ShopMetrics Inc.

30-Day Implementation Plan

Real-Time E-Commerce Analytics Platform

Project Code: SHOPMETRICS-DATA-001

Databricks Free Edition · Confluent Kafka · GitHub Actions · Streamlit

Total Cost: $0 / month

Data Engineering Team · February 2026 · v1.0

Databricks	Confluent Kafka	GitHub Actions	Delta Lake	Streamlit

■ Final Tech Stack (100% Free)

Component	Service	Purpose
Data Platform	Databricks Free Edition	Processing, Delta Lake, Unity Catalog
Streaming	Confluent Cloud Free Tier	Real Kafka cluster for clickstream events
Orchestration	Databricks Jobs (native)	Scheduling batch pipeline
CI/CD	GitHub Actions	Automated testing and deployment
IaC	Databricks Asset Bundles	Infrastructure as code
Dashboard	Streamlit Community Cloud	Public-facing analytics (FR-017)
Version Control	GitHub — shopmetrics/ecommerce-lakehouse	Source of truth (public repo)
Total Cost		**$0 / month**

Confluent Cloud Free Tier — What You Get

- 1 Basic Kafka cluster (forever free) — real managed Kafka, not simulated [FR-002]
- 5 GB storage, 250 MB/day throughput — sufficient for ShopMetrics clickstream volume
- Full Kafka API compatibility + Schema Registry + Kafka REST API

■ Project Context

Project	Real-Time E-Commerce Analytics Platform
Client	ShopMetrics Inc.
Project Code	SHOPMETRICS-DATA-001
BRD Version	v1.0 — Approved February 12, 2026
Team	Data Engineering Team — reviewed by Arjun Mehta (Head of Engineering) & Sonal Verma (Product Owner)
Approved By	Priya Sharma — VP Technology
Objective	Build zero-cost medallion lakehouse with real-time Kafka streaming to replace ad-hoc production DB queries
Repo	github.com/shopmetrics/ecommerce-lakehouse (public)

BRD Traceability

Each task in this plan is traceable to a functional requirement (FR-xxx) or non-functional requirement (NFR-xxx) from the ShopMetrics BRD v1.0. Acceptance criteria (AC-xxx) are referenced where a task directly satisfies a sign-off condition.

■ Week 1 — Foundation & Batch Data Setup

Day 1 — Tuesday — 30 mins

- Sign up for Databricks Free Edition (databricks.com/try-databricks)
- Create Unity Catalog per BRD §7.1: catalog=ecommerce, schemas=bronze/silver/gold [FR-003]
- Create volume: ecommerce.bronze.raw_data as CSV landing zone
- Verify Unity Catalog lineage tracking enabled — required for FR-012

✓ Deliverable: Databricks workspace ready — satisfies NFR-009 (no credentials in VCS)

Day 2 — Wednesday — 30 mins

- Create GitHub repo: shopmetrics/ecommerce-lakehouse (public) [FR-017]
- Initialise with README, .gitignore (Python), MIT LICENSE
- Create folder structure: /src/{bronze,silver,gold,utils}/, /tests/, /data-generator/, /docs/
- Reference BRD project code SHOPMETRICS-DATA-001 in README header

✓ Deliverable: GitHub repo initialised — foundation for FR-017 and the CI/CD pipeline

Day 3 — Thursday — 45 mins

- Write data-generator/generate_orders.py — 100K orders (2022–2026) [FR-001]
- Schema per BRD §7.2: order_id, customer_id, product_id, order_date, total_amount, status
- Statuses must match BRD-defined values: pending, completed, cancelled, refunded
- Write data-generator/generate_customers.py — 10K customers with email, region, signup_date

✓ Deliverable: Synthetic ShopMetrics source data ready — simulates operational DB exports for FR-001

Day 4 — Friday — 45 mins

- Write data-generator/generate_products.py — 1K products across 8 categories [FR-001]
- Upload all CSVs to Unity Catalog volume ecommerce.bronze.raw_data
- Create src/bronze/ingest_orders.py — add ingested_at and source_file audit columns per BRD §7.2
- Write to ecommerce.bronze.orders_raw — verify schema matches BRD §7.2 exactly
- Test with SELECT, COUNT, DESCRIBE

✓ Deliverable: First ShopMetrics Delta table: orders_raw — FR-001 batch ingestion verified

Day 5 — Saturday

3 hours

- Ingest ecommerce.bronze.customers_raw and ecommerce.bronze.products_raw [FR-001]
- Create src/silver/clean_orders.py: dedup on order_id, null handling, status validation [FR-003]
- Reject records with null order_id — supports AC-004 (data quality null check)
- Write to ecommerce.silver.orders_clean
- Draft docs/data-model.md — reference BRD §7 for all column definitions

✓ Deliverable: Bronze layer done (FR-001 satisfied) — first silver transformation delivering FR-003

Day 6 — Sunday

3 hours

- Create src/silver/dim_customers.py — SCD Type 2 per BRD FR-004
- Implement: surrogate_key, is_current, effective_start_date, effective_end_date using Delta MERGE
- Write to ecommerce.silver.dim_customers
- Manually test AC-003: update customer email, confirm old record end_date populated
- Create src/silver/dim_products.py — write to ecommerce.silver.dim_products
- Write tests/unit/test_scd2_customers.py covering AC-003 with pytest

✓ Deliverable: SCD Type 2 done — FR-004 implemented, AC-003 unit test passing

■ Week 2 — Gold Layer + Confluent Kafka Setup

Day 7 — Monday — 30 mins

- Create src/gold/daily_sales_summary.py [FR-006]
- Columns per BRD §7.3: date, category, revenue, order_count, avg_order_value, updated_at
- Join silver.orders_clean with dim_products for category breakdown
- Write to ecommerce.gold.daily_sales_summary — verify AC-005 (revenue within 0.01% of manual calc)

✓ Deliverable: daily_sales_summary gold table — FR-006 delivered, AC-005 verifiable

Day 8 — Tuesday — 30 mins

- Create src/gold/customer_ltv.py [FR-007]
- Columns per BRD §7.3: customer_id, total_revenue, order_count, first/last order dates, ltv_segment
- Segmentation per BRD FR-007: High (>$500), Medium ($100-500), Low (<$100)
- Write to ecommerce.gold.customer_ltv — manually verify AC-006 (3 test profiles segmented correctly)

✓ Deliverable: customer_ltv gold table — FR-007 delivered, AC-006 verifiable

Day 9 — Wednesday — 45 mins

- Create src/gold/product_performance.py [FR-008]
- Calculate units_sold, total_revenue, category_rank per reporting period
- Write to ecommerce.gold.product_performance
- Run full batch pipeline end-to-end — verify NFR-001 (pipeline completes within 60 minutes)

✓ Deliverable: Batch medallion pipeline complete — FR-006, FR-007, FR-008 all satisfied

Day 10 — Thursday — 45 mins

- Sign up for Confluent Cloud — create free Basic Kafka cluster [FR-002]
- Create topic: clickstream-events (3 partitions, 24-hr retention to stay within 5GB free quota)
- Store API key + secret ready for Databricks Secret Scope on Day 12 [NFR-009]
- Confirm cluster region matches Databricks workspace for lowest latency

✓ Deliverable: Live Confluent Kafka cluster ready — prerequisite for FR-002 streaming ingestion

Day 11 — Friday — 45 mins

- Create data-generator/produce_clickstream.py [FR-002]
- Events per BRD §8.3: page_view, add_to_cart, purchase, search at ~1 event/sec
- Schema per BRD §7.2: event_id, customer_id, product_id, event_type, session_id, page, event_ts, kafka_offset
- Test locally: run script, verify messages appear in Confluent Cloud UI

✓ Deliverable: Python Kafka producer running — ShopMetrics clickstream events flowing

Day 12 — Saturday — 4 hours

- Create Databricks Secret Scope — store CONFLUENT_BOOTSTRAP_SERVERS, API_KEY, API_SECRET [NFR-009]
- Create src/bronze/stream_clickstream.py — Structured Streaming from Kafka [FR-002]
- Parse JSON per BRD §7.2, add ingested_at audit column
- Write stream to ecommerce.bronze.clickstream_raw with checkpoint in Unity Catalog volume
- Apply 10-minute watermark for late-arriving events per BRD §8.3
- Confirm sub-5-minute end-to-end latency, satisfying NFR-002

✓ Deliverable: Kafka to bronze streaming working — FR-002 satisfied, NFR-002 latency verified

Day 13 — Sunday — 4 hours

- Create src/silver/sessionize_clickstream.py — 30-min inactivity window per BRD FR-010
- Compute session_duration and event_count, write to ecommerce.silver.clickstream_sessions
- Create src/gold/hourly_traffic_metrics.py [FR-009]
- Aggregate per BRD FR-009: page_views, unique_visitors, add_to_cart_rate, purchase_rate by hour
- Write to ecommerce.gold.hourly_traffic_metrics
- Full streaming E2E: producer > Kafka > bronze > silver > gold confirmed

✓ Deliverable: Full streaming pipeline done — FR-002, FR-009, FR-010 all satisfied

■ Week 3 — CI/CD, Testing & Asset Bundles

Day 14 — Monday — 30 mins

- Generate Databricks PAT: Settings > Developer > Access Tokens — 730-day max lifetime
- Add GitHub Secrets: DATABRICKS_TOKEN, DATABRICKS_HOST — never in repo [NFR-009]
- Add GitHub Secrets: CONFLUENT_API_KEY, CONFLUENT_API_SECRET
- Set calendar reminder 30 days before expiry — mitigates Risk R-003 from BRD §11

✓ Deliverable: All secrets in GitHub — NFR-009 satisfied, R-003 risk mitigation in place

Day 15 — Tuesday — 45 mins

- Create databricks.yml Asset Bundle config for SHOPMETRICS-DATA-001
- Define dev and prod targets matching BRD-defined environments
- Daily batch job: bronze_ingest > silver_transform > gold_aggregate > data_quality > optimise
- All tasks use serverless compute — respects constraint C-003 (max 5 concurrent tasks)
- Validate locally: databricks bundle validate — supports AC-013 (reproducible deploy)

✓ Deliverable: Asset Bundle config validated — IaC in place per BRD §8.1

Day 16 — Wednesday — 45 mins

- Write tests/unit/test_scd2_customers.py — covers AC-003 (end_date populated on update)
- Write tests/unit/test_gold_sales.py — covers AC-005 (revenue calculation accuracy)
- Write tests/unit/test_data_quality.py — covers AC-004 (null check fails pipeline at >1%)
- Write tests/unit/test_kafka_producer.py — validates event schema against BRD §7.2
- Run full suite locally — 12+ tests passing

✓ Deliverable: Test suite passing — AC-003, AC-004, AC-005, schema validation all covered

Day 17 — Thursday — 45 mins

- Create .github/workflows/ci.yml — triggers on every pull request [AC-007]
- Jobs: black (formatting), flake8 (linting), mypy (type checks), pytest
- Add .pre-commit-config.yaml for local enforcement before push
- Create test PR — verify CI completes within 10 minutes per AC-007

✓ Deliverable: Automated CI on every PR — AC-007 satisfied

Day 18 — Friday — 45 mins

- Create .github/workflows/cd.yml — triggers on merge to main [AC-008]
- Steps: databricks bundle deploy -t prod, trigger smoke test job
- Merge a real change — verify automated deployment to Databricks production workspace
- Confirm no secrets appear in workflow logs — final NFR-009 check

✓ Deliverable: Automated CD on merge to main — AC-008 satisfied, zero-downtime deployment confirmed

Day 19 — Saturday — 3 hours

- Create .github/workflows/schedule.yml — cron: 0 2 * * * (2 AM UTC daily) [NFR-008]
- Failure triggers email alert to Data Engineering Team within 5 minutes
- Create src/utils/data_quality.py: row count, null checks, schema validation [FR-005]
- Null check on order_id > 1% threshold halts pipeline — satisfies AC-004
- Add DQ gates to all gold notebooks before write
- Write docs/deployment-guide.md covering full setup — satisfies AC-013

✓ Deliverable: Scheduled pipeline with DQ gates — FR-005, NFR-008, AC-004, AC-013 all covered

Day 20 — Sunday — 3 hours

- Run OPTIMIZE + ZORDER on daily_sales_summary, customer_ltv, product_performance [FR-011]
- Add liquid clustering to daily_sales_summary (cluster on date, category) [FR-011]
- Write VACUUM with 7-day retention for old Delta file cleanup
- Tag Unity Catalog tables with owner=data-eng, domain=ecommerce, sensitivity=internal [FR-012]
- Write docs/architecture-decisions.md: Confluent vs self-hosted, native vs Airflow scheduling
- Full E2E run — confirm NFR-001 (< 60 mins) and NFR-007 (idempotent re-run)

✓ Deliverable: Optimised, governed codebase — FR-011, FR-012, NFR-001, NFR-007 all verified

■ Week 4 — Dashboard & Public Launch

Day 21 — Monday — 30 mins

- Sign up for Streamlit Community Cloud — connect to shopmetrics/ecommerce-lakehouse [FR-017]
- Create dashboard/streamlit_app.py with ShopMetrics branding and page config
- Create dashboard/requirements.txt: streamlit, databricks-sql-connector, pandas, plotly
- Store DATABRICKS_TOKEN and SQL_HTTP_PATH as Streamlit secrets (not in repo — NFR-009)

✓ Deliverable: Streamlit app scaffold live — FR-017 (public access) foundation in place

Day 22 — Tuesday — 45 mins

- Start Databricks SQL Warehouse — get HTTP path from warehouse settings [FR-013]
- Create dashboard/utils/databricks_connector.py with retry logic for C-004 cold-start (1-5 min)
- Implement Parquet fallback: load from snapshots if warehouse unavailable [FR-018]
- Test connection locally — verify AC-009 (dashboard loads within 10s after warm-up)

✓ Deliverable: SQL warehouse connected — FR-018 resilience and NFR-003 (95% uptime) safeguarded

Day 23 — Wednesday — 45 mins

- Build dashboard/pages/1_Sales_Performance.py — serves ShopMetrics Sales team [FR-013]
- KPIs from ecommerce.gold.daily_sales_summary: Total Revenue, Order Count, AOV (last 90 days)
- Daily revenue line chart and category breakdown pie from FR-006 gold table
- Add @st.cache_data(ttl=3600) on all queries — mitigates R-004 cold start risk

✓ Deliverable: Sales Performance page live — FR-013 delivered for Sales team

Day 24 — Thursday — 45 mins

- Build dashboard/pages/2_Customer_Analytics.py — serves Marketing team [FR-014]
- LTV segment breakdown from ecommerce.gold.customer_ltv (FR-007 data)
- Cohort analysis table, LTV distribution histogram, date range filter
- Enables Marketing team's campaign analysis use case from BRD §2.2

✓ Deliverable: Customer Analytics page live — FR-014 delivered for Marketing team

Day 25 — Friday — 45 mins

- Build dashboard/pages/3_Product_Insights.py — serves Finance team [FR-015]
- Top 10 products by revenue from ecommerce.gold.product_performance (FR-008 data)
- Category performance bar chart, units sold vs revenue scatter
- Revenue-by-category view addresses Finance KPI requirement from BRD §4

✓ Deliverable: Product Insights page live — FR-015 delivered for Finance team

Day 26 — Saturday — 5 hours

- Build dashboard/pages/4_Kafka_Traffic.py — showpiece real-time page [FR-016]
- Hourly traffic from ecommerce.gold.hourly_traffic_metrics (FR-009, Kafka-sourced data)
- Live conversion funnel: page_views > add_to_cart > purchase (Marketing's top request)
- Caption: 'Powered by Confluent Cloud Kafka + Databricks Structured Streaming'
- Export gold tables to Parquet snapshots in pipeline — activates FR-018 fallback
- Deploy to Streamlit Community Cloud — test public URL [FR-017]
- Verify AC-010: disconnect SQL warehouse, confirm Parquet fallback activates automatically

✓ Deliverable: All 4 pages live — FR-013 to FR-018 satisfied, AC-009 and AC-010 passed

Day 27 — Sunday — 5 hours

- Polish dashboard: ShopMetrics colour scheme, sidebar with architecture overview
- Add 'About SHOPMETRICS-DATA-001' section with BRD project code reference
- Verify all four pages render without errors — satisfies AC-011
- Write comprehensive README: architecture diagram, live dashboard link, quick start guide
- Run: grep -r 'CONFLUENT\|DATABRICKS_TOKEN' . — verify zero secrets in repo [AC-012]
- Screenshot all pages for LinkedIn portfolio post

✓ Deliverable: Polished dashboard, clean README — AC-011 and AC-012 verified

■ Week 5 — Documentation & Public Launch

Day 28 — Monday

45 mins

- Write docs/deployment-guide.md — step-by-step from zero to live [AC-013]
- Sections: Confluent Cloud cluster setup, Databricks Secret Scope, Asset Bundle deploy, Streamlit deploy
- Have a second person follow the guide on a fresh machine — validate AC-013 end-to-end
- Commit guide and confirm it is discoverable from the README

✓ Deliverable: Reproducible deployment guide complete — AC-013 formally satisfied

Day 29 — Tuesday

1 hour

- Record demo video (5-7 minutes) for portfolio and LinkedIn
- 1) ShopMetrics problem statement from BRD §2.1 (30s)
- 2) Architecture diagram: medallion + Kafka flow (1 min)
- 3) Confluent Cloud: live clickstream-events topic with messages (1 min)
- 4) Databricks: pipeline job run, Delta tables, Unity Catalog lineage (1 min)
- 5) GitHub Actions: CI/CD run passing (30s)
- 6) Live Streamlit dashboard: all 4 pages walkthrough (2 mins)
- Upload to YouTube — link added to README

✓ Deliverable: Demo video ready — captures full ShopMetrics platform story for interviewers

Day 30 — Wednesday

2 hours

- Write Medium post: 'ShopMetrics Case Study: $0 Real-Time E-Commerce Lakehouse'
- Frame as real client engagement — reference BRD, stakeholders, requirement IDs
- Highlight Kafka as the differentiator from standard batch-only portfolios
- LinkedIn: live dashboard URL, GitHub repo, demo video — tag Databricks + Confluent
- Share in r/dataengineering and relevant DE Slack communities
- Update resume with ShopMetrics project bullet points (see section below)

✓ Deliverable: ShopMetrics platform fully public — 30-day implementation plan complete!

■ Key Milestones

Milestone	Day	BRD Requirements Satisfied
Bronze layer complete	5	FR-001, FR-003 — batch ingestion + medallion foundation
Batch medallion done	9	FR-006, FR-007, FR-008 — all gold tables for Sales, Marketing, Finance
Kafka streaming live	13	FR-002, FR-009, FR-010 — real-time clickstream pipeline end-to-end
CI/CD operational	18	NFR-009, AC-007, AC-008 — automated test and deploy pipeline
All 4 dashboard pages live	26	FR-013 to FR-018 — all dashboard requirements satisfied
All ACs signed off	27	AC-001 to AC-014 — full acceptance criteria verified
Project publicly launched	30	Blog, LinkedIn, resume — portfolio asset complete

Emergency Fallback Plan

MUST HAVE	DROP IF BEHIND	ADD AFTER LAUNCH
• orders_raw bronze + silver clean • Kafka > clickstream_raw stream • daily_sales_summary gold • 1 Streamlit page (Sales) • CI/CD on GitHub Actions	• Customer & Product dashboard pages • Integration tests • Blog post • Liquid clustering (FR-011)	• Demo video • Medium article • Unity Catalog tags (FR-012) • Advanced NFR optimisations

Resume Bullet Points

• Architected and delivered real-time e-commerce analytics platform (ShopMetrics Inc.) on Databricks, ingesting live clickstream events from Confluent Cloud Kafka through medallion architecture (Bronze/Silver/Gold) with Delta Lake and Unity Catalog governance

• Implemented Confluent Cloud Kafka streaming pipeline with Databricks Structured Streaming — processing 1M+ clickstream events with sessionisation, watermarking, and < 5-minute end-to-end latency

• Built production CI/CD pipeline using GitHub Actions and Databricks Asset Bundles — automated testing (pytest, black, mypy), zero-downtime deployments across dev and prod environments

• Delivered public-facing Streamlit analytics dashboard backed by Databricks SQL warehouse, serving Sales, Marketing, and Finance teams — implemented caching and Parquet fallback for 95% uptime

Daily Tracking Template

```
Date: _____ Planned Time: ___ mins Actual Time: ___ mins Tasks Completed: [ ]
_____ [ ] _____ [ ] _____ Blockers:
_____ BRD Requirements Progressed: FR-___ /
NFR-___ / AC-___ Tomorrow's Priority: _____
```

SHOPMETRICS-DATA-001 · Start Day 1 tomorrow. Good luck! ■