

Case Studies

Senior Data Engineer

Two example case studies about Azure data platforms and production data integrations.

HOW TO READ THESE CASE STUDIES

In these case studies, I go through how to approach common data platform challenges. The idea is to show how approach platform modernization and data integrations from design to running in production.

CASE STUDIES

Modernizing a Legacy Data Platform into an Azure Lakehouse

Moving step by step from scattered source systems and old warehouse processes to an Azure lakehouse that is governed and monitored.

Reliable Production Data Integrations

Ingestion and transformation patterns for production: retries, schema changes, data quality and recovering from failures.

MODERNIZING A LEGACY DATA PLATFORM INTO AN AZURE LAKEHOUSE

Representative Scenario

Data is in several operational systems and in an old warehouse. Pipelines are hard to change and the same business logic is implemented in many places. Nobody really sees the data quality, the lineage or why production jobs fail.

Approach

1. List the important sources, data products, owners, access limits and freshness needs. Pick a small but useful part to migrate first.
2. Design a medallion style lakehouse in Azure. Use Databricks and Azure Data Factory where they make sense for the workload.
3. Build ingestion, transformations, data models and automated quality checks with Python, SQL and Spark.
4. Add access control, monitoring, CI/CD and infrastructure automation, so releases and environments can be reproduced.
5. Document who owns what, how to recover from failures and what the next migration steps are.

Outcomes to Measure

- How much time and effort it takes to add a new source or deliver a data product.
- Pipeline reliability, data freshness and how fast failures are noticed and fixed.
- Number of data quality issues and whether data can be traced from source to consumer.
- Running cost and performance compared to a baseline agreed beforehand.

Relevant Capabilities

Azure Databricks, Azure Data Factory, Apache Spark, Python, SQL, lakehouse architecture, data modeling, data quality, Terraform, CI/CD and production operations.

RELIABLE PRODUCTION DATA INTEGRATIONS

Representative Scenario

Business data comes from APIs, databases and files. The ingestion jobs break easily when a source is down, a schema changes or a run has to be repeated. It is also hard to find out what went wrong when there are failures or data quality problems.

Approach

1. Before implementation, agree on the source contracts, owners, how fresh the data needs to be and how fast recovery is needed.
2. Build ingestion with checkpoints and idempotent processing. Then retries and backfills do not create duplicates or skip data without anyone noticing.
3. Handle schema changes and validate data at the pipeline boundaries, so invalid or unexpected records are visible.
4. Add structured logs, metrics, alerts and runbooks. The operator can then see the impact and recover safely.
5. Automate tests and deployments. Choose batch or streaming based on the real latency requirement.

Outcomes to Measure

- Share of successful runs, missing or duplicate records and data freshness compared to the agreed service level.
- Time to detect, find the cause and recover when a source or pipeline fails.
- Effort to add a new source or handle a schema change.
- Share of critical data flows that have automated validation, monitoring and a documented owner.

Relevant Capabilities

Azure Data Factory, Azure Databricks, Python, SQL, Apache Spark, API and database integration, data quality, CI/CD, Terraform, monitoring and production operations.