PostgreSQL

PostgreSQL or **Postgres**, is a free and open-source relational database management system.

- □□ Upgrading to a Major Update
- □□♂ Renaming a Role (User)
- □□ Renaming a Database

? Upgrading to a Major Update

How to Upgrade a PostgreSQL database inside a Docker Container: A Step-by-Step Guide

Step 1: Backup the Database

Before upgrading, it's essential to back up your database to avoid data loss.

1. **Update `compose.yaml` to Add Volumes:**

In your `compose.yaml` file, add the necessary volumes for data and backups:

```
postgres:
    ...
    volumes:
        - data:/var/lib/postgresql/data
        - backup:/backup
    ...
```

2. **Shutdown All Containers Except the Database:**

Stop all your stack's containers except for the PostgreSQL container.

3. **Dump the Database into the Backup Directory:**

Run the following command to export all SQL tables into a dump file:

```
docker exec -it <DB_CONTAINER_NAME> pg_dumpall -U <POSTGRES_USER> > /backup/dump.sql
```

Step 2: Clean Up the Existing Stack

1. **Stop the Database Container:**

Bring down the PostgreSQL container:

```
docker compose down <DB_CONTAINER_NAME>
```

2. **Delete the Existing `data` Volume:**

To prepare for the upgrade, remove the current data volume associated with PostgreSQL.

Step 3: Upgrade PostgreSQL

1. **Update the PostgreSQL Version:**

In the `compose.yaml` file, change the PostgreSQL image to the new version.

2. **Bring Up the Updated PostgreSQL Container:**

Run the following command to start the new version:

```
docker compose up -d
```

Step 4: Clear the New Data Volume

1. **Stop All Stack Containers:**

Ensure all containers are stopped to avoid any conflicts.

2. **Delete Data Files in the Volume:**

Navigate to the PostgreSQL volume and remove the data files located in `/var/lib/docker/<DB_CONTAINER_NAME>/_data`.

Step 5: Restore the Database

1. **Start the Database Container:**

Power on the PostgreSQL container.

2. **Import the Data from Backup:**

Use the following command to restore the dumped SQL data into the new PostgreSQL container:

```
docker exec -it <DB_CONTAINER_NAME> psql -U <POSTGRES_USER> -d <POSTGRES_DATABASE_NAME> <
/backup/dump.sql
```

Final Step: Bring the Stack Back Online

Once the database has been restored, you can start all your other containers again.

And that's it! □□ You've successfully upgraded PostgreSQL in your Docker environment.

Happy me! □□

???? Renaming a Role (User)

Rename a PostgreSQL role in Docker by creating a temporary role, switching, renaming and deleting

---

> CHEATSHEET

1. `docker exec -it <POSTGRESQL_CONTAINER_NAME> psql -U <POSTGRESQL_USERNAME>` : *Connect to the PostgreSQL server as* `<POSTGRESQL_USERNAME>`
2. `\du` : *List all PostgreSQL roles*
3. `CREATE ROLE <NEW_USER> SUPERUSER LOGIN PASSWORD '<USER_PASSWORD>';` : *Create a new temporary role*
4. `\q` : *Exit the PostgreSQL prompt*
5. `ALTER USER <OLD_USER_NAME> RENAME TO <NEW_USER_NAME>;` : *Rename the desired role*
6. `DROP ROLE <TEMPORARY_ROLE>;` : *Delete the temporary role*

> DETAILS

Step 1: Connect to the PostgreSQL Server

To begin, connect to the PostgreSQL server within your Docker container using:

```
docker exec -it <POSTGRESQL_CONTAINER_NAME> psql -U <POSTGRESQL_USERNAME>
```

Step 2: Switch to a Different Role

 You’re likely connected with the role you intend to rename. PostgreSQL restricts renaming a role while connected to it.

 Solution: Connect as a Different Role

First, check existing roles:

```
\du
```

If there’s only one role (e.g., `postgres`), you’ll need to create a temporary role, `tmp`, switch to it, rename the target role, and then delete the temporary one.

Step 3: Create a Temporary Role

Create the new role by entering:

```
CREATE ROLE <NEW_USER> SUPERUSER LOGIN PASSWORD '<USER_PASSWORD>';
```

Exit the server using `\q` and reconnect with the new temporary role:

```
docker exec -it <POSTGRESQL_CONTAINER_NAME> psql -U <NEW_USER>
```

Step 4: Rename the Role

Now, rename the original role using:

```
ALTER USER <OLD_USER_NAME> RENAME TO <NEW_USER_NAME>;
```

Confirm the role changes by running `\du`. You should see both roles listed.

Step 5: Delete the Temporary Role

Finally, confirm the temporary role exists with `\du`:

Then, delete it with:

```
DROP ROLE <TEMPORARY_ROLE>;
```

Check your roles again with `\du` to verify the temporary role was removed.

Happy Me! □□

? Renaming a Database

Renaming a PostgreSQL Database Inside a Docker Container: Step-by-Step Guide

Step 1: Connect to the PostgreSQL Server

To start, connect to the PostgreSQL server within the Docker container by running the following command:

```
docker exec -it <POSTGRESQL_CONTAINER_NAME> psql -U <POSTGRESQL_USERNAME>
```

Step 2: Switch to a Different Database

 When you connect to the PostgreSQL server, you’re likely accessing the database you want to rename. However, PostgreSQL doesn’t allow renaming a database while you’re connected to it.

 The Fix: Connect to a Different Database

To proceed, list all available databases on the server, then connect to another one:

```
\l
\c <ANOTHER_DB_NAME>
```

Step 3: Rename the Database

Now you can rename your database using the `ALTER DATABASE...RENAME TO` command:

```
ALTER DATABASE <OLD_DB_NAME> RENAME TO <NEW_DB_NAME>;
```

Happy me!