Flyway Database migrations

CONCEPTS

> Never edit a migration that has been applied: its checksum is stored and validate will fail. Add a new one instead.

V1__init.sql files in locations
↓ `flyway migrate`
Target DB your schema
↓ `records`
Schema history flyway_schema_history
↓ `checked by`
validate checksums + order

> Versioned migrations run once in version order; repeatable ones (R__) rerun whenever their checksum changes, after all versioned ones.

MIGRATION FILES

`V<n>__<desc>.sql`
Versioned migration (runs once)

`R__<desc>.sql`
Repeatable migration

`U<n>__<desc>.sql`
Undo migration for a version

`flyway` **`add -description=<text>`**
Create next versioned script

`${<name>}`
Placeholder inside SQL

`-locations=filesystem:<dir>`
Where to scan for scripts

PROJECT AND SETUP

`flyway` **`init -projectName=<name> -databaseType=<type>`**
Generate project and flyway.toml

`flyway` **`-environment=<env>`**
Pick the target environment

`flyway` **`-url="<jdbc-url>" -user=<u> -password=<p> migrate`**
Connect without a config file

MIGRATE AND INSPECT

`flyway` **`migrate`**
Apply all pending migrations

`flyway` **`info`**
Show applied and pending

`flyway` **`validate`**
Compare files with history

`flyway` **`migrate -target=<version>`**
Stop at a version

`flyway` **`migrate -outOfOrder=true`**
Allow lower versions after higher

`flyway` **`migrate -baselineOnMigrate=true`**
Baseline a non-empty DB on first run

REPAIR AND RECOVER

`flyway` **`repair`**
Remove failed rows, realign checksums

`flyway` **`baseline`**
Mark existing DB as migrated

`flyway` **`baseline -baselineVersion=<v>`**
Baseline at a version

`flyway` **`clean -cleanDisabled=false`**
Drop all objects in schemas

> After a failed migration run repair, fix the script, then migrate again.

CONFIGURATION

`flyway.toml`
Project config file

`[environments.<name>] url, user, password`
Define an environment

`[flyway] locations = ["filesystem:sql"]`
Flyway settings in TOML

`FLYWAY_URL / FLYWAY_USER / FLYWAY_PASSWORD`
Env var overrides