Assignment 4 — Question 1

Exploring and Integrating Database Migration Tools

Date: August 16, 2025

This document compares two popular database migration tools—**Flyway** and **Liquibase**— and outlines a simple, practical strategy to integrate them into a CI/CD pipeline. The goal is to keep the approach clean, repeatable, and easy to run both locally and in GitHub Actions.

1) Tool Overviews & Key Features

Flyway is a lightweight, SQL-first migration tool that discovers versioned scripts (e.g., *V1__init.sql*) in order and applies them idempotently. It favors convention over configuration and is very easy to set up, often via Docker. Key features: Simple versioned migration naming and ordering SQL-first, with Java-based callbacks if needed Great for small-to-medium teams and microservices Works well in containers and CI/CD

Liquibase is a powerful, flexible migration tool that supports *changelogs* in SQL, XML, YAML, or JSON. It provides advanced features like *diff/generate-changelog*, rollback tags, and rich metadata. Key features: Multiple changelog formats (SQL/XML/YAML/JSON) Rollback, tagging, and built-in diff tools Large ecosystem and enterprise features Good for complex databases and regulated environments

2) Comparison (Flyway vs Liquibase)

Criteria	Flyway	Liquibase
Ease of Use	Very simple; SQL-first, convention over configuration.	More flexible; extra concepts (changelogs, contexts) to lea
CI/CD Integration	Straightforward via Docker/CLI; great in GitHub Actions.	Also good; richer options (tags, contexts, rollbacks).
Supported Databases	Broad (MySQL, PostgreSQL, SQL Server, Oracle, etc.)	Very broad; strong enterprise DB support.
Rollback Support	Primarily forward-only; clean rollback requires extra scripts.	First-class rollback via changesets and tags.
Schema Diff/Generation	Not built-in.	Built-in ‘diff’ and ‘generate-changelog’ tools.
Best Fit	Small/medium teams; microservices; simple SQL migrations	Complex/regulated systems needing rollbacks & metadata

3) CI/CD Integration Strategy

Goal: Make migrations automatic, reliable, and easy to run locally and in CI.

Branching: Use feature branches; PRs must pass migrations + tests.
Environments: *Local* (Docker MySQL), *CI* (GitHub Actions service DB), *Prod* (guarded).

Pipeline Steps (push/PR): Start database (service container or Docker Compose) Run **Flyway** initial migrations (schema bootstrap) Run **Flyway** incremental migrations Execute tests (CRUD/unit) On main branch (protected), gate deployments; for **Liquibase**, use tags and generate an audit trail **Rollbacks:** Flyway: provide down scripts or compensating migrations. Liquibase: use *rollback* with tags.
Observability: Log migration outputs; store artifacts (reports) if needed.

Summary

Flyway excels at simplicity and speed; Liquibase shines for richer governance and rollbacks. Either tool works well in CI/CD with containers. Choose Flyway for fast, SQL-first workflows; choose Liquibase when

you require changelog metadata, rollbacks, and auditing.