**General info**

* Config file for Luquibase **(CHANGELOG.YAML)** located in:   
  fax-service/src/main/resources/db/changelog.yaml
* DB version commits in liquibase called - **CHANGESETS**
* You can`t change existing changeset
* Rollback possible only if you manually remove **last** changeset from databasechangelog table

**Liquibase configuration (for Gradle)**

Add dependency to *build.gradle*

implementation 'org.liquibase:liquibase-core'

If you already connect Spring Boot application to db, Spring know how to configure liquibase under the hood, so required is only one property:

spring:

liquibase:

change-log: (path to changelog file)

*PS: You* ***CAN*** *connect sql, xml, yaml or json format type of changelog file.*

**Connecting more than one changelog file**

In case when you need two or more changelog files(for different envs, or tables/data), you can do it using yaml type of changelog file:

changelog.yaml

*Liquibase know how to handle it under the hood, so you don`t need any additional configurations*

**New db version commits**

DB version commits in liquibase called - **CHANGESETS**

***Important:***

* ***You can`t change existing changesets!***
* In our case, we have a separate file for each table, so if you want to create a new table, you need to create a new file with the following name: <previous file number+1>\_\_<table\_name>.sql ,   
  and start file with --liquibase formatted sql

If you want update db to new version, just follow next steps:

1. Navigate to the changelog file or the file with the required table (if you have more than one changelog file).   
   In our case, it is 1\_fax\_filter.sql
2. Create new changeset in next format:  
   *P.s: Don't forget increase version number*  
   --changeset <name>:<last db version +1> <any additional flags>
3. Provide sql ddl for your update  
   *P.s: you* ***can*** *add or update more than one table in single changeset*

(example of ddl)

*(example of change set, that is ready for commit)*

1. Restart the application and Spring will do all the work for you 🙂
2. You can also find the history of successful changesets in the databasechangelog table (will be generated automatically in schema you are connected to)

**Rollbacks**

***Rollback is not available for Gradle, or sql type of configuration***

(screenshot from documentation)

But we can do it manually if needed.  
If you have permission and access to db:

1. Navigate to the databasechangelog table
2. Find the **last** commit, you want to rollback
3. Remove it from table
4. You can now remove/update the sql code for the latest changeset
5. After you rerun your app, changeset will be rewritten