Solution Accelerator

1️⃣ What is Lakebridge?

Lakebridge is an **open-source migration and modernization framework by Databricks Labs**.

Its main purpose is to:

Assess, analyze, convert (transpile), and validate legacy SQL / data workloads into Databricks Lakehouse (Spark SQL / Databricks SQL)

Think of Lakebridge as:

🌉 **A bridge between legacy data platforms and Databricks Lakehouse**

2️⃣ Why Lakebridge Exists (Problem Statement)

Most enterprises still run data workloads on:

- SQL Server
- Oracle
- Teradata
- Netezza
- Hive
- On-prem data warehouses

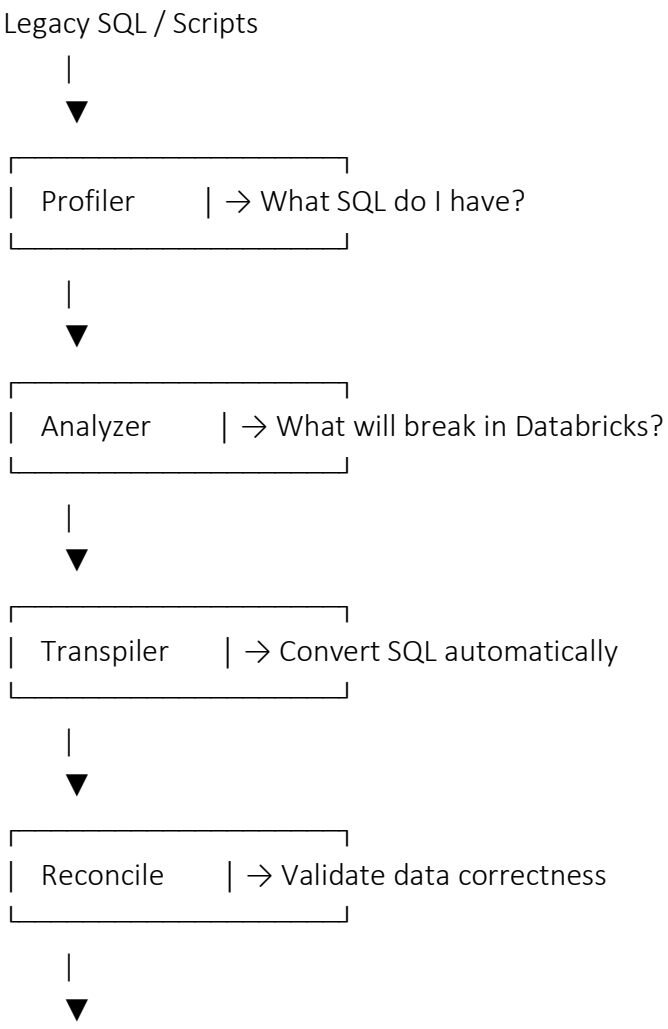
Common challenges:

- Thousands of stored procedures
- Vendor-specific SQL syntax
- Complex procedural logic
- Manual migration is slow and risky
- No clear impact analysis

Lakebridge solves this by:

- Automatically **profiling SQL**
- Identifying **conversion gaps**
- **Transpiling SQL** to Databricks-compatible SQL
- **Reconciling results** between source & target

3 Lakebridge High-Level Architecture



4️⃣ Key Components of Lakebridge

Lakebridge is **not a single tool**.

It is a **framework with multiple modules**.

5️⃣ Installation Overview

Lakebridge is:

- Python-based
- CLI-driven
- Runs locally or in Databricks

Prerequisites:

- Python 3.9+
- pip
- Git
- Access to Databricks workspace

Installation (high-level):

pip install lakebridge

Or:

git clone https://github.com/databrickslabs/lakebridge

6️⃣ Assessment Guide (Very Important Phase)

🔍 What is Assessment?

Assessment answers:

- How big is my migration?
- How complex is my SQL?
- What percentage can be auto-converted?

6.1 Profiler Guide

Purpose:

The **Profiler** scans your SQL assets and creates metadata.

What it profiles:

- Number of SQL files
- Stored procedures
- Views
- Functions
- SQL statements
- DDL vs DML
- Control flow usage (IF, WHILE, CURSOR)
- Vendor-specific syntax

Input:

- Folder containing SQL files

Output:

- JSON / CSV reports
- SQL complexity summary

Example Output:

Metric	Count
Stored Procedures	350
Views	120
Temp Tables	480

Cursors	45
Dynamic SQL	62

👉 This helps management estimate **effort, cost, and risk**.

6.2 Analyzer Guide

Purpose:

The **Analyzer** determines **what will or will not work in Databricks**.

It checks:

- Unsupported SQL syntax
- Vendor-specific functions
- Procedural constructs
- Cursor usage
- Temporary tables
- Transactions (BEGIN TRAN, COMMIT)

Example Findings:

- ❌ CURSOR not supported
- ❌ TRY…CATCH logic incompatible
- ⚠️ UNPIVOT needs rewrite
- ✅ CASE WHEN supported

Output:

- Compatibility score
- Refactoring recommendations
- Manual effort estimation

7️⃣ Transpile Guide (Core Feature)

🔁 What is Transpiling?

Transpiling = Automatically converting:

SQL Server / Oracle SQL
↓
Databricks SQL / Spark SQL

7.1 What Transpiler Can Convert Automatically

✅ SELECT statements

✅ JOINs

✅ CASE WHEN

✅ Aggregations

✅ DATE functions (partial)

✅ Basic temp table logic

7.2 What Needs Manual Work

❌ Cursors

❌ WHILE loops

❌ Dynamic SQL (EXEC(@sql))

❌ Complex procedural logic

❌ Exception handling

Lakebridge will:

- Convert what it can
- Add **comments** where manual rewrite is required

Example Conversion

SQL Server

```
SELECT ISNULL(amount, 0) FROM sales;
```

Databricks SQL

```
SELECT COALESCE(amount, 0) FROM sales;
```

8️⃣ SQL Splitter

What is SQL Splitter?

Many legacy SQL files contain:

- Multiple procedures
- Mixed DDL & DML
- Scripts without separators

SQL Splitter:

- Breaks large SQL scripts into logical units
- Makes profiling & transpiling easier

9️⃣ Reconcile Guide (Validation Phase)

🎯 Purpose of Reconciliation

Ensures:

Databricks output matches legacy system output

What Reconciliation Checks:

- Row count
- Column count
- Data mismatches
- Aggregation differences

Typical Use Case:

1. Run SQL on source system
2. Run transpiled SQL on Databricks
3. Compare outputs

🔟 FAQs (Common Questions)

Q1. Is Lakebridge 100% automatic?

❌ No.

It accelerates migration but **does not eliminate manual refactoring**.

Q2. Is Lakebridge Databricks-only?

✅ Yes.

It targets **Databricks Lakehouse**.

Q3. Can it migrate stored procedures fully?

⚠️ Partial.

Complex procedural logic must be redesigned using:

- CTEs
- Window functions
- Spark jobs

1️⃣1️⃣ Where Lakebridge Fits in a Real Migration Project

Typical Project Flow:

1. Collect SQL assets
2. Run **Profiler**
3. Review **Analyzer** reports
4. Auto-convert using **Transpiler**
5. Manual refactoring
6. Validate using **Reconcile**
7. Production deployment

1️⃣2️⃣ How Lakebridge Helps You Specifically (Based on Your Work)

Given you:

- Convert **SQL Server → Databricks SQL**
- Avoid UNPIVOT, use manual UNION
- Work on **IRRBB, loans, sanction, banking reports**

Lakebridge helps by:

- Quickly identifying **unsupported SQL**
- Reducing manual conversion effort

- Providing **structured migration documentation**
- Acting as a **baseline converter**, not final code

You still:

- Optimize CTEs
- Refactor procedural logic
- Tune performance

1️⃣3️⃣ Summary (One-Paragraph)

Lakebridge is a Databricks Labs framework that accelerates SQL and data warehouse migrations by profiling legacy SQL, analyzing compatibility, automatically converting supported syntax into Databricks SQL, and validating results through reconciliation. It is not a magic converter but a structured migration accelerator designed for large enterprise workloads

📦 Lakebridge Installation – Easy Step-by-Step Guide

STEP 1️⃣ – Databricks Workspace Setup

1.1 Get Databricks Workspace Access

You need **any one** of the following:

- ✅ **Production / Enterprise Workspace** (recommended for real projects)
- ✅ **Development Workspace** (for testing)
- ✅ **Free Databricks Workspace** (best for learning)

👉 Free workspace link:

https://databricks.com/try-databricks

💡 **Pro Tip**

For learning or POC, a **free workspace + Personal Access Token (PAT)** is the fastest way. No approvals needed.

STEP 2 – Install Databricks CLI

Lakebridge runs using **Databricks CLI**, so this is mandatory.

2.1 Install Databricks CLI

Follow the official guide based on your OS:

- Windows
- macOS
- Linux

(Official documentation link provided by Databricks)

2.2 Configure Databricks CLI

Step 2.2.1 – Generate Personal Access Token (PAT)

1. Log in to Databricks
2. Go to **User Settings**
3. Open **Developer**
4. Create **Access Token**
5. Copy and save it securely

Step 2.2.2 – Configure CLI with Cluster ID

Lakebridge **requires a cluster_id** in the CLI profile.

Run:

databricks configure --host <workspace-url> --configure-cluster --profile <profile_name>

👉 You will be prompted to:

- Paste PAT
- Select a cluster from the list

Alternative: Using Environment Variable (Optional)

export DATABRICKS_CLUSTER_ID=<cluster_id>
databricks configure --host <workspace-url> --profile <profile_name>

2.3 Verify Databricks CLI

Run:

databricks clusters list

✅ If cluster list appears → CLI is configured correctly

STEP 3 – Software Requirements

3.1 Python Installation

- ✅ Required Version: **Python 3.10 to 3.13**
- ❌ Python 3.14 is NOT supported

Install Python

- Windows → install from python.org
- macOS/Linux → install using brew

Verify:

python --version

3.2 Java Installation

- ✅ Required Version: **Java 11 or higher**
- Recommended: **OpenJDK 11+**

Verify:

```
java -version
```

👉 Java is required for the **Morpheus transpiler engine**

STEP 4️⃣ – Network & Internet Access

Ensure your machine can access:

Resource	Purpose
github.com	Download Lakebridge
pypi.org	Python dependencies
central.sonatype.com	Transpiler plugins

🔐 For Restricted / Corporate Environments

Ask IT/Security team to whitelist:

- GitHub: github.com, raw.githubusercontent.com
- Maven: repo1.maven.org, central.sonatype.com
- PyPI: pypi.org, files.pythonhosted.org
- Python downloads: python.org
- Java downloads: OpenJDK / Oracle sites

(Optional) Setup **Artifactory / Nexus** for internal hosting.

STEP 5 – Pre-Installation Checklist

Before proceeding, confirm:

☐ Databricks workspace access

☐ Databricks CLI installed

☐ CLI configured with PAT

☐ databricks clusters list works

☐ Python 3.10+ installed

☐ Java 11+ installed

☐ Internet / repository access available

STEP 6 – Install Lakebridge

Run:

databricks labs install lakebridge

Install Using Specific Profile (Optional)

databricks labs install lakebridge --profile <profile_name>

Check profiles:

databricks auth profiles

6.1 Verify Lakebridge Installation

Run:

databricks labs lakebridge --help

✅ If you see commands like analyze, transpile, reconcile → Installation successful

STEP 7 – Install Transpiler

The transpiler converts legacy SQL to Databricks SQL.

Run:

databricks labs lakebridge install-transpile

👉 This will:

- Install default transpilers
- Ask configuration questions once
- Save config automatically

7.1 (Optional) Override Default Transpiler Config

If you want custom conversion rules:

Provide path when prompted:

<local_path>/custom_<source>2databricks.json

⚠️ This can only be done **during installation**

7.2 Verify Transpiler Installation

Run:

databricks labs lakebridge transpile --help

✅ If transpile options appear → Transpiler is ready

STEP 8 – Configure Reconcile Module

Reconcile is used to **compare source vs Databricks data**.

Run:

databricks labs lakebridge configure-reconcile

8.1 SQL Warehouse Requirement

- Lakebridge **creates a SQL warehouse by default**
- If you **don't have permission**, use an existing warehouse

Update ~/.databrickscfg:

[profile-name]
host = <workspace-url>
warehouse_id = <existing-warehouse-id>

8.2 Verify Reconcile Configuration

Run:

databricks labs lakebridge reconcile --help

✅ If help output appears → Reconcile configured successfully

🚀 How to Use Lakebridge – Step-by-Step (With Commands)

This section explains **what to do after installation**, **which commands to run**, and **why you are running them**.

🔁 Lakebridge Usage Flow (Very Important)

Always follow this order:

1. Analyze → Understand your SQL
2. Transpile → Convert SQL to Databricks
3. Reconcile → Validate data

STEP 1️⃣ – Prepare Your Input SQL Files

What you need

Create a folder containing all your legacy SQL files.

📁 Example:

```
lakebridge-input/
  ├── irrbb_cashflow.sql
  ├── loan_account_report.sql
  ├── sanction_module.sql
```

✓ SQL can be from:

- SQL Server
- Oracle
- Teradata
- Synapse
- Datastage

STEP 2️⃣ – Analyze SQL (Profiler + Analyzer)

Purpose

- Identify **what SQL you have**
- Detect **unsupported syntax**
- Estimate **manual conversion effort**

☑️ Command: Analyze

```
databricks labs lakebridge analyze \
  --input-source ./lakebridge-input \
  --source-dialect sqlserver
```

What this command does

Option	Meaning
--input-source	Folder with SQL files
--source-dialect	Source database type

Common dialects:

- sqlserver
- oracle
- teradata
- synapse

Output Generated

- SQL inventory
- Compatibility issues
- Migration readiness

📁 Output location:

.lakebridge/analyze/

STEP 3️⃣ – Transpile (Convert SQL to Databricks)

Purpose

Convert supported SQL automatically into **Databricks SQL**.

☑️ Command: Transpile

```
databricks labs lakebridge transpile \
  --input-source ./lakebridge-input \
  --output-folder ./lakebridge-output \
  --source-dialect sqlserver \
  --target-technology databricks
```

What this command does

- Converts SQL syntax
- Replaces unsupported functions
- Adds comments where manual changes are needed

📁 Output folder:

```
lakebridge-output/
  ├── irrbb_cashflow.sql
  ├── loan_account_report.sql
  ├── sanction_module.sql
```

Example Conversion

Before (SQL Server)

```
SELECT ISNULL(balance,0) FROM loans;
```

After (Databricks SQL)

```
SELECT COALESCE(balance,0) FROM loans;
```

STEP 4 – Manual Refactoring (Mandatory)

Lakebridge **does not fully convert** everything.

You must manually refactor:

- ❌ Cursors
- ❌ WHILE loops
- ❌ Dynamic SQL
- ❌ Complex temp table chains

Recommended approach

- Cursor → Set-based SQL / window functions
- Temp tables → CTEs or Delta tables
- UNPIVOT → UNION logic (your preferred style)

STEP 5 – Reconcile (Validate Data)

Purpose

Ensure **Databricks output matches source system output**.

☑ Command: Reconcile

databricks labs lakebridge reconcile

What Reconcile checks

- Row counts
- Aggregate values
- Data mismatches

📁 Output:

.lakebridge/reconcile/
└── reconciliation_report.json

STEP 6 – Aggregated Reconciliation (Optional)

For large datasets:

databricks labs lakebridge aggregates-reconcile

✓ Faster

✓ Used for production-scale validation

🔍 Helpful Supporting Commands

Check installed transpilers

databricks labs lakebridge describe-transpile

Run with debug logs

databricks labs lakebridge transpile --debug

Get JSON output (for reports)

databricks labs lakebridge analyze --output json

🧠 How This Fits Real Projects (Your Context)

For SQL Server → Databricks migrations:

1. **Analyze** → find unsupported banking logic
2. **Transpile** → get baseline Databricks SQL
3. **Manual rewrite** → optimize CTEs, unions, buckets
4. **Reconcile** → validate IRRBB / loan numbers

Lakebridge helps you **start faster**, not finish blindly.

☑ One-Page Summary

Step	Command
Analyze	lakebridge analyze
Convert	lakebridge transpile
Validate	lakebridge reconcile