Liquibase Database change tracking

START HERE

`liquibase` **`init project`**
Scaffold a project

`liquibase` **`tag --tag=`***`<name>`*
Mark a rollback point

`liquibase` **`validate`**
Check the changelog

`liquibase` **`rollback --tag=`***`<name>`*
Undo back to a tag

`liquibase` **`status`**
What is pending?

`liquibase` **`update-sql`**
Preview the SQL

`liquibase` **`update`**
Deploy changesets

`liquibase` **`history`**
What was deployed?

Changelog changesets in a file → `update` → **Liquibase** compares ids + checksums → `applies new` → **Database** your schema → `records in` → **DATABASECHANGELOG** what already ran

CONCEPTS

Liquibase runs changesets (id + author + file) from a changelog once, and records them in the DATABASECHANGELOG table with a checksum.

Global flags (--log-level, --defaults-file) go before the command; command flags (--url, --changelog-file) go after it.

Never edit a changeset that has already been deployed: its checksum changes and the next run fails. Add a new changeset instead.

PROJECT AND CONNECTION

`liquibase` **`--version`**
Show the installed version

`liquibase` *`<command>`* **`--help`**
Show every flag of a command

`--url=`*`<jdbc-url>`*
JDBC connection URL

`--username=`*`<user>`* **`--password=`***`<pw>`*
Database credentials

`--changelog-file=`*`<path>`*
Root changelog (xml, yaml, json or sql)

PROJECT AND CONNECTION (cont.)

`--defaults-file=`*`<file>`*
Read settings from another properties file

`--log-level=`*`<level>`*
Set verbosity (global flag, goes first)

CHANGELOG FORMATS

`-- liquibase formatted sql`
First line of a SQL changelog

`-- changeset` *`<author>:<id>`*
Start a changeset in a SQL changelog

`-- rollback` *`<sql>`*
Rollback statement for the changeset above

`<include file="<path>"/>`
Pull another changelog into the root file

`-- changeset` *`<a>:<id>`* **`runOnChange:true`**
Re-run when its SQL changes

`<tagDatabase tag="<name>"/>`
Tag the database from inside a changelog

A changeset is identified by id + author + file path. Moving or renaming the file makes Liquibase treat it as new.

UPDATE

`liquibase` **`update`**
Deploy all pending changesets

`liquibase` **`update-sql`**
Print the SQL update would run

`liquibase` **`update-count --count=`***`<n>`*
Deploy only the next n changesets

`liquibase` **`update-testing-rollback`**
Deploy, roll back, redeploy to test rollbacks

`--context-filter=`*`<ctx>`*
Run only changesets in these contexts

Called --contexts in 4.23.0 and older.

ROLLBACK

`liquibase` **`rollback --tag=`***`<name>`*
Roll back to a tag

`liquibase` **`rollback-count --count=`***`<n>`*
Undo the last n changesets

`liquibase` **`rollback-to-date --date=`***`<date>`*
Undo everything after a date

`liquibase` **`rollback-sql --tag=`***`<name>`*
Preview the SQL of a tag rollback

ROLLBACK (cont.)

`liquibase` **`rollback-count-sql --count=`***`<n>`*
Preview the SQL of a count rollback

Rollback does not bring back data lost to DROP, DELETE or TRUNCATE. Preview with rollback-sql first.

Roll back before fixing a broken changeset. Editing it first causes a checksum mismatch that blocks the rollback.

INSPECT AND TRACK

`liquibase` **`status`**
List changesets not yet deployed

`liquibase` **`history`**
List changesets already deployed

`liquibase` **`tag --tag=`***`<name>`*
Mark the current database state

`liquibase` **`diff --reference-url=`***`<url>`*
Compare two databases

`liquibase` **`diff-changelog --reference-url=`***`<url>`*
Write a changelog that closes the diff

GENERATE AND SYNC

`liquibase` **`generate-changelog --changelog-file=`***`<file>`*
Reverse-engineer an existing database

`--schemas=`*`<a,b>`*
Limit generate or diff to some schemas

`liquibase` **`changelog-sync`**
Mark all pending changesets as run, without running

Baseline an existing database: generate-changelog, then changelog-sync, so the first update does nothing.

MAINTENANCE

`liquibase` **`validate`**
Find changelog errors before deploying

`liquibase` **`release-locks`**
Clear a lock left by a crashed run

`liquibase` **`clear-checksums`**
Reset stored checksums after an intended edit

A checksum error means a deployed changeset was edited. Revert the edit, or run clear-checksums if the change was deliberate.

CONFIGURATION

`url:` *`<jdbc-url>`*
Connection URL in liquibase.properties

`changelogFile:` *`<path>`*
Root changelog in liquibase.properties

Precedence: CLI flags beat system properties, then environment variables (LIQUIBASE_COMMAND_URL and so on), then liquibase.properties.

RECIPES

Safe deploy

```
liquibase validate
liquibase status --verbose
liquibase update-sql
liquibase tag --tag=before-release
liquibase update
```

Undo a release

```
liquibase rollback-sql --tag=before-release
liquibase rollback --tag=before-release
```

Adopt an existing database

```
liquibase generate-changelog --changelog-file=changelog.xml
liquibase changelog-sync
liquibase status
```