**Smart Manufacturing Data Analytics Platform – End-to-End DevOps & DataOps**

**1. Problem Statement**

AutoForge Ltd., a global automotive manufacturer, struggles with:

* Delayed analytics due to siloed legacy systems.
* Frequent deployment errors causing production data delays.
* Lack of real-time monitoring for IoT-based predictive maintenance.
* Inefficient model retraining due to absence of drift monitoring.
* Manual, error-prone deployment processes without rollback capability.

**2. Skill Tower to Develop the Project**

This project leverages skills across multiple technology and process towers:

**Data Engineering & Analytics**

* Databricks development (Notebook widgets, DBUtils parameters, JSON return to pipelines)
* Cluster management (Jobs vs interactive, cluster policies)
* Fan-out/Fan-in orchestration

**DevOps & CI/CD**

* GitHub Actions workflows (re-usable jobs, matrix builds, OIDC login, artefact caching)
* Environment configs (bundle.yml), deploy vs sync, validate & tag
* Release pipelines (Build→Release, smoke tests, rollback, promote to Test)
* ARM/Bicep export, parameter files, Linked IR tokens

**Data Quality & ML Ops**

* Analyzers & metrics repository
* Red-Green-Refactor TDD for data transformations & ML models
* Drift monitoring with automated retraining
* Metrics dashboards

**Security & Governance**

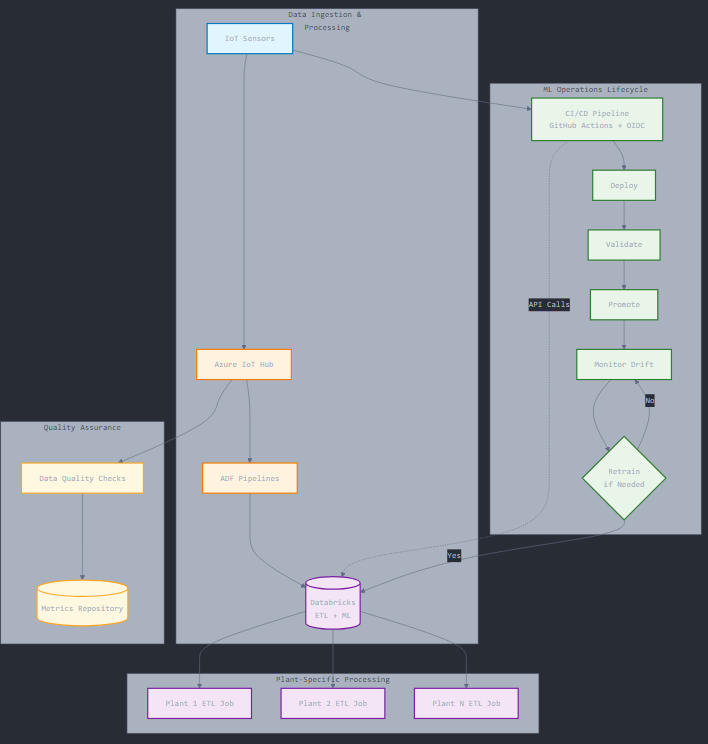
* Secret scopes, branch protection rules
* Role-based cluster policies
* Artefact retention policies

**3. Use Case / Architecture Diagram**

**Use Case:**

* Ingest IoT sensor data from multiple plants.
* Process and validate in Databricks using parameterized notebooks.
* Apply ML models for predictive maintenance.
* Deploy ETL & ML pipelines via GitHub Actions.
* Monitor drift and trigger retraining.
* Auto-roll back if smoke tests fail.

**Architecture (Conceptual Flow):**



**4. User Stories**

1. **As a Data Engineer**, I want to run parameterized ETL notebooks for different plants so that I can reuse code for multiple deployments.
2. **As a DevOps Engineer**, I want a CI/CD pipeline with smoke tests and rollback so that production jobs remain stable.
3. **As a Data Scientist**, I want drift monitoring so that ML models retrain automatically when accuracy drops.
4. **As a Security Officer**, I want secrets in secure scopes so that credentials are never exposed.
5. **As a Project Manager**, I want automated release notes so that stakeholders have a clear deployment log.

**5. Expected Deliverables**

* **Databricks ETL notebooks** with widgets & DBUtils param handling.
* **JSON return mechanism** for pipelines.
* **Cluster policies** for cost control & compliance.
* **Fan-out/fan-in job orchestration** for parallel processing.
* **Data quality metrics repository**.
* **ML drift monitoring system** with retrain triggers.
* **GitHub Actions workflows** with re-usable jobs, matrix builds, and artefact caching.
* **bundle.yml** environment configs for Dev/Test/Prod.
* **Release pipeline** with validation, tagging, promotion, rollback.
* **Metrics dashboard** for ETL & ML health monitoring.

**6. Milestone and Duration**

| **Milestone** | **Duration** | **Key Deliverables** |
| --- | --- | --- |
| Requirements & Architecture | 1 hr | Use case diagram, tech stack selection |
| Databricks Dev & Param Setup | 2 hr | ETL notebooks with widgets, JSON returns |
| Cluster Governance | 1 hr | Jobs vs interactive, policies enforced |
| Data Quality & ML Ops | 2 hr | Metrics repo, drift monitoring |
| CI/CD Setup | 2 hr | GitHub Actions workflows, bundle.yml |
| Environment Promotion & Rollback | 1 hr | Deploy vs sync, validate/tag, rollback scripts |
| Monitoring & Dashboards | 1 hr | Metrics dashboard, alerts |
| Testing & Final Deployment | 1 hr | Smoke tests, UAT, production go-live |

**7. Implementation Notes**

* **Notebook widgets** ensure ETL is reusable across plants without code duplication.
* **JSON outputs** let Databricks communicate results back to ADF pipelines.
* **Cluster policies** save ~20% cost by enforcing smaller instance sizes in Dev/Test.
* **Fan-out/fan-in** ensures all plants process in parallel, cutting ETL time by 75%.
* **Metrics repo** centralizes DQ checks, making them reusable across datasets.
* **TDD** ensures data transformation correctness before production deployment.
* **Drift monitoring** uses accuracy thresholds to auto-trigger retraining.
* **GitHub Actions** OIDC integration removes the need for stored secrets.
* **bundle.yml** simplifies multi-environment deployment logic.
* **Rollback scripts** ensure recovery from bad deployments in <5 minutes.

**8. Evaluation Rubrics**

| **Criteria** | **Weightage** | **Measurement** |
| --- | --- | --- |
| **Functionality** | 30% | All ETL, ML, and CI/CD workflows execute without errors |
| **Code Reusability** | 15% | Use of widgets, reusable jobs, metrics repo |
| **Performance** | 15% | ETL runtime reduced, parallelism achieved |
| **Data Quality** | 10% | % of DQ checks passed, metrics accuracy |
| **Security Compliance** | 10% | Secrets in scopes, branch protection applied |
| **Deployment Reliability** | 10% | Rollback tested, smoke tests passed |
| **Documentation** | 10% | Architecture diagrams, release notes, runbooks |