

2. Retain the existing Aivancity letterhead if available, or apply your institution-specific cover guidance before the declaration page.

3. Keep diagram exports (e.g., AWS deployment) under report/figures/ to avoid bloating the Git history; commit only the LaTeX references and supply binaries at release time.

References:

- Mohna, H. A., Barua, T., Mohiuddin, M., & Rahman, M. (2022). *AI-Ready Data Engineering Pipelines: A Review of Medallion Architecture and Cloud-Based Integration Models.* American Journal of Scholarly Research and Innovation, 1(1), 319–350.

- Powers, M. (2024). *Building the Medallion Architecture with Delta Lake.*

- Lee, D., Wentling, T., Haines, S., & Babu, P. (2024). *Delta Lake: The Definitive Guide.* (“Modern Data Lakehouse Architectures” chapter) O’Reilly / Databricks.

- van Eijk, T., Kumara, I., Di Nucci, D., Tamburri, D. A., & van den Heuvel, W.-J. (2024). *Architectural Design Decisions for Self-Serve Data Platforms in Data Meshes.*

- Goedegebuure, A., Kumara, I., Driessen, S., Di Nucci, D., Monsieur, G., van den Heuvel, W.-J., & Tamburri, D. A. (2023). *Data Mesh: A Systematic Gray Literature Review.*

- Wider, A., Verma, S., & Akhtar, A. (2023). *Decentralized Data Governance as Part of a Data Mesh Platform: Concepts and Approaches.*

- Bode, J., Kühl, N., Kreuzberger, D., Hirschl, S., & Holtmann, C. (2023). *Towards Avoiding the Data Mess: Industry Insights from Data Mesh Implementations.*

- Pingos, M., Ribeiro, L., & colleagues (2024). *Discovering Data Domains and Products in Data Meshes using Semantic Data Blueprints.* MDPI, Technologies, 12(7), 105.

- Francia, M., & others (2025). *Process-Driven Design of Cloud Data Platforms.*

- Dehghani, Z. (2020). *Data Mesh Principles and Logical Architecture.* Martin Fowler’s blog.

- Koppichetti, R. K. (2022 / uploaded 2025). *Decentralizing Data Management in Retail: A Review of Data Mesh Architecture.* (International Journal of Core Engineering & Management) via ResearchGate.

- Francia, M., & others (2025). *The Future of Data Management: A Delimitation of Data Platforms, Data Spaces, Data Meshes, and Data Fabrics. Information Systems & e-Business Management.*