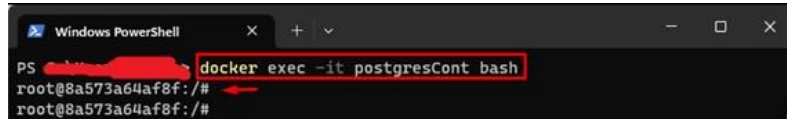


Step 4: Connect to the PostgreSQL Database

Now that our PostgreSQL container is up and running, let's connect to the database. We can do this by executing the psql command inside the container:

```
docker exec -it mypostgres psql -U postgres
```

This command opens an interactive terminal inside the container and connects to the PostgreSQL database using the postgres user.



Execute the "psql" command along with the hostname and user name to make a connection with the Postgres Database Server:

```
psql -h localhost -U postgres
```

```
root@8cda40e90b73:/# psql -h localhost -U postgres
psql (16.2 (Debian 16.2-1.pgdg120+2))
Type "help" for help.

postgres=#
```

Step 5: Create a New Database

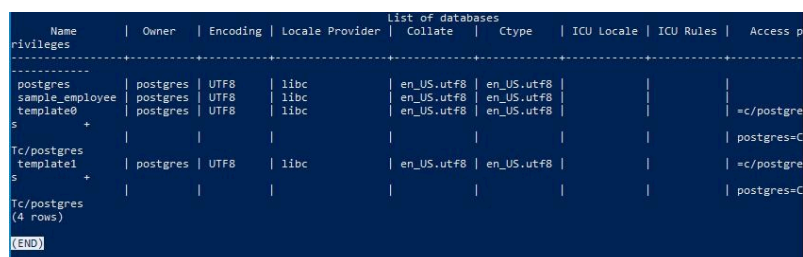
Once connected to the PostgreSQL database, we can create a new database using the following SQL command:

```
CREATE DATABASE mydatabase;
```

Replace mydatabase with your desired database name.

Before creating just to display list of all databases use command:

```
\1
```



Name	Owner	Encoding	Locale Provider	Collate	Ctype	ICU Locale	ICU Rules	Access privileges
postgres	postgres	UTF8	libc	en_US.utf8	en_US.utf8			
sample_employee	postgres	UTF8	libc	en_US.utf8	en_US.utf8			
template0	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres
template1	postgres	UTF8	libc	en_US.utf8	en_US.utf8			=c/postgres

As you can see I have already a sample database created "sample_employee"

Just to get out of (END) prompt use following command:

```
q
```

Step 6: Connect to the Newly Created Database

To connect to the newly created database, use the `\c` command followed by the database name:

```
\c mydatabase
```

```
postgres=# \l
postgres=# \c sample_employee
You are now connected to database "sample_employee" as user "postgres".
sample_employee=#
```

You are now connected to the mydatabase database and can start executing SQL queries and creating tables.

Step 7: Create a Table and Insert Data

Let's create a simple table called users and insert some sample data:

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100)
);

INSERT INTO users (name, email) VALUES ('John Doe',
'john@example.com');
INSERT INTO users (name, email) VALUES ('Jane
Smith', 'jane@example.com');
```

```
sample_employee=# CREATE TABLE users (
sample_employee(# id SERIAL PRIMARY KEY,
sample_employee(# name VARCHAR(50),
sample_employee(# email VARCHAR(100)
sample_employee(# );
CREATE TABLE
sample_employee=#
sample_employee=# INSERT INTO users (name, email) VALUES ('John Doe', 'john@example.com');
INSERT 0 1
sample_employee=# INSERT INTO users (name, email) VALUES ('Jane Smith', 'jane@example.com');
INSERT 0 1
sample_employee=#
```

Step 8: Query the Data

To verify that the data was inserted successfully, we can query the users table:

```
SELECT * FROM users;
```

You should see the inserted records displayed in the output.

```
sample_employee=# SELECT * FROM users;
 id |  name  | email
-----+-----+-----
  1 | John Doe | john@example.com
  2 | Jane Smith | jane@example.com
(2 rows)

sample_employee=#
```

Conclusion

In this basic article, we learned how to configure and connect to a PostgreSQL database using Docker. By leveraging Docker, we can easily set up a PostgreSQL environment without the need for local installation. We covered the steps to pull the PostgreSQL image, create and run a container, connect to the database, create a new database, and perform basic SQL operations. With this knowledge, you can start developing applications that utilize PostgreSQL as the database back end, all within the convenience of Docker containers.

[Report this article](#)

Enjoyed this article?

Follow to never miss an update.



Muhammad Asad Kamran PMP, ACP, Prince2, TOGAF

Enterprise Digital Transformation Architect, Data Scientist, Certified PMP, ACP, Prince2, TOGAF