# Reset Forgotten Password for postgres User

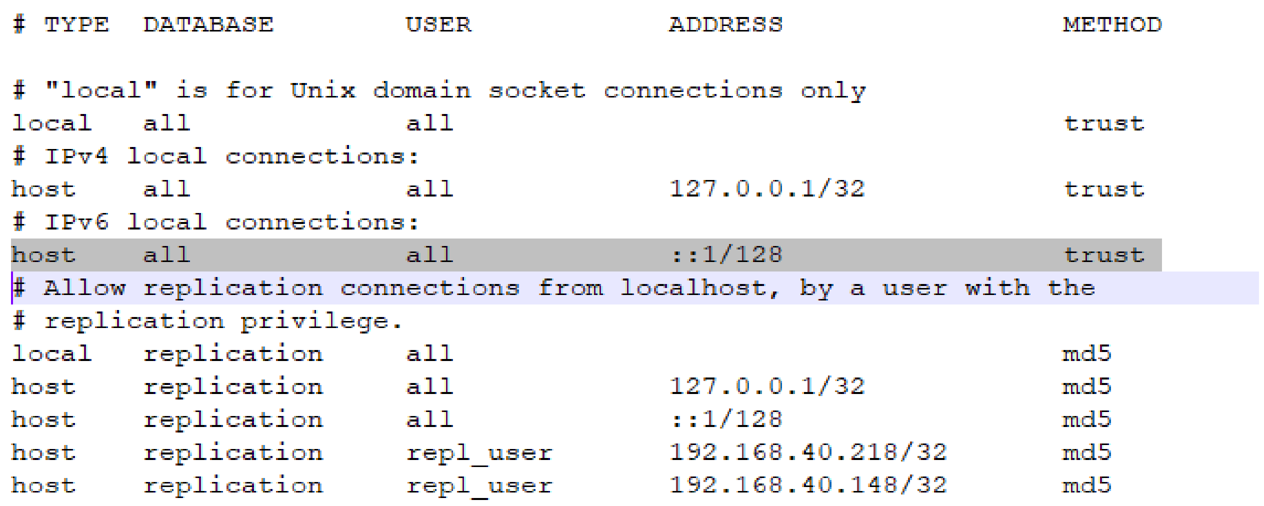
For some reason, after installing PostgreSQL, you may forget the password of the postgres user. In this case, you need to know how to reset the password**.**

**Step 1** PostgreSQL uses the pg_hba.conf configuration file stored in the database data directory (e.g., C:\Program Files\PostgreSQL\12\data on Windows) to control the client authentication. The hba in pg_hba.conf means host-based authentication.

To reset the password for the postgres user, you need to modify some parameters in this configuration file, login as postgres without a password, and reset the password.

The following steps show you how to reset a password for
the postgres user:

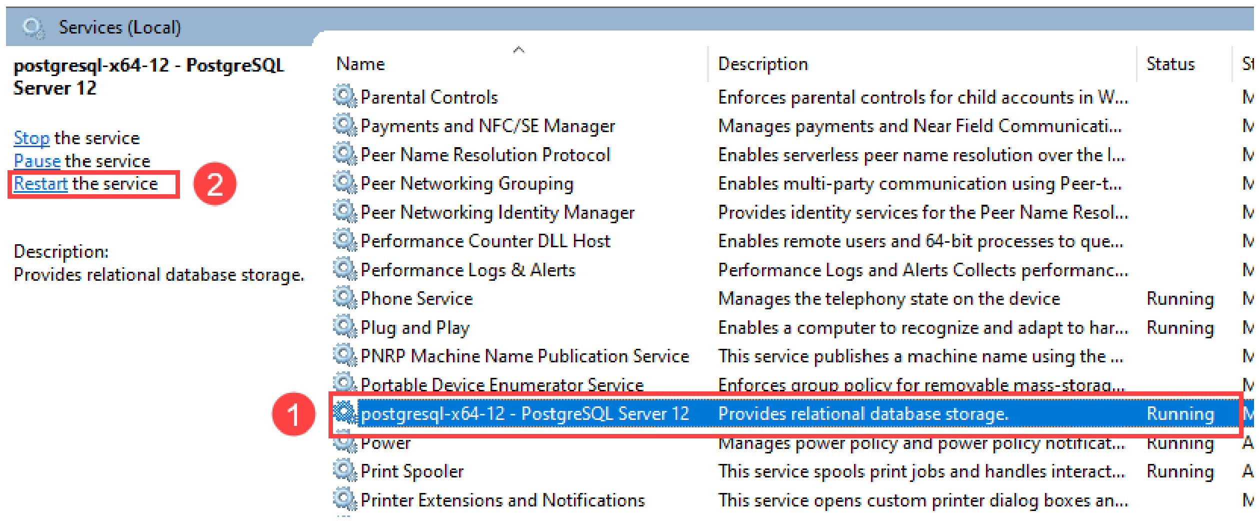
Edit the pg_dba.conf file and change all local connections from md5 to trust. By doing this, you can log in to the PostgreSQL database server without using a password.



```
# TYPE  DATABASE        USER            ADDRESS                 METHOD

# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    all             all             127.0.0.1/32            trust
# IPv6 local connections:
host    all             all             ::1/128                 trust
# Allow replication connections from localhost, by a user with the
# replication privilege.
local   replication     all                                     md5
host    replication     all             127.0.0.1/32            md5
host    replication     all             ::1/128                 md5
host    replication     repl_user       192.168.40.218/32       md5
host    replication     repl_user       192.168.40.148/32       md5
```

**Step 2.** Restart the PostgreSQL server. If you are on Windows, you can restart the PostgreSQL from **Services:**



**postgresql-x64-12 - PostgreSQL Server 12**

Stop the service
Pause the service
Restart the service

Description:
Provides relational database storage.

| Name | Description | Status | St |
|---|---|---|---|
| Parental Controls | Enforces parental controls for child accounts in W... | | M |
| Payments and NFC/SE Manager | Manages payments and Near Field Communicati... | | M |
| Peer Name Resolution Protocol | Enables serverless peer name resolution over the I... | | M |
| Peer Networking Grouping | Enables multi-party communication using Peer-t... | | M |
| Peer Networking Identity Manager | Provides identity services for the Peer Name Resol... | | M |
| Performance Counter DLL Host | Enables remote users and 64-bit processes to que... | | M |
| Performance Logs & Alerts | Performance Logs and Alerts Collects performanc... | | M |
| Phone Service | Manages the telephony state on the device | Running | M |
| Plug and Play | Enables a computer to recognize and adapt to har... | Running | M |
| PNRP Machine Name Publication Service | This service publishes a machine name using the ... | | M |
| Portable Device Enumerator Service | Enforces group policy for removable mass-storag... | | M |
| postgresql-x64-12 - PostgreSQL Server 12 | Provides relational database storage. | Running | M |
| Power | Manages power policy and power policy notificat... | Running | A |
| Print Spooler | This service spools print jobs and handles interact... | Running | A |
| Printer Extensions and Notifications | This service opens custom printer dialog boxes an... | | M |

**Step 3** connect to PostgreSQL database server using any tool such as psql or pgAdmin:

```
psql -U postgres
```

PostgreSQL will not require a password to login.

**Step 4**. Execute the following command to set a new password for the postgres user.

```
postgres=# ALTER USER postgres WITH PASSWORD 'new_password';
```
Code language: SQL (Structured Query Language) (sql)

**Step 5**. Restore the pg_dba.conf file, restart the PostgreSQL database server and connect to the PostgreSQL database server with the new password.

## TUNING MAX_CONNECTIONS IN POSTGRESQL

## What is max_connections?

max_connections determine the maximum number of concurrent connections to the database server.

## Why do people choose high values for max_connections?

There are several reasons:

1. Since you need a restart to increase max_connections, people want to be “on the safe side”.
2. The application developers convinced the DBA that they need many database connections for best performance.

## Risk of overloading the database

As long as all but three of your 500 database sessions are idle, not much harm is done. Perhaps taking the snapshot at the beginning of each query is a little slower, but you probably won’t notice that.

But there is nothing that can prevent 100 of the connections from becoming active at the same time. Perhaps a particular event occurred (everybody wants to buy a ticket the minute after sales started). Perhaps a rogue statement kept a lock too long and processes ready to run “piled up” behind the lock.

If that happens, your CPU and/or I/O subsystem will be overloaded. The CPU will be busy switching between the processes or waiting for I/O from the storage subsystem, and none of your database sessions will make much progress. The whole system can “melt down” and become unresponsive, so that all you can do is reboot.

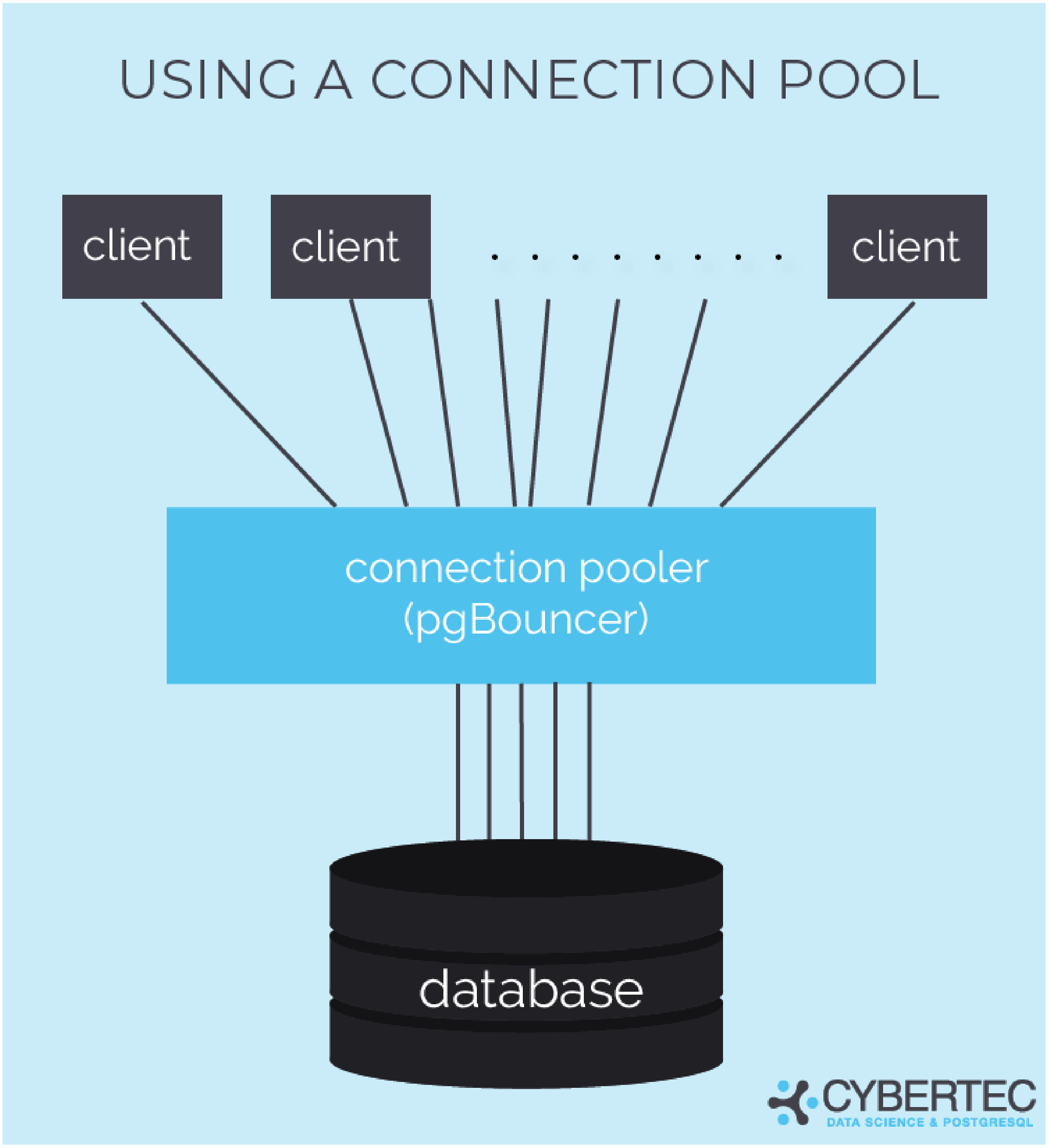
Obviously, you don’t want to get into that situation, and it is not going to improve application performance either.

## Insufficient resources for each database connection

There is a limited amount of RAM in your machine. If you allow more connections, you can allot less RAM to each connection. Otherwise, you run the danger of running out of memory. The private memory available for each operation during query execution is limited by work_mem. So you will have to set this parameter low if you use a high value for max_connections.

Now work_mem has a direct influence on query performance: sorting will be faster if it can use enough RAM, or PostgreSQL may prefer a faster hash join or hash aggregate and avoid a sort at all.

So setting max_connections high will make queries perform slower than they could, unless you want to risk running out of memory.



A connection pool is a piece of software that keeps a number of persistent database connections open. It uses these connections to handle database requests from the front-end. There are two kinds of connection pools:

- Connection pools built into the application or the application server. Rather than opening a database connection, application processes request a connection from the pool. When they are done, they return it to the pool rather than closing the connection.

- External connection pools like pgBouncer. Such a connection pool looks like a like a database server to the front end. SQL statements from the application are executed over a limited number of backend connections to the database.

Connection pools provide an artificial bottleneck by limiting the number of active database sessions. Using them increases the session_busy_ratio. This way, you can get by with a much lower setting
for max_connections without unduly limiting your application

F:\PostgreSQL\data\postgresql

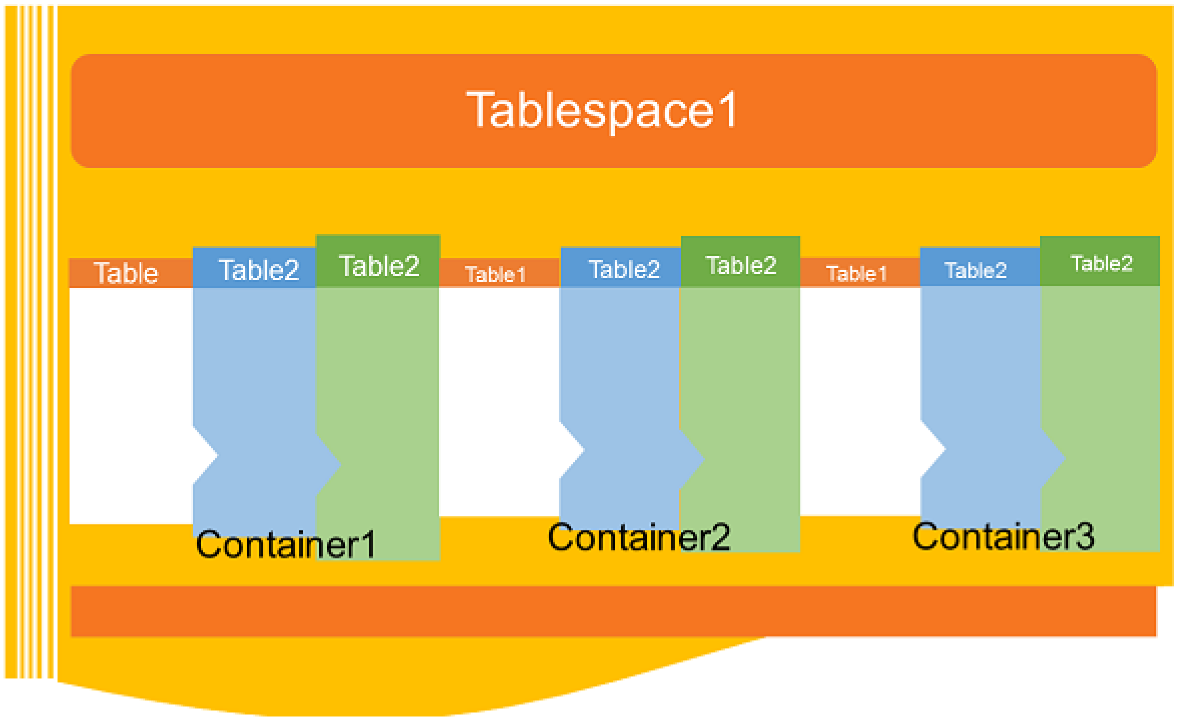
```
port = 5432                   # (change requires restart)

max_connections = 300         # (change requires restart)
```

then restart postgresql from services

## Introduction to PostgreSQL tablespace

A tablespace is a location on the disk where PostgreSQL stores data files containing database objects e.g., indexes, and tables.



PostgreSQL uses a tablespace to map a logical name to a physical location on disk.

PostgreSQL comes with two default tablespaces:

- pg_defaulttablespace stores user data.
- pg_globaltablespace stores global data.

Tablespaces allow you to control the disk layout of PostgreSQL. There are two main advantages of using tablespaces:

## Creating and using new tablespaces

To create a new tablespace, you first have to create a new directory. **Don’t create that directory in the PostgreSQL data directory!**

```
CREATE TABLESPACE mytbsp LOCATION 'D:\postgresql\data';


CREATE TABLE newtab (
   id  integer NOT NULL,
   val text    NOT NULL
) TABLESPACE mytbsp;

ALTER TABLE newtab
   ADD CONSTRAINT newtab_pkey PRIMARY KEY (id)
   USING INDEX TABLESPACE mytbsp;

CREATE INDEX newtab_val_idx ON newtab (val)
   TABLESPACE mytbsp;

  insert into newtab
  select x,'Bibek'
  from pg_catalog.generate_series(1, 1000000000) as x;
```

You can also create a database in a tablespace:

```
CREATE DATABASE newdb TABLESPACE mytbsp;
```

Then all objects you create in that database will automatically be placed in the database’s tablespace.