# PostgreSQL on Windows

**IMPORTANT**

> Disclaimer this document may break things on your Windows computer.
> Try at your own risk.
> It worked on my machine does NOT mean it will work on yours

That said:

This document explains where things are on a Windows 11 pc using PowerShell Admin.

The goal is to ensure that

- you start stop and monitor your local PostgreSQL server
- you can use command line tools such as pg_ctl,
- be able to start, stop, get the status of the PostgreSQL server with `pg_ctl`
- use `pg_restore` and `pg_dump` to respectively restore and dump a compressed sql backup file psql, pg*restore, pg*dump
- you know the location of PostgreSQL configuration files such as pg_hba.... `posgresql.conf` and .psqlrc
- you can add executables to the Machine PATH

We're exclusively working with **powershell Admin** terminal. No GUI.

The **powershell Admin** terminal is launched via: `Win + X` and selecting `Terminal Admin`.

# Install PostgreSQL on Windows 11

Downloading and installing PostgreSQL is straightforward. Go to PostgreSQL.org and find the download page or go directly to https://www.enterprisedb.com/downloads/PostgreSQL-PostgreSQL-downloads and click on the download arrow for Windows x86-64 and PostgreSQL 16

Then follow the instructions.

From now on we assume that PostgreSQL@16 has been installed in the folder:

```PowerShell
C:\Program Files\PostgreSQL\16\
```

You can check that the folder exists with the `Test-Path` utility

```PowerShell
Test-Path 'C:\Program Files\PostgreSQL\16\'
```

Note: the quotes around the path, which make up for the presence of a **space** in 'Program Files'.

> using space in filenames is a terrible practice, please don't do it

Anyway the above line should return a resounding **True**.

If not, that means PostgreSQL was installed elsewhere on your machine. Good luck.

# Manage PostgreSQL on the command line

On windows you start, stop, restart and monitor PostgreSQL with `pg_ctl`.

We need to make sure that program exists, is in the right place, does the job and finally we want to add it to the machine PATH.

`pg_ctl.exe` should be in the `'C:\Program Files\PostgreSQL\16\bin\'` folder. Which also contains things like `pg_restore`, `pg_dump`, `createdb`, `createuser` and `psql` and many other niceties.

To check if PostgreSQL is installed and if it can actually run, try:

```PowerShell
& "C:\Program Files\PostgreSQL\16\bin\pg_ctl" start -D "C:\Program Files
```

This should return:

```PowerShell
PS C:\Users\alexis> & "C:\Program Files\PostgreSQL\16\bin\pg_ctl" start
waiting for server to start....2024-09-15 14:36:07.370 GMT [10044] LOG:
2024-09-15 14:36:07.370 GMT [10044] HINT:  Future log output will appear
 done
server started
```

If that's the case, **congratulations** you have correctly installed PostgreSQL@16 on your windows box!

## Add pg_ctl to the PATH

Now we don't want to have to remember all these paths every time we want to start or stop the PostgreSQL server. So we need to first add the location of `pg_ctl` to the **PATH**.

On windows there are 2 different paths! The **machine** and the **user** path. Machine PATH makes thing available to the whole machine and user path only to the current user (hopefully, this would make sense).

The PATH is an environment variable that list all the directories where an executable could be found. This is super important because before windows (or any other OS for that matter) can execute / run a program it needs to know where to find it. Makes sense.

99% of *oops app not found errors* are due to a non complete PATH (trust me on that 99%).

So, if you want to start PostgreSQL server with a simple ...

```PowerShell
pg_ctl start
```

*(we'll deal with the rest of the command:*
*`-D "C:\Program Files\PostgreSQL\16\data"` in a bit )*

... you must first add *"C:\Program Files\PostgreSQL\16\bin"* to the PATH.

Let's see what's in the Machine PATH:

```PowerShell
$currentPath = [Environment]::GetEnvironmentVariable("Path", "Machine")
echo $currentPath
```

should return something like

```PowerShell
PS C:\Users\alexis> echo $currentPath
C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\Syste
```

By the way, notice how we've used a variable `$currentPath` to store the value of the PATH.

PowerShell variables are very nice and will facilitate our lives a lot.

To add the folder to the PATH, just concatenate the PATTH with the executable location

```PowerShell
$postgresPath = "C:\Program Files\PostgreSQL\16\bin"
$newPath = $currentPath + ";" + $postgresPath
```

and finally, store that `$newPath` value as the Machine PATH with:

```PowerShell
[Environment]::SetEnvironmentVariable("Path", $newPath, "Machine")
```

The PATH should now contain the right folder.

Before you shout victory, let's check if your change is resilient enough.

The PATH (and many other configuration files and variables) are loaded when you launch a terminal window. So when you make changes to the PATH you need to either reload it or open a new window.

To reload the PATH from scratch, open a new PowerShell Admin window and then

```PowerShell
echo [Environment]::GetEnvironmentVariable("Path", "Machine")
```

should return a string that includes the **PostgreSQL\16\bin** folder.

```PowerShell
C:\windows\system32;C:\windows;C:\windows\System32\Wbem;C:\windows\Syste
```

# PowerShell Profile

Now let's deal with the rest of the command:
`-D "C:\Program Files\PostgreSQL\16\data"`

The shortest simplest way to deal with this is simply to set the `PGDATA` environment variable with:

```
```

This sets a permanent environment variable

which you an check with

```PowerShell
echo $PGDATA
```

# the PROFILE

As I said when you launch a PowerShell session a lots of stuff is loaded

According to my dear friend Claude.ai:

PowerShell has configuration files similar to .zshrc in Unix-like systems. In PowerShell, these are called "profile" scripts. There are several profile scripts that can be loaded automatically when PowerShell starts, depending on the user and whether you're running PowerShell or PowerShell ISE.

Here are the main PowerShell profile scripts, in order of precedence:

- All Users, All Hosts $PSHOME\Profile.ps1
- All Users, Current Host $PSHOME\Microsoft.PowerShell_profile.ps1
- Current User, All Hosts $Home[My ]Documents\PowerShell\Profile.ps1
- Current User, Current Host $Home[My ]Documents\PowerShell\Microsoft.PowerShell_profile.ps1

The most commonly used profile for individual users is the "Current User, Current Host" profile. This is what we'll work with.

First, check if the profile exists:

```PowerShell
Test-Path $PROFILE
```

If it doesn't exist (returns False), create it:

```PowerShell
Copyif (!(Test-Path -Path $PROFILE)) {
    New-Item -ItemType File -Path $PROFILE -Force
}
```

Open the profile in a text editor:

```PowerShell
notepad $PROFILE
```

Add the following line to set the PGDATA environment variable:

```PowerShell
$env:PGDATA = "C:\Program Files\PostgreSQL\16\data"
```

Save the file and close the editor.

To apply the changes immediately, you can dot-source the profile:

```PowerShell
. $PROFILE
```

Or simply restart your PowerShell session.

Now, every time you start PowerShell, this profile will be loaded, and the `PGDATA` environment variable will be set automatically.

Remember that this method sets the environment variable for the duration of your PowerShell session. If you want to set it system-wide and persistently, you would still need to use the Environment::SetEnvironmentVariable method as mentioned in the previous response.

### Troubleshooting

> Is 'C:\Program Files\PostgreSQL\16\data'

> pg_ctl start -D "C:\Program Files\PostgreSQL\16\data"

> & "C:\Program Files\PostgreSQL\16\bin\pg_ctl" start -D "C:\Program Files\PostgreSQL\16\data"

## Start Stop and Monitor

You should now be able to run the following commands

```PowerShell

# start
pg_ctl start

# stop
pg_ctl stop

# restart
# sometimes restarting works better than start (go figure)
pg_ctl restart

# is the server running
pg_ctl status
```

## pg_restore

I was able to restore the sql.backup file with

```PowerShell
pg_restore -d "treesdb_v01" `
  -U PostgreSQL `
  --no-owner `
  --no-data-for-failed-tables `
  --no-privileges `
  --section=pre-data `
  --section=data `
  --section=post-data `
  --verbose `
  --exit-on-error `
  --single-transaction `
  "C:\Users\alexis\work\epitadb\data\treesdb_v01.08.sql.backup"
```