

# Quarkus - Using Liquibase

[Liquibase](#) is an open source tool for database schema change management.

Quarkus provides first class support for using Liquibase as will be explained in this guide.

## Setting up support for Liquibase

To start using Liquibase with your project, you just need to:

- add your changeLog to the `src/main/resources/db/changeLog.xml` file as you usually do with Liquibase
- activate the `migrate-at-start` option to migrate the schema automatically or inject the `Liquibase` object and run your migration as you normally do.

In your `pom.xml`, add the following dependencies:

- the Liquibase extension
- your JDBC driver extension (`quarkus-jdbc-postgresql`, `quarkus-jdbc-h2`, `quarkus-jdbc-mariadb`, ...)

```
<dependencies>
  <!-- Liquibase specific dependencies -->
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-liquibase</artifactId>
  </dependency>

  <!-- JDBC driver dependencies -->
  <dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-jdbc-postgresql</artifactId>
  </dependency>
</dependencies>
```

Liquibase support relies on the Quarkus datasource config. It can be customized for the default datasource as well as for every [named datasource](#). First, you need to add the datasource config to the `application.properties` file in order to allow Liquibase to manage the schema.

The following is an example for the `application.properties` file:

```
# configure your datasource
quarkus.datasource.db-kind=postgresql
quarkus.datasource.username=sarah
quarkus.datasource.password=connor
quarkus.datasource.jdbc.url=jdbc:postgresql://localhost:5432/mydata
base

# Liquibase minimal config properties
quarkus.liquibase.migrate-at-start=true

# Liquibase optional config properties
# quarkus.liquibase.change-log=db/changeLog.xml
# quarkus.liquibase.validate-on-migrate=true
# quarkus.liquibase.clean-at-start=false
# quarkus.liquibase.database-change-log-lock-table-
name=DATABASECHANGELOGLOCK
# quarkus.liquibase.database-change-log-table-
name=DATABASECHANGELOG
# quarkus.liquibase.contexts=Context1,Context2
# quarkus.liquibase.labels=Label1,Label2
# quarkus.liquibase.default-catalog-name=DefaultCatalog
# quarkus.liquibase.default-schema-name=DefaultSchema
# quarkus.liquibase.liquibase-catalog-name=liquibaseCatalog
# quarkus.liquibase.liquibase-schema-name=liquibaseSchema
# quarkus.liquibase.liquibase-tablespace-name=liquibaseSpace
```

Add a changeLog file to the default folder following the Liquibase naming conventions: `src/main/resources/db/changeLog.xml` The yaml, json, xml and sql changeLog file formats are also supported.

```

<?xml version="1.1" encoding="UTF-8" standalone="no"?>
<databaseChangeLog
  xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
  xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog-ext
    http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd
    http://www.liquibase.org/xml/ns/dbchangelog
    http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-
3.5.xsd">

  <changeSet author="quarkus" id="1">
    <createTable tableName="quarkus">
      <column name="ID" type="VARCHAR(255)">
        <constraints nullable="false"/>
      </column>
      <column name="NAME" type="VARCHAR(255)"/>
    </createTable>
  </changeSet>
</databaseChangeLog>

```

Now you can start your application and Quarkus will run the Liquibase's update method according to your config:

```

import org.quarkus.liquibase.LiquibaseFactory; ①

@ApplicationScoped
public class MigrationService {
    // You can Inject the object if you want to use it manually
    @Inject
    LiquibaseFactory liquibaseFactory; ②

    public void checkMigration() {
        // Get the list of liquibase change set statuses
        try (Liquibase liquibase =
liquibaseFactory.createLiquibase()) {
            List<ChangeSetStatus> status =
liquibase.getChangeSetStatuses(liquibaseFactory.createContexts(),
liquibaseFactory.createLabels());
        }
    }
}

```

① The Quarkus extension provides a factory to initialize a Liquibase instance

② Inject the Quarkus liquibase factory if you want to use the liquibase methods directly

# Multiple datasources

Liquibase can be configured for multiple datasources. The Liquibase properties are prefixed exactly the same way as the named datasources, for example:

```
quarkus.datasource.db-kind=h2
quarkus.datasource.username=username-default
quarkus.datasource.jdbc.url=jdbc:h2:tcp://localhost/mem:default
quarkus.datasource.jdbc.min-size=3
quarkus.datasource.jdbc.max-size=13

quarkus.datasource.users.db-kind=h2
quarkus.datasource.users.username=username1
quarkus.datasource.users.jdbc.url=jdbc:h2:tcp://localhost/mem:users
quarkus.datasource.users.jdbc.min-size=1
quarkus.datasource.users.jdbc.max-size=11

quarkus.datasource.inventory.db-kind=h2
quarkus.datasource.inventory.username=username2
quarkus.datasource.inventory.jdbc.url=jdbc:h2:tcp://localhost/mem:inventory
quarkus.datasource.inventory.jdbc.min-size=2
quarkus.datasource.inventory.jdbc.max-size=12

# Liquibase configuration for the default datasource
quarkus.liquibase.schemas=DEFAULT_TEST_SCHEMA
quarkus.liquibase.change-log=db/changeLog.xml
quarkus.liquibase.migrate-at-start=true

# Liquibase configuration for the "users" datasource
quarkus.liquibase.users.schemas=USERS_TEST_SCHEMA
quarkus.liquibase.users.change-log=db/users.xml
quarkus.liquibase.users.migrate-at-start=true

# Liquibase configuration for the "inventory" datasource
quarkus.liquibase.inventory.schemas=INVENTORY_TEST_SCHEMA
quarkus.liquibase.inventory.change-log=db/inventory.xml
quarkus.liquibase.inventory.migrate-at-start=true
```

Notice there's an extra bit in the key. The syntax is as follows: `quarkus.liquibase.[optional name.][datasource property]`.



Without configuration, Liquibase is set up for every datasource using the default settings.

# Using the Liquibase object

In case you are interested in using the `Liquibase` object directly, you can inject it as follows:



If you enabled the `quarkus.liquibase.migrate-at-start` property, by the time you use the Liquibase instance, Quarkus will already have run the migrate operation.

```
import org.quarkus.liquibase.LiquibaseFactory;

@ApplicationScoped
public class MigrationService {
    // You can Inject the object if you want to use it manually
    @Inject
    LiquibaseFactory liquibaseFactory; ①


    @Inject
    @LiquibaseDataSource("inventory") ②
    LiquibaseFactory liquibaseFactoryForInventory;


    @Inject
    @Named("liquibase_users") ③
    LiquibaseFactory liquibaseFactoryForUsers;


    public void checkMigration() {
        // Use the liquibase instance manually
        try (Liquibase liquibase =
liquibaseFactory.createLiquibase()) {
            liquibase.dropAll(); ④
            liquibase.validate();
            liquibase.update(liquibaseFactory.createContexts(),
liquibaseFactory.createLabels());
            // Get the list of liquibase change set statuses
            List<ChangeSetStatus> status =
liquibase.getChangeSetStatuses(liquibaseFactory.createContexts(),
liquibaseFactory.createLabels()); ⑤
        }
    }
}
```

- ① Inject the LiquibaseFactory object
- ② Inject Liquibase for named datasources using the Quarkus `LiquibaseDataSource` qualifier
- ③ Inject Liquibase for named datasources
- ④ Use the Liquibase instance directly
- ⑤ List of applied or not applied liquibase ChangeSets

# Configuration Reference

 Configuration property fixed at build time - All other configuration properties are overridable at runtime

Configuration property	Type	Default
 <code>quarkus.liquibase.change-log</code> The liquibase change log file. All included change log files in this file are scanned and add to the projects.	string	db/changeLog.xml
<code>quarkus.liquibase.migrate-at-start</code> <code>true</code> to execute Liquibase automatically when the application starts, <code>false</code> otherwise.	boolean	false
<code>quarkus.liquibase.validate-on-migrate</code> <code>true</code> to validate the applied changes against the available ones, <code>false</code> otherwise. It is only used if <code>migration-at-start</code> is <code>true</code>	boolean	true
<code>quarkus.liquibase.clean-at-start</code> <code>true</code> to execute Liquibase clean command automatically when the application starts, <code>false</code> otherwise.	boolean	false
<code>quarkus.liquibase.contexts</code> Comma-separated case-sensitive list of ChangeSet contexts to execute for liquibase.	list of string	
<code>quarkus.liquibase.labels</code> Comma-separated case-sensitive list of expressions defining labeled ChangeSet to execute for liquibase.	list of string	
<code>quarkus.liquibase.database-change-log-lock-table-name</code> The liquibase change log lock table name. Name of table to use for tracking concurrent Liquibase usage.	string	DATABASECHANGELOGLOCK
<code>quarkus.liquibase.database-change-log-table-name</code> The liquibase change log table name. Name of table to use for tracking change history.	string	DATABASECHANGELOG

<code>quarkus.liquibase.default-catalog-name</code>		
The name of Liquibase's default catalog.	string	
<code>quarkus.liquibase.default-schema-name</code>		
The name of Liquibase's default schema. Overwrites the default schema name (returned by the RDBMS) with a different database schema.	string	
<code>quarkus.liquibase.liquibase-catalog-name</code>		
The name of the catalog with the liquibase tables.	string	
<code>quarkus.liquibase.liquibase-schema-name</code>		
The name of the schema with the liquibase tables.	string	
<code>quarkus.liquibase.liquibase-tablespace-name</code>		
The name of the tablespace where the -LOG and -LOCK tables will be created (if they do not exist yet).	string	
 <code>quarkus.liquibase."named-data-sources".change-log</code>		
The liquibase change log file. All included change log files in this file are scanned and add to the projects.	string	db/changeLog.xml
<code>quarkus.liquibase."named-data-sources".migrate-at-start</code>		
<code>true</code> to execute Liquibase automatically when the application starts, <code>false</code> otherwise.	boolean	false
<code>quarkus.liquibase."named-data-sources".validate-on-migrate</code>		
<code>true</code> to validate the applied changes against the available ones, <code>false</code> otherwise. It is only used if <code>migration-at-start</code> is <code>true</code>	boolean	true
<code>quarkus.liquibase."named-data-sources".clean-at-start</code>		
<code>true</code> to execute Liquibase clean command automatically when the application starts, <code>false</code> otherwise.	boolean	false
<code>quarkus.liquibase."named-data-sources".contexts</code>		
Comma-separated case-sensitive list of ChangeSet contexts to execute for liquibase.	list of string	

<code>quarkus.liquibase."named-data-sources".labels</code>	list of string	
Comma-separated case-sensitive list of expressions defining labeled ChangeSet to execute for liquibase.		
<code>quarkus.liquibase."named-data-sources".database-change-log-lock-table-name</code>	string	DATABA SECHAN GELOGL OCK
The liquibase change log lock table name. Name of table to use for tracking concurrent Liquibase usage.		
<code>quarkus.liquibase."named-data-sources".database-change-log-table-name</code>	string	DATABA SECHAN GELOG
The liquibase change log table name. Name of table to use for tracking change history.		
<code>quarkus.liquibase."named-data-sources".default-catalog-name</code>	string	
The name of Liquibase's default catalog.		
<code>quarkus.liquibase."named-data-sources".default-schema-name</code>	string	
The name of Liquibase's default schema. Overwrites the default schema name (returned by the RDBMS) with a different database schema.		
<code>quarkus.liquibase."named-data-sources".liquibase-catalog-name</code>	string	
The name of the catalog with the liquibase tables.		
<code>quarkus.liquibase."named-data-sources".liquibase-schema-name</code>	string	
The name of the schema with the liquibase tables.		
<code>quarkus.liquibase."named-data-sources".liquibase-tablespace-name</code>	string	
The name of the tablespace where the -LOG and -LOCK tables will be created (if they do not exist yet).		