# Executive Summary

This report documents the end-to-end ecommerce data pipeline and customer experience platform that the team implemented for analytics, automation, and continuous delivery. The initiative delivers a modular architecture spanning data ingestion, transformation, reporting, and customer-facing services that can be deployed on AWS today with a clear path to Azure tomorrow.

The solution unifies Apache Airflow, dbt, and a React customer portal on top of container-native infrastructure. Terraform, Jenkins, and a shared observability fabric allow each component to be managed through infrastructure-as-code and repeatable pipelines.

- Accelerate data availability for merchandising and marketing teams with near real-time KPIs.
- Enable blue/green-style application releases through Jenkins-driven automation for QA and main branches.
- Lay the groundwork for multi-cloud portability by modelling every environment through configurable variables.

Document generation completed on 13 November 2025.

# Introduction

Ecommerce organisations operate under constant pressure to interpret signals from web, mobile, fulfilment, and customer support channels. The project establishes a single governed data pipeline that hydrates the analytical data mart, while the customer application exposes curated experiences powered by the same data foundation.

## Background and Motivation

The legacy environment relied on ad-hoc scripts, manual database exports, and department-specific dashboards that were difficult to reconcile. Engineers spent a significant portion of their week maintaining brittle ETL jobs.

By introducing Apache Airflow orchestrations, dbt modelling, and a managed artifact repository, the team centralises governance, testing, and delivery mechanisms while reducing operational overhead.

## Objectives

- Consolidate source data streams from order intake, catalogue management, and customer feedback into a trusted warehouse.
- Provide stakeholders with curated dimensional models and KPIs refreshed within five minutes of operational events.
- Deliver a responsive customer experience application that surfaces personalised promotions and order insights.
- Automate infrastructure provisioning, application deployment, and validation through infrastructure-as-code and CI/CD workflows.

# Problem Definition and Scope

The project addresses data latency, inconsistent reporting, and limited deployment automation. It targets ecommerce order and catalogue data, marketing campaign performance, and customer service interactions.

## Business Challenges

Merchandising teams could not react to trending products because analytics lagged by several hours. Customer service lacked unified views of orders and returns, fragmenting the brand experience.

- Manual SQL exports delayed revenue reporting by up to two business days.
- Nightly cron jobs failed silently, creating blind spots in marketing attribution.
- Deployment scripts required administrator access on individual servers, increasing risk and cycle time.

## Project Scope

In scope: near real-time ingestion, curated marts for sales and marketing analytics, a React customer portal, and the automation toolchain spanning Terraform and Jenkins.

Out of scope: re-platforming transaction processing systems, custom machine learning models, and full ERP modernisation.

# Requirements Analysis

## Functional Requirements

- Schedule ingestion of order, inventory, campaign, and customer support feeds with retry-aware workflows.
- Transform raw events into conformed dimensional tables and aggregate marts accessible by BI tools.
- Expose REST endpoints for customer order status, personalised recommendations, and marketing content.
- Provide administrative dashboards summarising pipeline health and deployment status.

## Non-Functional Requirements

- Achieve 99.5% availability for customer-facing services with automated recovery strategies.
- Deliver analytics datasets within a five-minute freshness window from event capture.
- Support multi-cloud portability with abstracted configuration variables and stateless containers.
- Maintain compliance with data privacy regulations by enforcing encryption in transit and at rest.

## Stakeholder Expectations

Stakeholder	Interests	Success Indicators
Chief Digital Officer	Accelerated release cadence, resilient infrastructure	Deployment cycle time reduced by 60
Marketing Analytics	Trusted campaign attribution metrics	Dashboards refreshed within five minutes
Customer Support	Unified order and return history	Faster case resolution and higher CSAT
DevOps Team	Repeatable provisioning, observability	Infrastructure codified and tested in CI

[Stakeholder goals and measurable outcomes]

# Architecture Overview

The solution follows a modular layered architecture: data ingestion, orchestration, modelling, serving, and experience delivery. Infrastructure modules group networking, compute, storage, and security primitives.

## High-Level Components

- Data Ingestion Layer: Kafka-compatible stream via AWS Kinesis Data Streams (extensible to Azure Event Hubs).
- Processing Layer: Apache Airflow orchestrations executing Python operators and dbt models.
- Storage Layer: Amazon S3 for raw and curated zones, Amazon RDS PostgreSQL warehouse (portable to Azure Blob and Azure Database for PostgreSQL).
- Serving Layer: RESTful APIs and React frontend containerised for ECS Fargate with parallels in Azure Container Apps.
- Observability Layer: Prometheus-compatible metrics and structured logging shipped to CloudWatch (or Azure Monitor).

## Data Flow

Synthetic orders emulate browse-to-purchase journeys and land in the ingestion bucket. Airflow DAGs apply validation, enrichments, and merge operations before persisting dimensional models. The dbt project builds incremental models, and the resulting datasets feed visualisations as well as the customer API.

## Component Interaction

- Airflow invokes dbt via KubernetesPodOperator or ECSOperator, ensuring execution parity across environments.
- Jenkins pipelines publish application images to Amazon ECR and trigger ECS rolling updates with health checks.
- Terraform outputs share load balancer endpoints, security group IDs, and credential references back to Jenkins for smoke testing.

# Data Pipeline Design

## Source Systems

Order intake events, inventory adjustments, marketing campaign metadata, and customer service notes are synthesised to mirror enterprise patterns. Each source publishes JSON payloads with schema versioning to simplify evolution.

## Airflow Orchestration

Two primary DAGs run on staggered schedules: the order generator and the end-to-end data pipeline. Task groups isolate staging, validation, transformation, and reporting phases.

- Automatic retries with exponential backoff for transient failures.
- Dataset-level SLAs tracked with Airflow’s alerting mechanisms.
- Config-driven environment variables for bucket names, prefixes, and schema locations.

## dbt Modelling

The dbt project structures staging, intermediate, and mart models. Macros enforce naming conventions, and tests cover relationships, uniqueness, and freshness. Snapshots preserve slowly changing dimension history.

## Data Quality Controls

- Great Expectations-style assertions within Airflow operators for row counts, null ratios, and schema compatibility.
- dbt tests run within CI pipelines to prevent regressions before deployment.
- MinIO-based quarantine buckets capture anomalous files for investigation.

# Customer Application Experience

The customer-facing web application builds on React and Tailwind CSS, consuming the same API layer surfaced by the data pipeline. User journeys include order tracking, personalised recommendations, and loyalty insights.

## Frontend Architecture

Component-based design patterns enable reuse across dashboards. State management leverages Redux Toolkit, and API clients utilise Axios with interceptors for authentication and telemetry.

- Responsive layouts supporting desktop, tablet, and mobile breakpoints.
- Accessibility-first semantics with WCAG 2.1 AA targets.
- Feature flags delivered through environment variables to toggle beta modules.

## API Layer

FastAPI orchestrates REST endpoints packaged into containers. Routes cover customer profiles, orders, inventory lookups, and marketing banners. JSON Web Tokens secure the endpoints, with scope-based authorisation for support teams versus end customers.

## Performance Considerations
- Server-side caching for personalised recommendations to meet sub-200ms response targets.
- CDN-backed static asset delivery through Amazon CloudFront (or Azure Front Door).
- Synthetic monitoring via lightweight Playwright scripts executed post-deployment.

# Infrastructure Automation
Terraform codifies VPCs, ECS services, RDS instances, S3 buckets, and IAM policies. Modules parameterise provider-specific attributes, enabling a swap between AWS and Azure with minimal changes to variables.

## Module Strategy
- Core networking module defines CIDR blocks, subnets, route tables, and security groups.
- Data services module provisions PostgreSQL, S3 buckets, and KMS keys.
- Application module deploys ECS clusters, task definitions, and load balancers.
- Jenkins module configures EC2 (or Azure VM) hosts with bootstrap scripts for Terraform, Docker, and pipeline agents.

## Environment Configuration
Separate ‘qa‘ and ‘main‘ variable files tune capacity, instance sizes, and observability flags. Workspace usage keeps state isolated per environment.

Setting	QA Branch	Main Branch
Fargate Task Count	1 per service	2 per service
Database Class	db.t3.small	db.m5.large
NAT Gateway	Disabled	Enabled
Alerting	Slack webhooks	Slack + PagerDuty

[Key infrastructure parameter differences]

## Azure Portability
Provider-agnostic variables abstract network CIDR ranges, container registries, secret stores, and monitoring endpoints. Implementing the Azure module requires mapping these variables to Azure Virtual Networks, Container Apps, Key Vault, and Application Insights.

# Continuous Integration and Delivery
Jenkins orchestrates build, test, security scanning, and deployment stages. Pipelines are declarative with shared libraries for Terraform wrappers and notification hooks.

## Pipeline Stages

- Checkout and dependency caching.
- Static code analysis via Flake8, Bandit, and SonarQube scanners.
- Unit and integration testing triggered per branch.
- Docker image build, tagging, and push to Amazon ECR (or Azure Container Registry).
- Terraform plan and apply stages gated by approval for the main branch.

## Quality Gates

SonarQube enforces code coverage, duplication, and security standards. OWASP ZAP baseline scans detect regressions in the customer application before promotion.

## Release Management

QA branch deployments provision ephemeral infrastructure for user acceptance testing. Once approved, the main branch pipeline promotes artefacts and applies Terraform changes with blue/green-style ECS updates.

# Security and Compliance

Security is embedded across the stack: network segmentation, encryption, identity governance, and observability.

## Identity and Access Management

- Least-privilege IAM roles for Airflow, dbt, ECS tasks, and Jenkins agents.
- Secrets stored in AWS Systems Manager Parameter Store with envelope encryption.
- Multi-factor authentication enforced for console and bastion access.

## Data Protection

- TLS termination at the application load balancer with automatic certificate rotation.
- KMS-managed encryption for S3 buckets and RDS storage.
- Audit trails captured through CloudTrail and forwarded to a central log bucket.

## Governance and Compliance

Policies align with GDPR and CCPA for customer data minimisation and retention. Automated policy-as-code checks via Terraform Cloud or Open Policy Agent validate infrastructure changes.

# Testing Strategy

Testing spans unit, integration, system, performance, and user acceptance layers. Test data management ensures consistent results across runs.

## Automated Testing

- Pytest-based unit tests cover data transformation helpers and API utilities.
- dbt ‘run‘ and ‘test‘ commands executed in CI to validate SQL models.
- Containerised integration suites spin up temporary Postgres and MinIO instances for end-to-end verification.

## Performance and Load

Locust scenarios simulate concurrent order lookups and marketing banner rotations. Metrics feed into Jenkins to track latency thresholds over time.

## User Acceptance Testing

QA environments allow merchandisers and customer support representatives to validate workflows before release. Feedback loops integrate into Jira for rapid iteration.

# Operations and Monitoring

## Observability Stack

Metrics collected through CloudWatch Container Insights, Application Load Balancer access logs, and custom business KPIs exported from Airflow.

## Alerting

- Slack notifications for build failures and Airflow SLA misses.
- PagerDuty escalations for production incidents exceeding defined thresholds.
- Daily health summaries combining pipeline statistics and application uptime.

## Runbooks

Runbooks outline recovery steps for ingestion backlog, ECS deployment rollback, and database failover. They include decision matrices, command snippets, and escalation contacts.

# Project Management

Delivery followed an agile cadence with two-week sprints, backlog grooming, and regular stakeholder demos.

## Timeline

Phase	Duration	Key Deliverables
Initiation	2 weeks	Charter, stakeholder alignment
Architecture & Design	4 weeks	Solution blueprints, tech stack selection
Implementation	10 weeks	Data pipeline, customer app, Terraform modules
Testing & Hardening	4 weeks	Performance tuning, security reviews
Deployment	2 weeks	Production rollout, documentation

## Resource Plan
- Product Owner providing continuous backlog prioritisation.
- Data Engineer, Analytics Engineer, and Backend Engineer collaborating on ingestion and modelling.
- Frontend Engineer and UX Designer crafting the customer experience.
- DevOps Engineer maintaining CI/CD and infrastructure automation.

## Communication
Weekly status reports, daily stand-ups, and stakeholder reviews ensured transparency and rapid risk mitigation.

# Risk Assessment and Mitigation

Risk	Impact	Probability	Mitigation
Cloud service quota exhaustion	Delays in provisioning infrastructure	Medium	Monitor AWS service quotas and
Data quality regressions	Incorrect KPIs affecting decisions	Medium	Automated validation with alerting and dat
Pipeline deployment failure	Extended downtime for services	Low	Blue/green deployments with automated rollb
Skill gaps for Azure migration	Slower adoption of alternate cloud	Medium	Documented abstraction layers and

[Risk register snapshot]

## Assumptions
- Student AWS accounts maintain baseline service availability throughout the academic term.
- Business stakeholders are available for UAT sign-off each sprint.
- Data privacy requirements remain consistent during project delivery.

# Cost Optimisation
Cost governance focuses on leveraging free-tier or student-program entitlements wherever possible while planning for production-ready scaling.

## AWS Cost Profile

Service	QA Estimate	Main Estimate	Notes
ECS Fargate	$35/month	$140/month	Autoscaling adjusts tasks during campaigns.
RDS PostgreSQL	$45/month	$220/month	Reserved instances reduce steady-state costs.
S3 Storage	$5/month	$15/month	Lifecycle policies transition logs to Glacier.
Jenkins EC2	$20/month	$40/month	Spot instances during non-peak hours.

## Azure Equivalents

Equivalent services include Azure Container Apps, Azure Database for PostgreSQL Flexible Server, Azure Blob Storage, and Azure DevOps or Jenkins on Azure VM. Cost calculators map the same utilisation profiles to ensure comparability.

## Optimisation Strategies

- Automated shutdown of non-production environments outside business hours.
- Rightsizing through monthly review of CloudWatch utilisation metrics.
- Use of serverless offerings (Lambda, Azure Functions) for burst workloads such as batch enrichment jobs.

# Future Enhancements

The modular design enables incremental adoption of advanced analytics and experience features.

- Introduce machine learning models for personalised recommendations using Amazon SageMaker or Azure Machine Learning.
- Expand the customer portal with proactive delivery notifications and augmented reality product previews.
- Adopt event-driven microservices for payment, loyalty, and returns to decouple teams further.
- Integrate data governance catalogues such as DataHub or Azure Purview for lineage and stewardship.

# Conclusion

The ecommerce data pipeline and customer application deliver a cohesive platform that accelerates insight-to-action loops. Terraform and Jenkins automation lower the barrier to repeatable deployments, while the cloud-neutral approach keeps future strategic options open.

With observability, testing, and governance embedded across the lifecycle, the organisation can confidently evolve the solution, onboard new data domains, and support global customer experiences.

# Glossary

Term	Definition
Airflow	Open-source workflow orchestrator for authoring, scheduling, and monitoring data pipelines.
dbt	Data build tool used to transform data in warehouses through SQL and testing frameworks.
ECS Fargate	Serverless compute engine for containers managed by Amazon Elastic Container Service.
Infrastructure-as-Code	Approach of managing infrastructure through declarative configuration files.
KPI	Key Performance Indicator quantifying business objectives.
Terraform	HashiCorp tool for building, changing, and versioning infrastructure safely.

# References

1. AWS Architecture Center. ’Deploying Containerized Applications on AWS Fargate’.
2. dbt Labs. ’dbt Documentation and Best Practices’.
3. Apache Airflow. ’Airflow Scheduler Concepts’.
4. HashiCorp. ’Terraform Module Design Guidelines’.
5. OWASP Foundation. ’Zed Attack Proxy Baseline Scan’.
6. Microsoft. ’Azure Container Apps Overview’.

# Appendix A: Jenkins Pipeline Listing

The appendix summarises the declarative stages configured in Jenkins, highlighting shared libraries, Terraform wrappers, and notification hooks used across QA and main branch pipelines.

# Appendix B: Environment Variable Catalogue

Variable	Description	Scope
DATA_PIPELINE_SOURCE_DIR	Override path for synthetic CSV inputs	Airflow
DBT_TARGET	Environment profile executed by dbt	Airflow, Jenkins
REACT_APP_API_URL	Endpoint for customer API	Customer App
TERRAFORM_VAR_file	Selected variable set for environment	CI/CD