

# Q1 – Tools Analysis for CI/CD Database Migrations (MySQL)

---

**Context:** Target DB is **MySQL 8**. Migrations will be executed from containers in CI (GitHub Actions). We compare **Flyway** and **Liquibase** and propose a CI/CD strategy that cleanly separates initial vs. incremental migrations and validates changes with automated CRUD tests.

---

## Flyway — SQL-first, lightweight versioning

**What it is:** A migration tool that applies versioned SQL files (e.g., `V1__init.sql`) and tracks state in a `flyway_schema_history` table.

### Key features (high-value):

- Versioned and repeatable migrations (`V*` / `R__*`) with checksums and validation.
- Simple CLI and official Docker image; easy to call from CI.
- Baseline, validate, repair commands for messy histories.
- Placeholders and callbacks (pre/post migration hooks).
- Idempotent behavior encouraged by convention (e.g., `IF NOT EXISTS`).
- Very quick onboarding for SQL-centric teams.

**When it shines:** Teams that prefer **plain SQL**, want minimal ceremony, and value **fast CI** and simple containerized execution.

---

## Liquibase — changelogs, rollbacks, and governance

**What it is:** A migration platform using **changelogs** (YAML/JSON/XML/SQL) with strong metadata and governance features.

### Key features (high-value):

- Rich changelog model with **contexts/tags** and **preconditions**.
- Built-in **rollback** (auto or custom) and **diff/generateChangeLog**.
- CLI and Docker support; enterprise reporting/governance available.
- Checksums, labels, and advanced ordering rules.
- Good for regulated environments and multi-DB support at scale.

**When it shines:** Teams that need **formal rollbacks, auditable change controls**, and **metadata-rich** change management beyond plain SQL.

---

## Comparison (requested columns)

Criterion	Flyway	Liquibase
-----------	--------	-----------

Criterion	Flyway	Liquibase
Ease of Use	Very simple: versioned <b>SQL</b> files + CLI/Docker	Steeper ramp (changelogs, contexts, preconditions)
CI/CD Integration	Excellent: tiny CLI/Docker, fast validation + migrate	Excellent: CLI/Docker; more knobs for gates & governance
Supported Databases	MySQL, MariaDB, PostgreSQL, SQL Server, Oracle, etc.	MySQL, MariaDB, PostgreSQL, SQL Server, Oracle, etc.

**Why I'd choose Flyway here:** The course workflow is SQL-first, runs entirely in containers, and only needs forward migrations in CI. Flyway's simplicity and speed are ideal. Liquibase remains a great option if you must **enforce rollbacks** or require **changelog metadata** for compliance.

## Proposed CI/CD Strategy (steps + diagram)

**Goal:** Deterministic, container-only pipeline that (1) starts MySQL, (2) applies **initial** and **incremental** migrations safely, (3) runs **CRUD tests**, and (4) emits a clear deployment summary.

### Steps (concrete):

1. **Checkout** repository.
2. **Start MySQL service** (container) with `MYSQL_ROOT_PASSWORD` and a **course DB** (e.g., `prog8850_db`).
3. **Create app user** with least privileges for migrations/tests (e.g., `app_user/app_password`).
4. **Apply initial migrations (V1)**
  - Flyway Docker/CLI with `-locations=filesystem:./flyway/migrations_initial`.
  - Use `?allowPublicKeyRetrieval=true&useSSL=false` on MySQL 8 to avoid RSA key issues in CI.
5. **Apply incremental migrations (V2+)**
  - **Important:** point Flyway to **both** locations
    - 
    - `locations=filesystem:./flyway/migrations_initial,filesystem:./flyway/migrations_incremental`
 so validation sees the previously applied V1 while applying V2.
6. **Run automated tests (pytest)** against the running DB (CRUD: create, read, update, delete).
7. **Publish test report** (JUnit XML) as a CI artifact for traceability.
8. **Echo deployment summary:** `Deployment done for commit $GITHUB_SHA`.
9. **(Optional) Teardown** in local automation (CI ephemerality already isolates state).

### Simple pipeline diagram (Mermaid):

```
flowchart LR
A[Push / PR] --> B[GitHub Actions Job]
B --> C[MySQL Service (Container)]
C --> D[Flyway: V1 (initial)]
D --> E[Flyway: V2+ (initial + incremental locations)]
E --> F[Pytest CRUD]
```

```
F --> G[Upload JUnit XML Artifact]
G --> H[Echo "Deployment done for commit ..."]
```