PostgreSQL

Postgres is an open-source relational database management system.

- Major Upgrade
- Renaming a Database
- Renaming a Role (User)

Major Upgrade

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 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! □□

Renaming a Role (User)

Renaming a PostgreSQL Role 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 Role

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

The Fix: Create and Connect with a Different Role

To proceed, enter the following command:

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

Disconnect from the server with the `\q` command then reconnect with the **<NEW_USER>** created:

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

Step 3: Rename the Role

Now you can rename your role using the `ALTER USER...RENAME TO` command:

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

Happy me!