



## How to Create a PostgreSQL Database in Docker

- ⌚ July 31, 2023, 7:51 p.m.
- 💬 [Talha Saif Malik](#)
- RSS [Follow](#)
- 🏷️ [OpenSource Postgres](#)

**PostgreSQL** is a well-known relational DBMS that provides a variety of features, such as built-in or user-defined functions, operators, data types, and many more. It is capable of running on numerous platforms, including Windows, Docker, and Linux. Moreover, users can also use PostgreSQL in Docker to easily create and manage the PostgreSQL database without installing it on the local host machine.

Try the new [PgManage \(Open Source\)](#) and get rid of PgAdmin!

This article will demonstrate the method of creating a PostgreSQL database in Docker.

## How to Create/Set up a Postgres Database With Docker?

To create a PostgreSQL database in Docker, check out the below-listed steps:

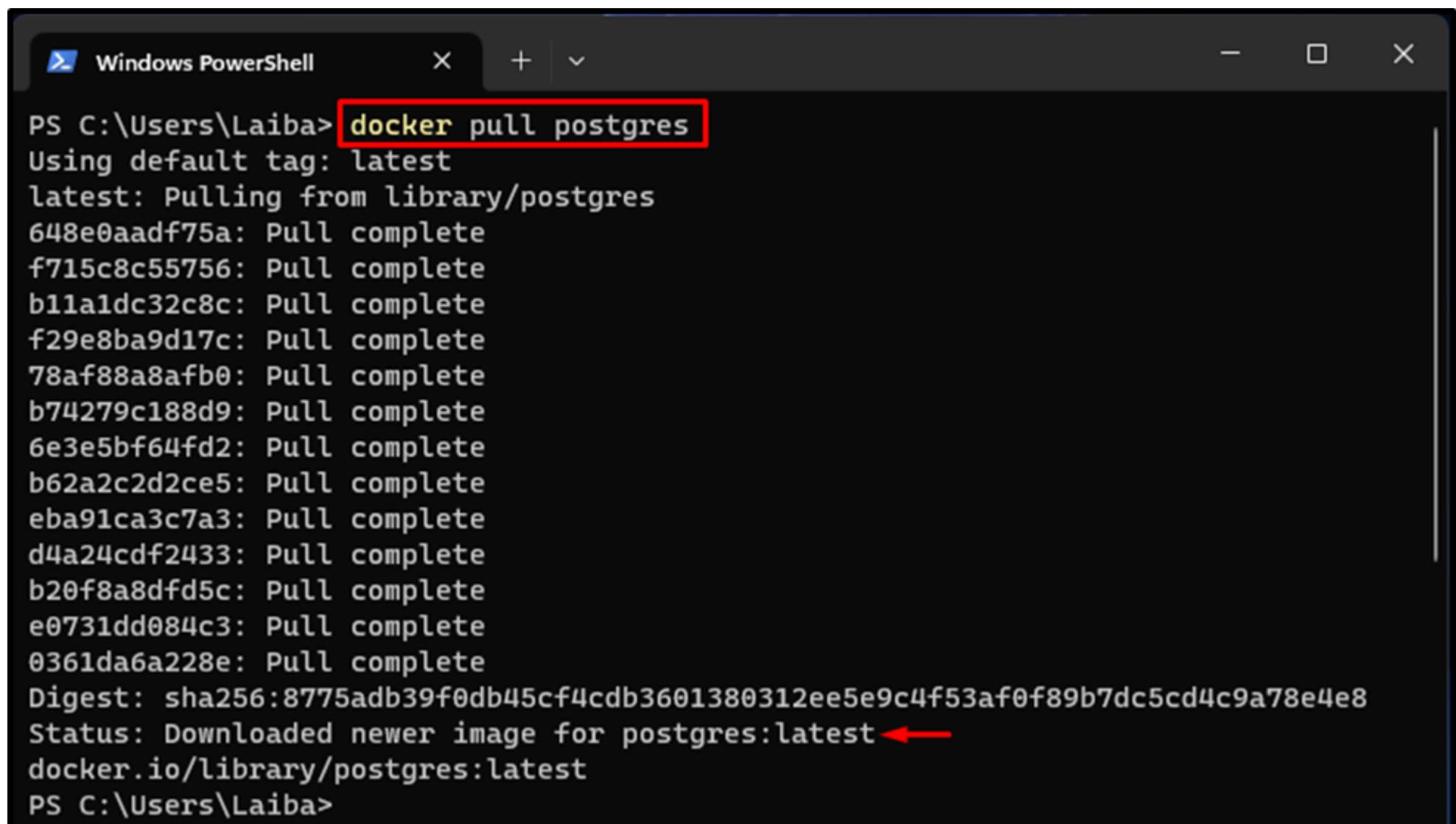
1. [Pull Official Postgres Image from Docker Hub](#)
2. [Create and Run Postgres Container](#)
3. [Verify Executing Container](#)
4. [Interact With Postgres Container](#)
5. [Connect to a PostgreSQL Database Server](#)
6. [Create a PostgreSQL Database](#)
7. [Confirm Database Creation](#)
8. [Establish a Connection With a Database](#)
9. [Create a Table in the Database](#)
10. [Insert Records into a Postgres Table](#)
11. [Fetch Table Data](#)

### [Pull/Download Official Postgres Image From Docker Hub](#)

Launch the Windows PowerShell from the start menu and execute the below-provided “**docker pull**” command to download the official Postgres image from Docker Hub to your local system:

```
docker pull postgres
```

Executing the above-stated command will download the latest version of the Postgres image, as shown in the following output snippet:



```
PS C:\Users\Laiba> docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
648e0aadf75a: Pull complete
f715c8c55756: Pull complete
b11a1dc32c8c: Pull complete
f29e8ba9d17c: Pull complete
78af88a8afb0: Pull complete
b74279c188d9: Pull complete
6e3e5bf64fd2: Pull complete
b62a2c2d2ce5: Pull complete
eba91ca3c7a3: Pull complete
d4a24cdf2433: Pull complete
b20f8a8df5c: Pull complete
e0731dd084c3: Pull complete
0361da6a228e: Pull complete
Digest: sha256:8775adb39f0db45cf4cdb3601380312ee5e9c4f53af0f89b7dc5cd4c9a78e4e8
Status: Downloaded newer image for postgres:latest
docker.io/library/postgres:latest
PS C:\Users\Laiba>
```

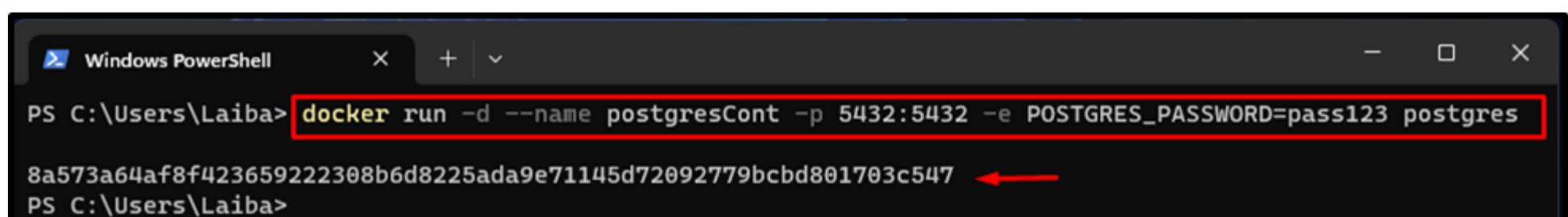
## Create and Run Postgres Container

Create and run the Postgres container using the Postgres image via the “**docker run --name -d <cont-name> -p 5432:5432 -e POSTGRES\_PASSWORD=<password> postgres**” command:

```
docker run -d --name postgresCont -p 5432:5432 -e POSTGRES_PASSWORD=pass123 postgres
```

Here:

- “**-d**” flag specifies that the container should execute in the background.
- “**--name**” option assigns the container’s name, i.e., “**postgresCont**”.
- “**-p**” assigns the port for the container i.e. “**5432:5432**”.
- “**-e POSTGRES\_PASSWORD**” configures the password to be “**pass123**”.
- “**postgres**” is the official Docker image:



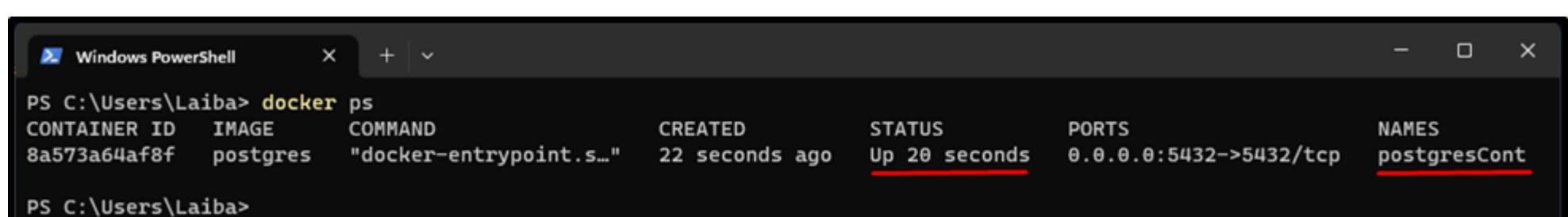
```
PS C:\Users\Laiba> docker run -d --name postgresCont -p 5432:5432 -e POSTGRES_PASSWORD=pass123 postgres
8a573a64af8f423659222308b6d8225ada9e71145d72092779bcd801703c547
PS C:\Users\Laiba>
```

Upon doing so, the Postgres container has been created and started.

## Verify Executing Container

Ensure that the Postgres container is built and currently executing via the given-provided command:

```
docker ps
```



COLUMN	CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
	8a573a64af8f	postgres	"docker-entrypoint.s..."	22 seconds ago	Up 20 seconds	0.0.0.0:5432->5432/tcp	postgresCont

The above output indicates that the “PostgresCont” container is running.

## Interact With Executing Container

Type out the “`docker exec -it <cont-name> bash`” command and specify the executing Postgres container name to open the shell within it:

```
docker exec -it postgresCont bash
```

```
PS C:\Users\Laiba> docker exec -it postgresCont bash
root@8a573a64af8f:/#
```

Subsequently, the “PostgresCont” container has been accessed and now we can run commands in it.

## Connect to a PostgreSQL Database Server

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
```

Executing the above-stated command will take us to the SQL Shell, where we can execute/run SQL queries and psql commands:

```
root@8a573a64af8f:/# psql -h localhost -U postgres
psql (15.3 (Debian 15.3-1.pgdg120+1))
Type "help" for help.

postgres=#
```

## Create a PostgreSQL Database

Now we are all set to create a new Postgres database. For this purpose, use/execute the “**CREATE DATABASE**” command along with the database name. For instance, we are creating a database named “`tsl_employee`”:

```
CREATE DATABASE tsl_employee;
```

```
postgres=# CREATE DATABASE tsl_employee;
CREATE DATABASE
postgres=#

```

## Confirm Database Creation

Display all the databases to view the newly created database:

```
\l
```

List of databases								
Name	Owner	Encoding	Collate	Ctype	ICU Locale	Locale Provider	Access privileges	Access privileges
postgres	postgres	UTF8	en_US.utf8	en_US.utf8		libc	=c/postgres	+
template0	postgres	UTF8	en_US.utf8	en_US.utf8		libc	=c/postgres	+
	postgres=CTc/postgres							
template1	postgres	UTF8	en_US.utf8	en_US.utf8		libc	=c/postgres	+
	postgres=CTc/postgres							
tsl_employee	postgres	UTF8	en_US.utf8	en_US.utf8		libc		
(4 rows)								

It can be seen that the “**tsl\_employee**” database has been created successfully.

## Establish a Connection With a Database

Type out the “**\c**” meta-command along with the database name of your choice to establish a connection with it. For instance, we want to connect to the “**tsl\_employee**” database:

```
\c tsl_employee;
```

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "\c tsl\_employee;". The response is "You are now connected to database \"tsl\_employee\" as user \"postgres\"." A red box highlights the command, and a red arrow points to the response message.

A connection has been successfully established with the specified database.

## Create a Table in the Database

To make a new table in the selected database, utilize the “**CREATE TABLE <table\_name>(col\_1 <data\_type>, col\_2 <data\_type>, col\_3 <data\_type>,..., col\_N <data\_type>);**” command. Where table\_name represents a table to be created, col\_1, col\_2, ..., col\_N are the column names, and data\_type represents any valid data type. Here, we are creating a table named “**tech\_authors**”:

```
CREATE TABLE tech_authors(ID INT PRIMARY KEY NOT NULL, NAME TEXT NOT NULL, TYPE TEXT NOT NULL, CATEGORY TEXT NOT NULL, ATICLES INT NOT NULL);
```

Executing the “**CREATE TABLE**” command will create a new table with the desired columns:

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "CREATE TABLE tech\_authors(ID INT PRIMARY KEY NOT NULL, NAME TEXT NOT NULL, TYPE TEXT NOT NULL, CATEGORY TEXT NOT NULL, ATICLES INT NOT NULL);". A red box highlights the entire command, and a red arrow points to the "CREATE TABLE" keyword.

## Insert Records into a Postgres Table

Use the “**INSERT INTO <table\_name> VALUES (value\_1, value\_2, value\_3, ...);**” command for inserting new records into the newly created table. For instance, we have inserted the following values:

```
INSERT INTO tech_authors VALUES (1, 'Laiba', 'Senior', 'Docker', 50);
```

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "INSERT INTO tech\_authors VALUES (1, 'Laiba', 'Senior', 'Docker', 50);". A red box highlights the command, and a red arrow points to the "INSERT" keyword.

## Fetch Table Data

Write out the provided command to view the specific table’s data:

```
SELECT * FROM tech_authors;
```

A screenshot of a Windows PowerShell window titled "Windows PowerShell". The command entered is "SELECT \* FROM tech\_authors;". The output shows a table with columns id, name, type, category, and aticles. One row is displayed: id=1, name='Laiba', type='Senior', category='Docker', aticles=50. A red box highlights the command, and a red arrow points to the value '50' in the output.

id	name	type	category	aticles
1	Laiba	Senior	Docker	50

In the above screenshot, the data of the “**tech\_authors**” table can be seen.

# Conclusion

To create a PostgreSQL database in Docker, first, pull/download the official Postgres image using the “`docker pull postgres`” command. Then, create and start the Postgres container via the “`docker run --name <cont-name> -p 5432:5432 -e POSTGRES_PASSWORD=<password> postgres`” command. After that, access the Postgres container and make a connection with the desired database. Next, create a Database utilizing the “`CREATE DATABASE <database-name>;`” command. Furthermore, users can create tables in the database, insert values, and select data from the database.

## Related Articles

[How to Create a Database in Postgres](#)

## Get Postgres Help!

FEATURES	CONSULTING	SLA	PROACTIVE
Enterprise Wide 24 x 7 x 365	✓	✓	✓
Service Level Agreement		✓	✓
Postgres and Full Stack Professional Services	✓	✓	✓
Monthly* billing cycle	✓	✓	✓
Fixed hourly rate	✓	✓	✓
Priority Response		✓	✓
Immunity from after hours and emergency rates		✓	✓
Advanced monitoring			✓
Four Hours of Professional Services or Support			✓

\*discounted quarterly and annual options are available

## Podcasts



# More than a refresh



# ON THE FLIP SIDE

A Podcast  
about  
funerals,  
books, and  
the essentials  
of brushing  
your teeth

By  
Lindsay Rae Hooper &  
Amanda Nystrom



## Recent blogs

[How to Use Insert Query in PostgreSQL](#)

April 22, 2024, 8 a.m.

[Talha Saif Malik](#)

[How to Rename Databases in PostgreSQL](#)

April 9, 2024, 8 a.m.

[Talha Saif Malik](#)

[A Comprehensive Guide on PostgreSQL SCHEMA](#)

April 2, 2024, 8 a.m.

[Talha Saif Malik](#)

[How to Use RENAME TABLE Statement in PostgreSQL](#)

March 1, 2024, 5:10 p.m.

[Talha Saif Malik](#)

[PostgreSQL INTEGER Data Type With Examples](#)

March 1, 2024, noon

[Talha Saif Malik](#)

[PostgreSQL TIME Data Type With Examples](#)

March 1, 2024, 5:07 a.m.

[Talha Saif Malik](#)

[PostgreSQL Aggregate Functions With Practical Examples](#)

Feb. 2, 2024, 1:10 a.m.

[Talha Saif Malik](#)

[PostgreSQL Data Types Explained](#)

Feb. 1, 2024, 12:52 a.m.

[Talha Saif Malik](#)

## Connect with us



Copyright © 2000-2024 Command Prompt, Inc. All Rights Reserved. All trademarks property of their respective owners.

[Terms of Use](#) [Privacy Policy](#)