ONE PLATFORM · EVERY MODERNIZATION PATH

Migrate every legacy ETL workload to the modern stack.

MigryX converts **SAS, Talend, Alteryx, IBM DataStage, Informatica, Oracle ODI, SSIS, Teradata, and 15+ SQL dialects** directly to Databricks, Snowflake, BigQuery, Fabric, Iceberg, and AWS — with 95%+ parser accuracy, 99% with AI augmentation, and column-level lineage from day one.

TARGETS · MODERN PLATFORMS

- Databricks
- Snowflake
- BigQuery
- Fabric
- Iceberg
- AWS / Redshift
- PySpark
- Anaconda
- PyFluent
- dbt

SOURCES · LEGACY ENGINES

- SAS
- SAS DataFlux
- Talend
- Alteryx
- DataStage
- SSIS
- Informatica
- IDMC
- Oracle PL/SQL
- Oracle ODI
- Teradata
- SQL Dialects
- Python
- Polars

PRODUCTS · MIGRYX MODULES

- Compass · Discovery
- Atlas · Lineage
- Merlin AI · Engine

99%	60%+	10×	8+
AI ACCURACY	COST SAVINGS	FASTER	SOURCE ENGINES

Run the Pilot Yourself →

THE PARSER ENGINE

One parser engine. Every legacy source. Every modern target.

Custom AST parsers — purpose-built for each source language, not generic scanners — combined with the Merlin AI engine for ambiguity resolution. The same proven methodology applies to every legacy source landing on every modern target.

Precision Parsing

Custom ASTs per source. SAS macros, DataStage XML, Talend .item, SSIS .dtsx — full fidelity, deterministic output, no approximation.

Merlin AI Augmentation

Resolves macro variables, ambiguous logic, and runtime parameters that line-by-line transpilers silently mistranslate. 95% → 99%.

Air-Gapped Capable

Runs entirely in your network. Zero external API calls. Source code and data never leave your boundary. Deploy in under an hour.

5-STAGE METHODOLOGY

From legacy artifact to production — in five proven steps.

1 Ingest
SAS, Talend, DataStage, .dtsx, ODI repos.

2 Parse + Lineage
ASTs, macro expansion, column-level lineage.

3 Convert
PySpark, Delta, Snowpark, Dataform, Iceberg.

4 Validate
Row-level + aggregate matching legacy ↔ target.

5 Govern
STTM to Unity Catalog, Horizon, Knowledge.

SOURCE COVERAGE

14+ legacy source engines. Every dialect parsed natively.

- SAS Base / EGP / DI
- SAS DataFlux
- Talend
- Alteryx
- IBM DataStage
- SSIS (.dtsx)
- Informatica
- IDMC / IICS
- Oracle PL/SQL
- Oracle ODI
- Teradata BTEQ
- 15+ SQL Dialects
- Python
- Polars
- Mainframe / COBOL
- JCL / PL/1

DATABRICKS.MIGRYX.COM

Land every legacy ETL on the Databricks Lakehouse.

MigryX converts SAS, Talend, Alteryx, DataStage, Informatica, ODI, SSIS, Teradata, and SQL dialects directly to PySpark, Delta Lake, Unity Catalog, DLT pipelines, and Databricks Workflows — with Medallion Architecture and Photon optimization out of the box.

Native Lakehouse Outputs

COMPUTE & SQL

PySpark · Databricks SQL · Photon Engine · Serverless SQL · UDFs · UDTFs · 500+ Function Maps

STORAGE & PIPELINES

Delta Lake · MERGE INTO · Z-ORDER · Liquid Clustering · Auto Loader (CDC) · DLT Pipelines · @dlt.table · @dlt.expect

ORCHESTRATION & GOVERNANCE

Workflows · Asset Bundles (DABs) · Unity Catalog · Column-Level Lineage · Attribute Tags · MLflow · Feature Store

What you get

🥉 **Medallion Architecture · Auto-Generated**
Bronze raw, Silver cleansed, Gold business-ready — auto-routing of ETL logic to the correct layer.

⚡ **Photon-Optimized SQL**
Legacy SQL transpiled to Photon-optimized Databricks SQL on serverless warehouses. Faster queries, lower cost.

🔄 **DLT + Streams Unified**
Streaming and batch unified via declarative @dlt.table with data quality expectations and enhanced autoscaling.

🛡️ **Unity Catalog Native**
STTM, lineage, and data contracts published directly into Unity Catalog with attribute tags from day one.

silver_claims.py · DLT

```python
# SAS DI Studio job → Databricks DLT
import dlt
from pyspark.sql import functions as F

@dlt.table(name="silver_claims")
@dlt.expect_or_drop("valid", "id IS NOT NULL")
def silver_claims():
  return dlt.read("bronze_claims")
```

99%	10×	60%+	Δ
CONVERSION ACCURACY	FASTER THAN MANUAL	COST SAVINGS	DELTA-NATIVE OUTPUT

SNOWFLAKE.MIGRYX.COM

Land every legacy ETL on Snowflake.

MigryX converts legacy ETL directly to Snowpark Python, Dynamic Tables, Streams & Tasks, Snowpipe, and Cortex AI — with Horizon Catalog governance, Iceberg Tables interoperability, and Zero-Copy Cloning out of the box.

Native Snowflake Outputs

COMPUTE & SQL

Snowpark Python · DataFrames · Virtual Warehouses · Snowflake SQL · UDFs · UDTFs · Stored Procedures

PIPELINES & STREAMING

Dynamic Tables · TARGET_LAG · Streams (CDC) · Tasks (DAGs) · Snowpipe · Openflow

STORAGE, GOVERNANCE & AI

Iceberg Tables · Zero-Copy Cloning · Time Travel · Horizon Catalog · Object Tags · Cortex AI / LLM · Snowflake ML

What you get

❄ **Snowpark-Native Execution**
Legacy ETL converted to Snowpark Python DataFrames — pushdown computation runs natively inside Virtual Warehouse.

🔄 **Dynamic Tables · Declarative Refresh**
Incremental pipelines with declarative TARGET_LAG, automatic refresh, and full lineage tracking built in.

🤖 **Cortex AI from Day One**
SAS analytical models land as Snowflake ML with Model Registry, Feature Store, and Cortex LLM functions.

📊 **Horizon Catalog Governance**
STTM, lineage, and data contracts published directly into the Snowflake object catalog with Object TAGs.

gold_claims_dt.sql · Snowflake

```
-- SAS PROC SQL → Dynamic Table
CREATE OR REPLACE DYNAMIC TABLE
  GOLD.CLAIMS_SUMMARY
TARGET_LAG = '5 minutes'
WAREHOUSE = XS_WH
AS SELECT region,
  SUM(amt) AS total_amt
FROM PROD.SILVER.CLAIMS
GROUP BY region;
```

99%	10×	60%+	❄
CONVERSION ACCURACY	FASTER THAN MANUAL	COST SAVINGS	SNOWPARK-NATIVE

BIGQUERY.MIGRYX.COM

Land every legacy ETL on BigQuery.

MigryX converts legacy ETL directly to BigQuery SQL, Dataform SQLX, Dataproc PySpark, Cloud Composer DAGs, BigQuery ML, and Vertex AI — with Knowledge Catalog governance and Gemini-powered AI from day one.

Native Google Cloud Outputs

COMPUTE & SQL

- BigQuery SQL (GoogleSQL)
- Serverless
- Slot Reservations
- UDFs / Remote Functions
- Stored Procedures

PIPELINES & ORCHESTRATION

- Dataform (SQLX)
- Dataproc (PySpark)
- Cloud Dataflow
- Cloud Composer (Airflow)
- Pub/Sub → BQ

STORAGE, GOVERNANCE & AI

- BigLake
- Iceberg Tables
- Time Travel
- Knowledge Catalog
- Policy Tags
- BigQuery ML
- Vertex AI
- Gemini in BigQuery

What you get

📐 **Dataform SQLX · Native Output**
ETL logic converted to Dataform SQLX with refs, tags, and assertions — incremental tables, version control via Git.

⚡ **Heavy Spark on Dataproc**
Compute-intensive ETL converted to Dataproc PySpark with BigLake/Iceberg interoperability for open lakehouse architecture.

🤖 **Gemini-Native AI**
SAS analytical models become BigQuery ML and Vertex AI deployments. Conversational Analytics, AI Functions, Vector Search.

📊 **Knowledge Catalog Governance**
STTM and lineage published into Knowledge Catalog with Policy Tags and DLP enforcement.

silver_claims.sqlx · Dataform

```
-- SAS DI Studio → Dataform
config { type: "table",
  tags: ["daily"] }
SELECT claim_id, region, amt
FROM ${ref("bronze_claims")}
QUALIFY ROW_NUMBER() = 1
```

99%	54%	10×	G
CONVERSION ACCURACY	LOWER TCO	FASTER MIGRATION	GEMINI-POWERED

FABRIC.MIGRYX.COM

Land every legacy ETL on Microsoft Fabric.

MigryX converts legacy ETL directly to Fabric Spark Notebooks, Data Warehouse, Lakehouse, Data Factory pipelines, Real-Time Analytics, Power BI Dataflows, and Fabric AI Copilot — with OneLake catalog governance.

Native Fabric Outputs

COMPUTE & NOTEBOOKS

Spark Notebooks · PySpark on Fabric · Compute Pools · Fabric SQL · UDFs

PIPELINES & STORAGE

Data Factory · Pipelines · Dataflows · Lakehouse · Data Warehouse (T-SQL) · OneLake / Delta · Real-Time Analytics (KQL)

BI, GOVERNANCE & AI

Power BI Dataflows · Semantic Models · OneLake Catalog · Lineage View · Fabric AI · Copilot

What you get

Spark Notebooks Native
Legacy ETL converted to Fabric Spark Notebooks running on Fabric compute pools with seamless OneLake integration.

Data Warehouse + Lakehouse Unified
T-SQL warehouse, Delta-backed lakehouse, KQL streaming — all queryable from one OneLake substrate.

Data Factory Pipelines
Scheduled ETL converted to Fabric Data Factory pipelines — parameterized execution replaces legacy schedulers.

Power BI Direct
Reporting/data-prep logic migrates to Power BI Dataflows Gen2 for self-service analytics with Copilot-assisted modeling.

silver_claims.py · Fabric Spark

```
# SSIS .dtsx → Fabric Spark Notebook
df = spark.read
  .format("delta")
  .load("abfss://lh.../bronze/claims")
df.write
  .format("delta")
  .saveAsTable("lakehouse.silver.claims")
```

99%	85%	60%+	◆
CONVERSION ACCURACY	FASTER DELIVERY	COST SAVINGS	ONELAKE NATIVE

ICEBERG.MIGRYX.COM

Land every legacy ETL on open Iceberg tables.

Open table format running on Spark, Trino, Flink, or Dremio — with schema evolution, hidden partitioning, time travel, ACID transactions, and full column-level lineage. No vendor lock-in. Same tables, queryable from any engine.

Native Iceberg Outputs

ENGINES THAT RUN ICEBERG

PySpark + Iceberg · Trino SQL · Flink Streaming · Dremio · Snowflake Iceberg · Databricks Iceberg

ICEBERG CAPABILITIES

Hidden Partitioning · Schema Evolution · Partition Evolution · Time Travel · Snapshot Isolation · ACID Transactions · CDC via Flink

CATALOG INTEGRATION

Iceberg REST · AWS Glue · Hive Metastore · Nessie · Polaris · Unity Catalog

What you get

🧊 **Open Table Format · No Lock-In**
Output runs on any engine that supports Iceberg — Spark, Trino, Flink, Dremio, Snowflake, Databricks.

🔄 **Schema & Partition Evolution**
Add, rename, drop columns without rewrite. Change partition strategy on the fly. Iceberg tracks everything in metadata.

⏪ **Time Travel · Audit-Ready**
Point-in-time queries via snapshot isolation. Rollback to any previous state. Reproduce historical results.

🤖 **Merlin AI · Partition Optimization**
AI analyzes data patterns to recommend hidden partitioning, sort orders, and compaction policies.

silver_claims.py · Iceberg

```
# SAS DATA step → Iceberg write
df.writeTo(
  "glue.silver.claims")
  .partitionedBy(
    days("event_ts"))
  .tableProperty(
    "format-version", "2")
  .createOrReplace()
```

99%	5+	0	🧊
CONVERSION ACCURACY	ENGINES SUPPORTED	VENDOR LOCK-IN	OPEN FORMAT

AWS.MIGRYX.COM

Land every legacy ETL on AWS analytics services.

MigryX converts legacy ETL directly to Amazon Redshift, AWS Glue PySpark, EMR Spark, Athena SQL, MWAA (Managed Airflow), Lake Formation, and SageMaker — with AWS Glue Data Catalog lineage and Lake Formation policy governance.

Native AWS Outputs

COMPUTE & WAREHOUSE

Redshift Serverless | Redshift Spectrum | Redshift ML | Athena SQL | EMR Spark | EMR Serverless

ETL & ORCHESTRATION

AWS Glue (PySpark) | Glue Studio Visual | Glue DataBrew | MWAA (Airflow) | Step Functions | EventBridge

STORAGE, GOVERNANCE & ML

S3 + Iceberg | Lake Formation | Glue Data Catalog | DataZone | SageMaker | Bedrock

What you get

🔶 **Redshift-Native SQL**
Legacy SQL transpiled to Redshift Serverless or provisioned clusters with 500+ function mappings and Spectrum federation.

⚡ **Glue PySpark Jobs**
Heavy ETL converted to AWS Glue PySpark with Glue Data Catalog registration and dynamic frame transformations.

🛡️ **Lake Formation Governance**
STTM and lineage published into Glue Data Catalog with Lake Formation row/column-level policies and tags.

🤖 **SageMaker + Bedrock Ready**
SAS analytical models migrate to SageMaker Pipelines, Feature Store, Model Registry. Bedrock for generative AI on top.

silver_claims.py · AWS Glue

```
# Informatica → AWS Glue PySpark
from awsglue.context import GlueContext
gc = GlueContext(SparkContext.getOrCreate())

df = gc.create_dynamic_frame
  .from_catalog(
    database="prod",
    table_name="bronze_claims")
```

99%	10×	60%+	☁
CONVERSION ACCURACY	FASTER MIGRATION	COST SAVINGS	VPC-NATIVE DEPLOY

PYSPARK.MIGRYX.COM

PySpark · Distributed compute target.

MigryX generates production-grade PySpark — DataFrame APIs, Spark SQL, structured streaming, broadcast joins — runnable on Databricks, EMR, Dataproc, Fabric, or any Spark cluster.

WHAT MIGRYX GENERATES

- ✓ DataFrame transformations from DATA steps and tMap joins
- ✓ Spark SQL from PROC SQL, BTEQ, T-SQL, PL/SQL
- ✓ UDFs / pandas UDFs for custom logic and macros
- ✓ Structured Streaming for CDC and near-real-time
- ✓ Broadcast joins, partitioning, AQE optimization hints

RUNS ON

- ✓ Databricks · EMR · Dataproc
- ✓ Fabric Spark · Glue · Synapse
- ✓ Cloudera CDP · OpenShift
- ✓ Self-managed Spark on Kubernetes

ANACONDA.MIGRYX.COM

Anaconda · Enterprise Python distribution.

For shops standardizing on Anaconda for governed package management, MigryX outputs Python that's reproducible across air-gapped enterprise environments — pinned envs, conda recipes, secure repositories.

WHAT MIGRYX GENERATES

- ✓ Pinned conda environment.yml per project
- ✓ pandas / numpy / scipy code from PROC steps
- ✓ scikit-learn / statsmodels from SAS/STAT
- ✓ Jupyter notebooks with inline lineage
- ✓ Reproducible builds for air-gapped deployment

WHY IT MATTERS FOR SAS SHOPS

- ✓ Governed package mirror — no public PyPI calls
- ✓ Compliance-ready environment provenance
- ✓ Anaconda Server / Repository integration
- ✓ Migrate SAS analytics to data-science teams safely

PYFLUENT.MIGRYX.COM

PyFluent · Engine-agnostic Python framework.

MigryX's runtime layer — write once, run anywhere. PyFluent converts SAS, Talend, DataStage, Informatica, ODI, Teradata, Oracle, and PL/1 into a single fluent Python API portable across Snowpark, PySpark, BigQuery, and Polars.

ONE API, EVERY BACKEND

- ✓ Snowpark · PySpark · BigQuery DataFrames
- ✓ Polars · pandas · DuckDB local execution
- ✓ Switch backends without code rewrites
- ✓ Identical semantics across compute engines

WHY IT MATTERS

- ✓ Hedge against single-vendor lock-in
- ✓ Develop locally on Polars, deploy to Snowpark
- ✓ Auto-tune for the chosen target's optimizer
- ✓ Future-proof migrations as targets evolve

DBT.MIGRYX.COM

dbt · SQL-first transformation framework.

MigryX generates dbt models, sources, exposures, and tests from legacy ETL — PROC SQL, Informatica mappings, DataStage transformers — directly into Git-versioned dbt projects.

WHAT MIGRYX GENERATES

- ✓ dbt models · sources · exposures · tests
- ✓ schema.yml with column-level documentation
- ✓ Jinja macros from SAS macros
- ✓ Refs and dependencies preserved as DAG
- ✓ Materialization strategies (table/view/incremental)

ADAPTERS SUPPORTED

- ✓ dbt-databricks · dbt-snowflake · dbt-bigquery
- ✓ dbt-redshift · dbt-fabric · dbt-spark
- ✓ dbt Cloud · dbt Core · CI/CD ready

SAS.MIGRYX.COM · SAS2PY.COM

SAS · Modernize the entire ecosystem.

Purpose-built parsers for Base SAS, Enterprise Guide (.egp), DI Studio, SAS Viya/CAS, and the SAS macro language. 95%+ deterministic accuracy. Macro expansion, format resolution, PROC translation included.

WHAT MIGRYX PARSES

- ✓ Base SAS · DATA steps · SET/MERGE · PROC SQL
- ✓ SAS Macros · %INCLUDE · Formats / Informats
- ✓ Enterprise Guide (.egp) · DI Studio metadata
- ✓ SAS Viya · CAS engine · Stored Processes
- ✓ SAS/IML · SAS/STAT analytical procs

SAS → MODERN

- ✓ DATA step → PySpark / pandas DataFrame
- ✓ PROC SQL → Databricks / Snowflake / BQ SQL
- ✓ %macro → Python functions · Jinja templates
- ✓ DI Studio → Workflows / Tasks / Composer
- ✓ CAS → MLflow / SageMaker / Vertex AI

DATAFLUX.MIGRYX.COM

SAS DataFlux · Data-quality jobs modernized.

Migrate SAS DataFlux dfPower Studio jobs and DQ schemes — standardize/parse/match/validate patterns — to PySpark UDFs, Databricks Lakehouse Monitoring, Snowflake data quality functions, or Fabric AI anomaly detection.

WHAT MIGRYX PARSES

- ✓ dfPower Studio jobs · DMS schemes
- ✓ Standardize / Parse / Match / Validate steps
- ✓ Address standardization rules
- ✓ Customer matching logic and weights
- ✓ Reference data and lookup tables

DATAFLUX → MODERN

- ✓ PySpark UDFs for parse/match/validate
- ✓ Databricks Lakehouse Monitoring rules
- ✓ Snowflake DQ functions + Cortex anomaly
- ✓ BigQuery data profiling + Knowledge Catalog
- ✓ Fabric AI anomaly detection

TALEND.MIGRYX.COM

Talend · Studio · tMap · Cloud.

Parse Talend project exports (ZIP/Git), .item and .properties artifacts, Standard and Big Data jobs, tMap joins, metadata, contexts, and connections — converted to PySpark, Databricks Workflows, Snowpark, Dataform, or Fabric pipelines with full component-level lineage.

WHAT MIGRYX PARSES

- ✓ .item / .properties artifacts (ZIP / Git)
- ✓ Standard Jobs · Big Data Jobs · Routes
- ✓ tMap joins · expressions · variables
- ✓ Metadata repository · contexts · connections
- ✓ Routines (Java) · custom components

TARGETS

- ✓ PySpark · Databricks Workflows + DABs
- ✓ Snowpark · Snowflake Tasks + Streams
- ✓ Dataform · Cloud Composer · Fabric pipelines
- ✓ Component-level lineage to Unity Catalog

ALTERYX.MIGRYX.COM

Alteryx · Designer · Macros · Apps.

Convert Alteryx Designer workflows (.yxmd / .yxwz), macros, and analytic apps to PySpark, Snowpark, Databricks SQL, BigQuery, or Iceberg — with tool-by-tool translation and full lineage preservation.

WHAT MIGRYX PARSES

- ✓ .yxmd / .yxwz workflow files
- ✓ Standard, batch, iterative, location macros
- ✓ Analytic apps with input questions
- ✓ Custom Python / R tools
- ✓ Predictive tools (linear/logistic/forest)

TARGETS

- ✓ PySpark · Snowpark · Databricks notebooks
- ✓ Predictive tools → MLflow / SageMaker / BQ ML
- ✓ Macros → Python functions / dbt macros
- ✓ Tool-by-tool lineage with audit trail

DATASTAGE.MIGRYX.COM

IBM DataStage · Parallel · Server · DataStage X.

Migrate DataStage parallel and server jobs, sequences, shared containers, and XML/DSX/ISX definitions to PySpark, Databricks DLT, Snowflake Tasks, Dataform, or Fabric Spark Notebooks — transformer logic fully preserved with column-level lineage.

WHAT MIGRYX PARSES

- ✓ Parallel jobs · Server jobs · Sequences
- ✓ Shared containers · Local containers
- ✓ DSX / ISX / XML exports
- ✓ Transformer stages · Routines · BASIC code
- ✓ Parameter sets · Job activities

TARGETS

- ✓ PySpark · Databricks DLT + Workflows
- ✓ Snowpark · Snowflake Streams & Tasks
- ✓ Dataform · Dataproc · Cloud Composer
- ✓ Fabric Spark Notebooks · Data Factory

SSIS.MIGRYX.COM

SSIS · .dtsx · .ispac · Script Tasks.

Parse SSIS .dtsx packages and .ispac archives — data flow, control flow, SSIS expressions, and C#/VB.NET script tasks — to PySpark, Databricks Workflows, Synapse Pipelines, or Fabric Data Factory with Lakehouse ingestion.

WHAT MIGRYX PARSES

- ✓ .dtsx packages · .ispac project archives
- ✓ Control flow · data flow · sequences
- ✓ SSIS expressions and variables
- ✓ C# / VB.NET Script Tasks
- ✓ Connection managers · package config

TARGETS

- ✓ PySpark · Databricks Workflows + DLT
- ✓ Fabric Data Factory pipelines + Lakehouse
- ✓ Synapse Pipelines · Azure Data Factory
- ✓ Snowpark · Cloud Composer DAGs

INFORMATICA.MIGRYX.COM

Informatica PowerCenter · .xml mappings.

Migrate Informatica PowerCenter XML exports — sources, targets, transformations, sessions, workflows, and worklets — to PySpark, Databricks DLT, Snowpark, Dataform, or Fabric Spark with Unity Catalog / Horizon Catalog lineage registration.

WHAT MIGRYX PARSES

- ✓ PowerCenter .xml repository exports
- ✓ Sources · Targets · Transformations
- ✓ Mappings · Mapplets · Worklets
- ✓ Sessions · Workflows · parameter files
- ✓ Custom transformations · Java transformations

TARGETS

- ✓ PySpark · Databricks DLT + Asset Bundles
- ✓ Snowpark · Dynamic Tables · Streams & Tasks
- ✓ Dataform · Dataproc · Cloud Composer
- ✓ Fabric Spark · Data Factory pipelines

IDMC.MIGRYX.COM

Informatica IDMC / IICS · Cloud-native mappings.

Migrate Informatica Intelligent Data Management Cloud (IDMC) and IICS — Cloud Data Integration mappings, Application Integration processes, Mass Ingestion — to modern cloud-native equivalents.

WHAT MIGRYX PARSES

- ✓ IDMC / IICS asset exports (REST API)
- ✓ Cloud Data Integration (CDI) mappings
- ✓ Application Integration (CAI) processes
- ✓ Mass Ingestion · Data Profiling
- ✓ Connections · runtime environments

TARGETS

- ✓ Snowpark · Dynamic Tables / Streams
- ✓ Databricks DLT + Workflows
- ✓ Dataform · Cloud Composer · Fabric Pipelines
- ✓ Drop-in equivalents for all CDI transforms

ORACLE.MIGRYX.COM

Oracle PL/SQL · Procedures · Packages · Triggers.

Migrate Oracle PL/SQL procedures, packages, functions, and triggers with 2,000+ function mappings — CONNECT BY rewriting to recursive CTEs, BULK COLLECT to PySpark batching, MERGE/UPSERT translation, and exception handling preserved.

WHAT MIGRYX PARSES

- ✓ Procedures · Packages · Functions
- ✓ Triggers · Sequences · Synonyms
- ✓ CONNECT BY hierarchies · MODEL clause
- ✓ BULK COLLECT / FORALL bulk patterns
- ✓ Exception handling · pragma directives

TARGETS

- ✓ Databricks SQL · Snowflake SQL · BigQuery
- ✓ PySpark for procedural logic
- ✓ CONNECT BY → recursive CTE
- ✓ Stored Procs preserved across targets

ODI.MIGRYX.COM

Oracle ODI · Mappings · KMs · Load Plans.

Parse Oracle Data Integrator repository exports — interfaces, mappings, knowledge modules (LKMs/IKMs/CKMs), packages, and load plans — converted to PySpark, Databricks DLT, Snowpark, Dataform, or Fabric pipelines with full column-level lineage.

WHAT MIGRYX PARSES

- ✓ ODI repository exports (.xml)
- ✓ Mappings / Interfaces · Reusable Mappings
- ✓ Knowledge Modules (LKM · IKM · CKM · JKM)
- ✓ Packages · Load Plans · Scenarios
- ✓ Variables · Sequences · User Functions

TARGETS

- ✓ PySpark · Databricks DLT + Workflows
- ✓ Snowpark · Streams & Tasks · Dynamic Tables
- ✓ Dataform · Dataproc · Cloud Composer
- ✓ Fabric Spark Notebooks · Data Factory

TERADATA.MIGRYX.COM

Teradata · BTEQ · FastLoad · MultiLoad · QUALIFY.

Migrate Teradata BTEQ scripts, FastLoad / MultiLoad / TPump utilities, and Teradata SQL — QUALIFY rewriting, BTEQ command translation, PRIMARY INDEX advisory, OREPLACE/CSUM patterns — to Databricks, Snowflake, BigQuery, or Iceberg.

WHAT MIGRYX PARSES

- ✓ BTEQ scripts · TDLOAD utilities
- ✓ FastLoad / MultiLoad / TPump / FastExport
- ✓ QUALIFY · QUALIFY ROW_NUMBER
- ✓ Recursive views · macros · triggers
- ✓ PRIMARY INDEX · join indexes · stats

TARGETS

- ✓ Databricks SQL · Snowflake SQL · BigQuery
- ✓ QUALIFY → window function rewriting
- ✓ FastLoad → COPY INTO / Auto Loader / Snowpipe
- ✓ Index hints → Z-ORDER / clustering keys

SQL.MIGRYX.COM

SQL Dialects · 15+ dialects · 500+ function mappings.

Transpile SQL from Oracle, T-SQL, Teradata, DB2, Netezza, Greenplum, Hive HQL, Vertica, Sybase, Redshift, SAP HANA, and more — to Databricks SQL, Snowflake SQL, BigQuery SQL, Trino, or Fabric Warehouse SQL.

SOURCE DIALECTS SUPPORTED

- ✓ Oracle · T-SQL · PL/pgSQL · DB2
- ✓ Teradata · Netezza · Greenplum · Vertica
- ✓ Hive HQL · Spark SQL · SAP HANA · Sybase
- ✓ Redshift · MySQL · PostgreSQL

TRANSLATION FEATURES

- ✓ 500+ function mappings curated per pair
- ✓ Window function normalization
- ✓ Stored procs · UDFs · views · CTEs
- ✓ Vendor extensions handled natively

PYTHON.MIGRYX.COM

Python · Already-Python? Refactor to scale.

When the source is already Python — pandas notebooks, scripts, ad-hoc analytical code — MigryX refactors it for scale: pandas → PySpark or Snowpark, single-machine to distributed, untested to test-covered, ungoverned to lineage-tracked.

WHAT MIGRYX PARSES

- ✓ Python scripts · .py modules
- ✓ Jupyter notebooks (.ipynb)
- ✓ pandas · numpy · scipy code
- ✓ scikit-learn / statsmodels / xgboost
- ✓ Custom analytical libraries

REFACTORING OUTCOMES

- ✓ pandas → PySpark / Snowpark / BQ DataFrames
- ✓ scikit-learn → MLflow / SageMaker / BQ ML
- ✓ Notebooks → Databricks / Fabric / Vertex
- ✓ Column-level lineage extracted automatically

POLARS.MIGRYX.COM

Polars · Local-first DataFrame target.

Polars is a high-performance Rust-backed DataFrame library. MigryX targets Polars as a fast local development backend — for prototyping migrations, single-node analytical pipelines, or as a portable PyFluent backend.

WHY POLARS AS TARGET

- ✓ 10–100× faster than pandas on single node
- ✓ Lazy evaluation with query optimizer
- ✓ Local development → cloud production handoff
- ✓ Identical API via PyFluent across backends

WHAT MIGRYX GENERATES

- ✓ Polars expressions from DATA steps
- ✓ LazyFrame pipelines from PROC SQL
- ✓ Streaming queries with sink chains
- ✓ Same logic that runs on Snowpark / PySpark

COMPASS.MIGRYX.COM

Understand your estate before you migrate.

Compass scans your entire legacy environment — every file, program, and dependency — then classifies each asset as MIGRATE, ARCHIVE, or DELETE. Multi-factor scoring evaluates usage, complexity, recency, and risk. Convert only what matters. Archive the rest. Delete the noise.

Classification Engine

INCREASES MIGRATION PRIORITY

- Recent access (≤6 months)
- High frequency
- Small size
- SQL-heavy
- Low complexity
- Error-free

DECREASES MIGRATION PRIORITY

- Stale (2+ yrs) → ARCHIVE
- Never used → DELETE
- Large files
- High complexity
- Execution errors
- Orphaned

OUTPUTS

- Inventory + Catalog
- Dependency Graphs
- Risk Scoring
- Migration Roadmap
- Cost Projections
- Phased Plan
- Executive Dashboards

Typical Customer Outcomes

📦 **60–80% Data Reduction**
Most enterprise SAS estates contain massive archive- and delete-candidates. Compass surfaces them automatically with evidence-based scoring.

⚡ **5–10× Faster Analysis**
High-performance parallel processing. Hashing detects duplicates. Hours to complete a scan — not weeks of manual assessment.

🎯 **100% Asset Coverage**
Every file scanned. Every program parsed. Every execution log analyzed. Complete inventory visibility before any decision.

📊 **Phased Migration Plan**
Priority-based phase assignment optimized for risk and business continuity. Phase 1 (Critical), Phase 2–3, Archive, Delete.

CLASSIFICATION OUTPUT

- 5 · **MIGRATE Critical** — Phase 1 · daily reports, SQL-heavy
- 3–4 · **MIGRATE Standard** — Phase 2–3 · regular usage
- A · **ARCHIVE** — cold storage · reduce licensing
- D · **DELETE** — duplicates, orphans, never executed

60-80%	5-10×	100%	Hours
DATA REDUCTION	ANALYSIS SPEED	ASSET COVERAGE	NOT WEEKS

ATLAS.MIGRYX.COM

Map your entire data ecosystem. Column by column.

Atlas delivers column-level lineage and Source-to-Target Mapping (STTM) across SAS, Python, PySpark, R, Polars, SQL dialects, Informatica, Talend, Alteryx, DataStage, SSIS, ODI, and 30+ languages — unified into a single lineage graph.

What Atlas Parses

PROGRAMMING & ANALYTICS

SAS Base / Macros · Python · pandas · PySpark · R · Polars

SQL DIALECTS (15+)

Snowflake SQL · Databricks SQL · BigQuery SQL · T-SQL · PL/SQL · Teradata BTEQ · DB2 · Netezza · Hive HQL · Redshift · Vertica · SAP HANA

ETL & BI TOOLS

Informatica · IICS · Talend · Alteryx · IBM DataStage · SSIS · Oracle ODI · SAS DataFlux · dbt · ADF · Glue · Airflow

What you get

🧬 **Column-Level Traceability**
Every column traced from source to target with transformation logic, operation type, and module reference — deterministic, not approximate.

🌐 **Cross-Platform Unified Graph**
Informatica → Databricks → BigQuery — Atlas maps every hop. One graph spans your entire enterprise.

📋 **Audit-Ready STTM Export**
Export lineage to CSV, JSON, Excel, and interactive HTML. GDPR, SOX, BCBS 239 audit reports in minutes.

🔍 **Visual Lineage Explorer**
Click any column to see everything that feeds it and everything it feeds. Impact analysis across the entire estate.

PLUS: MAINFRAME & STORED OBJECTS

JCL · COBOL · PL/1 · RPG/AS400 · DDL · Views · UDFs · Triggers · Stored Procedures

30+	15+	100%	1
LANGUAGES PARSED	SQL DIALECTS	COLUMN-LEVEL	UNIFIED GRAPH

MERLINAI.MIGRYX.COM

Custom parsers get you to 95%. Merlin AI gets you to 99%.

Merlin AI is the domain-trained intelligence layer that resolves what deterministic parsers cannot: ambiguous macro variables, runtime parameters, undocumented transformations, and business intent buried in legacy code. Trained on billions of lines of legacy ETL and analytics code.

Merlin Capabilities

RESOLUTION & INFERENCE

- Macro variable resolution
- Runtime parameter inference
- Implicit join detection
- Business intent extraction
- Schema inference
- Type promotion

AUTO-DOCUMENTATION

- Inline code comments
- Module summaries
- Data contracts
- STTM enrichment
- Test scaffolds
- Mapping suggestions

OPTIMIZATION & Q&A

- Partitioning recommendations
- Clustering hints
- Z-ORDER candidates
- Warehouse sizing
- Conversational Q&A
- Diff explanation

What you get

Context-Aware Assistance
Merlin knows your inventory, lineage, and conversion plans. Generate unit tests, explain diffs, suggest mappings, draft notebooks with your rules applied.

95% → 99% Accuracy
Parsers handle deterministic conversions. Merlin resolves the remaining ambiguity — macro chains, env-dependent logic, business intent.

Runs in Your Environment
Merlin runs locally inside the MigryX container — air-gapped capable. No code or data leaves your network. SOC 2 audit trails included.

Continuous Learning
Improves as it processes more of your codebase, adapting to your organization's unique patterns, naming conventions, and macro libraries.

merlin_resolved.py

```
# Source: SAS PROC SQL with macro
# &env_schema..claims → ?
# Merlin resolved: env_schema = "PROD"
# Inferred: implicit equijoin on a.id
SELECT a.*, b.region
FROM PROD.claims a
INNER JOIN PROD.regions b
  ON a.id = b.id
```

95 → 99%	B+	In-VPC	SOC 2
ACCURACY LIFT	LINES TRAINED ON	LOCAL INFERENCE	AUDIT TRAILS

Stop rewriting legacy.
Start running on modern platforms.

Join the enterprises that have already landed millions of lines of SAS, Talend, DataStage, Informatica, ODI, and SSIS code on Databricks, Snowflake, BigQuery, Fabric, Iceberg, and AWS — in weeks, not years. Self-service pilots run entirely in your environment.

Run the Pilot Yourself

MIGRATION READINESS

Discovery & Insights

1 wk DURATION | **100K** LOC DISCOVERY

- ✓ Inventory + dependency mapping
- ✓ Visual lineage + risk scoring
- ✓ Self-service · runs in your env

FULL PILOT · END-TO-END

Convert · Execute · Validate

4–6 wk DURATION | **10K** LOC CONVERSION

- ✓ Discovery + pilot conversion
- ✓ Data matching + validation
- ✓ Enterprise data workflows

LARGE SCALE PILOT

Enterprise Migration

2–4 mo DURATION | **100K** LOC CONVERSION

- ✓ 1M LoC discovery scope
- ✓ Full project + JCL reports
- ✓ Production execution + cutover

Deployment · On-Premise · Air-Gapped

No outbound connections. Zero external API calls. Source code and data stay entirely on your infrastructure. Deploy in under an hour.

DOCKER QUICK START · 8 CORES · 32 GB · 100 GB

```
docker run -d \
  -p 1443:1443 -p 3174:3174 \
  -p 3177:3177 -p 8080:8080 \
  --name migryx \
  -e PASSWORD="StrongPassword" \
  migryx:3.2.1
```

CLOUD OPTIONS

AWS	EC2 m5.2xlarge · VPC · S3 · KMS
Azure	D8s v3 · VNet · ADLS · Key Vault
GCP	n2-standard-8 · VPC · GCS · KMS
K8s	Kubernetes / OpenShift
Windows	VM with same hardware profile

See your code converted — live, in 30 minutes.

No slides. No generic demos. Send a sample of SAS, Talend, DataStage, Informatica, ODI, SSIS, or BTEQ — we'll convert it, deploy it, and return column-level lineage. Free, no commitment, runs entirely in your environment.

Schedule a Demo →

migryx.com | **hello@migryx.com** | **(617) 512-9530**