Baseline an Existing Database in Flyway

How to Baseline an Existing Database in Flyway (Step-by-Step)

Since your Application is already deployed and running with a PostgreSQL database, you need to "baseline" Flyway so it recognizes the existing schema without applying the initial scripts again.

Step 1: Add Flyway Dependency

Since you are using **Gradle**, add Flyway to `build.gradle`:

```
dependencies {
    implementation 'org.flywaydb:flyway-core:9.0.0'
    implementation 'org.postgresql:postgresql:42.5.0'
}
```

Run:

```
./gradlew build
```

Step 2: Enable Flyway in `application.properties`

Modify `src/main/resources/application.properties`:

```
# Database Configuration
spring.application.name=Credit_Card_Service
spring.datasource.url=jdbc:postgresql://localhost:5432/cred
it_card_db
spring.datasource.username=postgres
spring.datasource.password=******
spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.diale
ct.PostgreSQLDialect

# Server Configuration
server.port=8080

# location of the swagger json
#springfox.documentation.swagger.v2.path=/swagger.json

# Enable Flyway
spring.flyway.enabled=true

# Baseline the existing schema
spring.flyway.baseline-on-migrate=true

# Set initial baseline version
spring.flyway.baseline-version=1.0
```

This ensures that Flyway **does not try to apply scripts** on an already existing database and considers version `1.0` as the baseline.

Step 3: Manually Insert a Baseline Record in Flyway Table

Flyway keeps track of executed migrations in a table called `flyway_schema_history`. You need to create this table manually if it doesn't exist.

Run the following SQL query in **pgAdmin** or PostgreSQL CLI:

`Right click on your DB → query tool → paste the code → RUN`

```
CREATE TABLE IF NOT EXISTS flyway_schema_history (
    installed_rank INT PRIMARY KEY,
    version VARCHAR(50),
    description VARCHAR(200),
    type VARCHAR(20),
    script VARCHAR(1000),
    checksum INT,
    installed_by VARCHAR(100),
    installed_on TIMESTAMP DEFAULT now(),
    execution_time INT,
    success BOOLEAN
);

INSERT INTO flyway_schema_history (installed_rank, version,
description, type, script, checksum, installed_by, installe
d_on, execution_time, success)
VALUES (1, '1.0', 'Baseline', 'BASELINE', '', NULL, 'postgr
es', now(), 0, true);
```

This tells Flyway **not to reapply scripts** for version `1.0`.

Step 4: Move SQL Scripts to Flyway Directory

Flyway expects all SQL migrations to be inside:

```
src/main/resources/db/migration/
```

You can simply copy your existing sql script into your migration file..

For example you can follow this one -

Move your existing `schema.sql` and `data.sql` and rename them:

```
src/main/resources/db/migration/V1_0__Initial_Schema.sql
src/main/resources/db/migration/V1_1__Insert_Initial_Data.s
ql
```

Example of `V1_0__Initial_Schema.sql`:

```
CREATE TABLE IF NOT EXISTS agent (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    age INT NOT NULL,
    email VARCHAR(255) UNIQUE
);
```

Example of `V1_1__Insert_Initial_Data.sql`:

```
INSERT INTO agent (name, age, email) VALUES ('John Doe', 3
0, 'john@example.com');
```

Step 5: Run Flyway Migrations

Restart your application, and Flyway should now recognize that the database is at **version 1.0** and only apply new migrations.

Alternatively, run the Flyway command manually:

```
./gradlew flywayMigrate
```

Step 6: Verify in PostgreSQL

Check if Flyway has recorded the baseline:

```
SELECT * FROM flyway_schema_history;
```

Expected output:

```
+----------------+---------+------------+---------+--------
---------+----------+
| installed_rank | version | description | type   | script
| success  |
+----------------+---------+------------+---------+--------
---------+----------+
| 1              | 1.0     | Baseline    | BASELINE |
| true     |
+----------------+---------+------------+---------+--------
---------+----------+
```

Final Notes

- Flyway will now manage **all future schema changes**.
- **DO NOT** manually modify the database anymore—always create Flyway migration scripts.
- If a new column is required, create a `V1_2__Add_New_Column.sql` and let Flyway handle it.

NOTE : AFTER ADDING YOUR MIGRATION SCRIPT IF YOUR SPRING BOOT APPLICATION SHOULD NOT FAILED TO START THAT MEANS YOUR MIGRATION SCRIPTS ARE WORKING FINE. AND RUN SUCCESSFULLY...

Once your baseline migration will completed it can works normally, you can create more version of migration as you need for it.