GIT Repositories

Recommended end-to-end Spark ETL Scala repositories

1. **yennanliu / spark-etl-pipeline** — Demo **ETL/stream & batch examples** (multiple small projects and examples). Good for learning practical ETL flows and many small demos in one repo.
   Repository: GitHub

   Quick note: contains multiple subprojects (batch, streaming, EMR demos). To run on Spark 4: update `build.sbt` dependency versions to `4.0.0`, set `scalaVersion := "2.13.16"`, and build with sbt 1.10.x. Watch for plugin/dependency upgrades.

2. **skalskibukowa / Project-Spark-Scala-ETL** — A focused ETL project that extracts from CSV / Postgres, transforms and writes to sinks (**CSV/Parquet/Postgres**). Good for typical ingestion → transform → load pipelines.
   Repository: GitHub
   Quick note: likely uses older Spark/Scala versions — update `build.sbt/pom.xml` as above; confirm any JDBC driver compatibility with Java 17.
3. **guruprvn84 / ScalaSparkETLArchetype** — An ETL project template/archetype for Scala + Spark. Useful as a starting skeleton to create production ETL apps with structure & conventions.
   Repository: GitHub
   Quick note: project templates often only need dependency updates and minimal code changes to compile on Spark 4 + Scala 2.13.16.
4. **vbounyasit / MyDataFramework** — A reusable Scala ETL framework built on Spark (framework-style rather than a single app). Good if you want to adopt a framework pattern for multiple pipelines.

Repository: GitHub
Quick note: framework-level code may require more changes when upgrading Spark versions — run tests after bumping Spark artifacts to 4.0.0.

5. **qwshen / spark-etl-framework** — Pipeline-based Spark-SQL transformation framework. Defines readers/transformers/writers as pipeline actors — great for building structured ETL pipelines.
   Repository: GitHub
   Quick note: primarily Spark-SQL focused — easier migration to Spark 4 if you keep SQL/DataFrame APIs.
6. **tosun-si / teams-league-airflow-spark-scala-etl** — End-to-end example showing Airflow orchestration, Scala Spark ETL on Dataproc/Serverless, and BigQuery sink. Real-world orchestration + cloud sink example.
   Repository: GitHub
   Quick note: this repo is useful if you plan to run ETL in cloud (GCP Dataproc). For Spark 4 check Dataproc runtime compatibility and change job submission parameters accordingly.
7. **bernhard-42 / Spark-ETL-Atlas** — Example project to demonstrate adding lineage (Apache Atlas) to Spark ETL jobs. Good if you need governance/lineage in your ETL.
   Repository: GitHub
   Quick note: Atlas integration often uses Spark listeners/APIs; verify listener APIs still match Spark 4 listener hooks.

Spark 4.0.0 / Scala / Java compatibility references

- Spark 4.0.0 release page and notes (Spark 4 is pre-built with Scala 2.13). Apache Spark+1

Quick migration checklist (how to make a repo Spark-4/Scala-2.13.16/Java-17 ready)

1. **Update build file**

For sbt (`build.sbt`):

```
scalaVersion := "2.13.16"
libraryDependencies += "org.apache.spark" %% "spark-sql" %
"4.0.0" % "provided"
```

   -
   - For Maven, set Spark `4.0.0` artifacts and `maven-compiler-plugin/source/target` to Java 17.
2. **sbt & plugins**
   - Use sbt **1.10.x** (`project/build.properties`) and upgrade any sbt plugins (sbt-assembly, etc.) to versions compatible with sbt 1.10.
3. **Java 17**
   - Compile with `-source 17 -target 17` / set `javaHome` to JDK 17.
   - Ensure any third-party native libs or connectors are Java 17 compatible.
4. **API changes**
   - Run `sbt compile` / `mvn -DskipTests=false test` and fix deprecations or method signature changes. Spark 4 tightened and removed some older APIs — focus on listener, RDD internals and deprecated SQL functions.
5. **Third-party libs**
   - Update connectors (JDBC drivers, Hive client, Hadoop client) to versions that support Java 17 and Spark 4.
6. **Testing**
   - Add small local smoke tests (run in `local[*]`) and integration tests. Use `spark-submit --master local[*]` to simulate cluster execution.

c

Which repos to pick depending on your goal

- Want **many small examples** & quick demos → `yennanliu/spark-etl-pipeline`. GitHub
- Need a **project skeleton / best-practices template** → `guruprvn84/ScalaSparkETLArchetype`. GitHub

- Want **framework-style pipelines** (reader/transformer/writer) → `qwshen/spark-etl-framework`. GitHub
- Building **cloud-orchestrated ETL** (Airflow + Dataproc) → `tosun-si/teams-league-airflow-spark-scala-etl`. GitHub
- Need **lineage/governance** integration → `bernhard-42/Spark-ETL-Atlas`. GitHub

---

1. stonezhong/spark_etl
   A generic ETL pipeline framework for Spark applications built in Python. It supports deployment on various cloud Spark platforms like Databricks, AWS EMR, Google Cloud, Azure, and more.
   GitHub Link
2. jamesbyars/apache-spark-etl-pipeline-example
   An example demonstrating robust ETL pipelines with Apache Spark, using Python and PostgreSQL for data retrieval and processing.
   GitHub Link
3. temcavanagh/Spark_ETL_Pipeline
   A Spark ETL pipeline for predictive modeling feature engineering, implemented with Spark-SQL and PySpark.
   GitHub Link
4. qwshen/spark-etl-framework
   A Scala-based Spark ETL framework using Spark-SQL with pipeline architecture defined via YAML/JSON/XML, suitable for complex workflows.
   GitHub Link
5. aphp/spark-etl
   Contains modules focusing on scalable ETL processes combining Apache Spark, Hive, Solr, and PostgreSQL.
   GitHub Link
6. SETL-Framework/setl
   A simple Spark-powered ETL framework in Scala designed to modularize and structure Spark

    ETL projects.
    GitHub Link

7. aws-samples/amazon-eks-apache-spark-etl-sample
   Examples and best practices for running Apache Spark ETL jobs on Amazon EKS with Dockerized Spark applications.
   GitHub Link

These repositories cover a range of **Scala** and **Python** with **Spark ETL projects** that could match your requirements for Spark 4.0.0, Python 3, and Java 17 environments. Let me know if you want detailed exploration or sample code from any specific repo.