POSTGRES DATABASE ADMINISTRATOR COURSE

BY: Mukesh Chaurasia

DAY – 10

TOPIC: POSTGRES INSTALLATION

MIND MAP

Postgres SQL Installation & Usage on Windows

- **Introduction**
  - Install & Use Postgres SQL
  - Comprehensive Guide
  - Support: Comment Section for Issues
- **Installation Steps**
  - Download PostgreSQL
    - Go to browser
    - Search 'Postgre SQL download'
    - Open official download link
    - Choose OS (windows)
    - Select latest version
    - Click 'Download the installer'
    - Enterprise DB site
    - Download starts (e.g 361MB)
  - Run Installer
    - Double click installer in download directory
    - Trust Microsoft unverified app prompt
    - Grant administrative privilege
    - Welcome Screen: Click Next
  - Configuration
    - Installation Directory (default: fine or change)
    - Component to install (keep default)
      - PostgreSQL Server (most important)
      - PGAdmin 4 (graphic user interface)
      - Stack Builder (overview provided)
      - Command Line Tools
    - Data directory folder (keep default)

        - Password setup
            - Remember Password for Login
            - Port Number: 5432 (important for front-end connection)
        - Advanced Options (Keep default)
    - Installation Process (Takes time)
    - Completion: Stack Builder Prompt
- **Stack Builder Overview**
    - Purpose: Install drivers and downgrade versions
    - Select PostgreSQL 16 on port 5432
    - Categories
        - Database drives (connect front-end / back-end)
        - Database Server (downgrade version e.g 13/14)
    - Cancel for now (not deep dive)
- **PGAdmin 4 Usage**
    - Open PGAdmin 4
    - Connect to Server
        - Expand Servers
        - Enter Password (set during installation)
        - Save Password (optional)
        - Conneccted: PostgreSQL server 16
    - Database Operations
        - Create Database
            - Right – click ‘Database’ -> Create -> Database
            - Enter Database Name (e.g testDB)
            - Optional: Encoding Style, Security
            - Script view (backend command)
            - Click save
        - Create Table
            - Expand ‘testDB’ -> Schemas -> public -> Tables
            - Right -click ‘Tables’ -> Create -> Table
            - Enter Table Name (e.g ‘employee’)
            - Add Columns
                - Click + icon
                - Column 1: Name ID, Type Integer
                - Column 2: Name full name,Type Character, Length 50
            - Click Save
            - Verify ‘employee’ table with ID and Full name columns
        - Query Tool (write script)

- Click on Database Name 9e.g testDB) -> Query Tool
- Write Custom Scripts

- Insert Data
  - Right -click 'employee' table -> Script -> Insert Script
  - Template provided
  - Insert values (e.g 1, Rahul Dravid)
  - Select query, Click run button
  - Insert more values (e.g 2,Rohit Sharma)
- Select Data
  - Right -click 'employee' table -> scripts -> select script
  - Template provided
  - Execute query to retrieve data
  - Verify inserted entries

- **Conclusion**
  - Successful PostgreSQL 16 Installation
  - PGAdmin 4 GUI usage: Create DB, Table, Insert, Select
  - More tutorials (SQL, PL/SQL, Trigger, Functions)

PostgreSQL Installation: A Comprehensive Guide

POSTGRES DATABASE ADMINISTRATOR COURSE

BY: Mukesh Chaurasia

PostgreSQL, often lauded as "the world's most advanced open-source relational database," offers unparalleled data integrity, robustness, and extensibility. This guide provides an in-depth, end-to-end walkthrough of its installation on two prevalent operating systems: Linux (specifically Ubuntu) and Windows, accompanied by a detailed conceptual flow diagram.

1. PostgreSQL Installation on Linux (Ubuntu Example)

Installing PostgreSQL on Ubuntu via the official PostgreSQL APT repository ensures you receive the latest stable versions and security updates efficiently.

Prerequisites: Before you begin, open your terminal. This is where all commands will be executed. Ensure you have sudo privileges.

Step 1: Install Prerequisite Packages These packages are fundamental for securely downloading and managing software repositories.

```
sudo apt install wget ca-certificates
```

- wget: A non-interactive network downloader, crucial for retrieving files from the web, including the GPG key.
- ca-certificates: Contains the list of trusted Certificate Authorities (CAs) which enable your system to verify the authenticity of SSL/TLS connections, essential for secure https communication with the PostgreSQL repository.

Step 2: Create the PostgreSQL Repository File This command adds the official PostgreSQL APT repository to your system's list of software sources.

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
```

- sudo sh -c '...': Executes the enclosed command with root privileges.
- echo "..." > /etc/apt/sources.list.d/pgdg.list: Writes the specified string into a new file named pgdg.list within the /etc/apt/sources.list.d/ directory. This directory is where APT looks for additional repositories.
- deb http://apt.postgresql.org/pub/repos/apt: Specifies the base URL for the PostgreSQL APT repository.
- $(lsb_release -cs): This command substitutes with your Ubuntu distribution's codename (e.g., jammy for Ubuntu 22.04 LTS). This ensures you're pulling packages specifically compiled for your version of Ubuntu.
- main: Refers to the main component of the repository, containing the primary packages.

Step 3: Add the PostgreSQL GPG Key This step imports the GPG (GNU Privacy Guard) key used to sign the PostgreSQL packages. This is a critical security measure that allows apt to verify the authenticity and integrity of the packages you download from the PostgreSQL repository, preventing tampering or malicious injections.

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

- wget --quiet -O - ...: Downloads the GPG key from the specified URL.
    - --quiet: Suppresses wget's output for a cleaner terminal.
    - -O -: Directs wget to output the downloaded content to standard output (stdout) instead of a file.
- | sudo apt-key add -: Pipes the output of wget (the GPG key) to sudo apt-key add - .
    - apt-key add -: Adds the GPG key read from standard input to your system's trusted keyrings.

Step 4: Update Package Lists After adding a new repository and its GPG key, you must update your local package index so apt is aware of the new packages available.

```
sudo apt update
```

- This command fetches the package information from all configured repositories, including the newly added PostgreSQL repository.

Step 5: Install PostgreSQL Now, you can install the PostgreSQL server and its essential extensions.

```
sudo apt install postgresql postgresql-contrib
```

- postgresql: Installs the core PostgreSQL server package, including the database server daemon, client utilities, and documentation.
- postgresql-contrib: Installs a collection of useful modules and extensions that are not part of the core server but provide additional functionality (e.g., uuid-ossp, pg_stat_statements).

Step 6: Verify Installation and Start Service Confirm that PostgreSQL is installed correctly and its service is running.

```
sudo systemctl status postgresql
```

```
sudo systemctl start postgresql
```

- sudo systemctl status postgresql: Displays the current status of the postgresql service. Look for Active: active (running) to confirm it's operational.
- sudo systemctl start postgresql: If the service is not running, this command will start it. PostgreSQL is typically configured to start automatically on boot after installation.

Step 7: Configure User and Password By default, PostgreSQL creates a superuser named postgres that corresponds to a system user of the same name. It's crucial to set a secure password for this database user.

```
sudo -i -u postgres

psql

ALTER USER postgres WITH PASSWORD 'your_new_password';

\q

exit
```

- sudo -i -u postgres: Switches to the postgres system user. The -i flag ensures an interactive login shell, sourcing the user's environment. This is necessary because only the postgres system user has trust authentication by default to access the psql command-line client for the postgres database user without a password.
- psql: Launches the PostgreSQL interactive terminal.
- ALTER USER postgres WITH PASSWORD 'your_new_password';: This SQL command changes the password for the postgres database superuser. **Replace 'your_new_password' with a strong, unique password.**
- \q: Quits the psql interactive terminal.
- exit: Exits the postgres system user session and returns to your original user.

PostgreSQL Installation on Windows

The EnterpriseDB (EDB) installer provides a straightforward, bundled installation experience for PostgreSQL on Windows, including essential tools like pgAdmin 4.

Prerequisites: Download the PostgreSQL installer from the official EnterpriseDB website. Always download from the *official* source to ensure security and integrity: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

Step 1: Run the Installer Locate the downloaded .exe file and double-click it to initiate the installation wizard.

Step 2: Follow the Installation Wizard The wizard guides you through various configuration options. Pay attention to the following:

- **Welcome Screen:** Click "Next" to proceed.
- **Installation Directory:** The default location (e.g., C:\Program Files\PostgreSQL\<version>) is generally recommended unless you have specific organizational requirements.
- **Select Components:**
    - **PostgreSQL Server:** The core database engine. **Must be selected.**
    - **pgAdmin 4:** A popular and powerful graphical administration and development tool for PostgreSQL. **Highly recommended.**
    - **Stack Builder:** A utility to download and install additional tools and drivers (e.g., PostGIS for spatial data, database drivers). **Optional but useful for future extensions.**
    - **Command Line Tools:** Includes psql, pg_dump, pg_restore, and other command-line utilities. **Essential for advanced administration and scripting.**
- **Data Directory:** This specifies where your actual database files will be stored (e.g., C:\Program Files\PostgreSQL\<version>\data). Choose a location with sufficient disk space and ideally on a drive separate from your OS drive for performance and recovery considerations.
- **Password for postgres superuser:** Set a strong password for the default postgres database superuser. **Remember this password; it's critical for accessing your databases.**
- **Port:** The default PostgreSQL port is 5432. It's generally advisable to keep this default unless there's a port conflict on your system.
- **Locale:** Defines the language, character set, and sorting rules for your database cluster. The default system locale is usually appropriate, but you can select a specific one if needed for internationalization.

Step 3: Complete the Installation After configuring the options, click "Next" to begin the installation process. The installer will extract files, create services, and set up the database cluster. Once complete, click "Finish."

- **Stack Builder Launch (Optional):** The installer might prompt you to launch Stack Builder. If you plan to install additional components immediately (like PostGIS), you can proceed; otherwise, you can uncheck this option and launch it later.

Step 4: Verify Installation and Connect (using pgAdmin 4) pgAdmin 4 provides a user-friendly interface to manage your PostgreSQL servers and databases.

1. **Launch pgAdmin 4:** You can typically find it in your Start Menu under "PostgreSQL
2. **Add a New Server Connection:**
    - In pgAdmin 4, under "Servers," right-click and select "Register" -> "Server...".
    - **General Tab:** Provide a meaningful name for your server connection (e.g., "My Local PostgreSQL").
    - **Connection Tab:**
        - **Host name/address:** localhost (if PostgreSQL is running on the same machine) or the IP address/hostname of the server.
        - **Port:** 5432 (or the port you configured).
        - **Maintenance database:** postgres (the default administrative database).
        - **Username:** postgres (the superuser you configured).
        - **Password:** Enter the password you set during the installation.
    - Click "Save."
3. **Connect to the Server:** pgAdmin 4 will attempt to connect to your PostgreSQL server. If successful, you'll see your server listed, and you can expand it to view databases, roles, tablespaces, and other database objects. This confirms a successful installation and connectivity.

Advanced Considerations and Best Practices:

- **Firewall Configuration:** Ensure your firewall (e.g., ufw on Linux, Windows Defender Firewall) is configured to allow incoming connections on port 5432 if you intend to access PostgreSQL from other machines.
- **pg_hba.conf:** For production environments, understanding and configuring pg_hba.conf (PostgreSQL Host-Based Authentication) is crucial for securing access to your database. This file controls which hosts are allowed to connect, which database users they can connect as, and what authentication methods are required.
- **postgresql.conf:** This is the main configuration file for the PostgreSQL server. Tuning parameters like shared_buffers, work_mem, maintenance_work_mem, and wal_level can significantly impact performance.
- **Backup Strategy:** Implement a robust backup strategy (e.g., pg_dump, pg_basebackup, continuous archiving with WALs) from the outset.
- **User Management:** Avoid using the postgres superuser for routine application connections. Create specific, least-privilege users and roles for your applications.
- **Monitoring:** Set up monitoring tools (e.g., pg_stat_activity, Prometheus/Grafana with node_exporter and postgres_exporter) to track database performance and health.
- **Disk I/O:** For optimal performance, ensure your data directory resides on fast storage (SSD preferred) with high I/O capabilities.

Conceptual Flow Diagram (Enhanced):

```
graph TD
    A[Start Installation Process] --> B{Choose Operating System};

    B -- Linux (Ubuntu) --> C[Linux Path: Pre-Installation Setup];
    C --> C1[Install 'wget' & 'ca-certificates'];
    C1 --> C2[Create PostgreSQL APT Repository File<br>/etc/apt/sources.list.d/pgdg.list];
    C2 --> C3[Add PostgreSQL GPG Key<br>(for package authenticity)];
    C3 --> C4[Update APT Package Lists<br>(to recognize new repo)];
    C4 --> C5[Install PostgreSQL Server & Contrib Packages];
```

C5 --> C6[Verify Service Status & Start if Needed
systemctl status postgresql];

C6 --> C7[Configure 'postgres' Superuser Password
(via psql interactive shell)];

C7 --> D[Installation & Initial Configuration Complete];

B -- Windows --> E[Windows Path: Installer Download & Execution];

E --> E1[Download EDB PostgreSQL Installer
from Official Site];

E1 --> E2[Run Installer Executable
(Double-click .exe)];

E2 --> E3[Installation Wizard Walkthrough];

E3 --> E3a[Select Components: Server, pgAdmin 4, CLI, Stack Builder];

E3a --> E3b[Specify Data Directory Location];

E3b --> E3c[Set 'postgres' Superuser Password];

E3c --> E3d[Define Port (Default 5432) & Locale];

E3d --> E4[Execute Installation Process
(File extraction, Service creation, Cluster init)];

E4 --> E5[Complete Installation & Finish Wizard];

E5 --> E6[Launch pgAdmin 4
(Graphical DB Management Tool)];

E6 --> D;

D --> F[Verify Database Connectivity];

F --> F1[Linux: Connect via psql
psql -U postgres -h localhost -d postgres];

F1 --> G[Begin Database Operations];

F --> F2[Windows: Register Server in pgAdmin 4];

F2 --> G;

G --> H[End of Initial Setup];

style A fill:#DDEBF7,stroke:#333,stroke-width:2px

style H fill:#DDEBF7,stroke:#333,stroke-width:2px

style B fill:#FFF2CC,stroke:#F8B500,stroke-width:2px

style D fill:#C6E0B4,stroke:#70AD47,stroke-width:2px

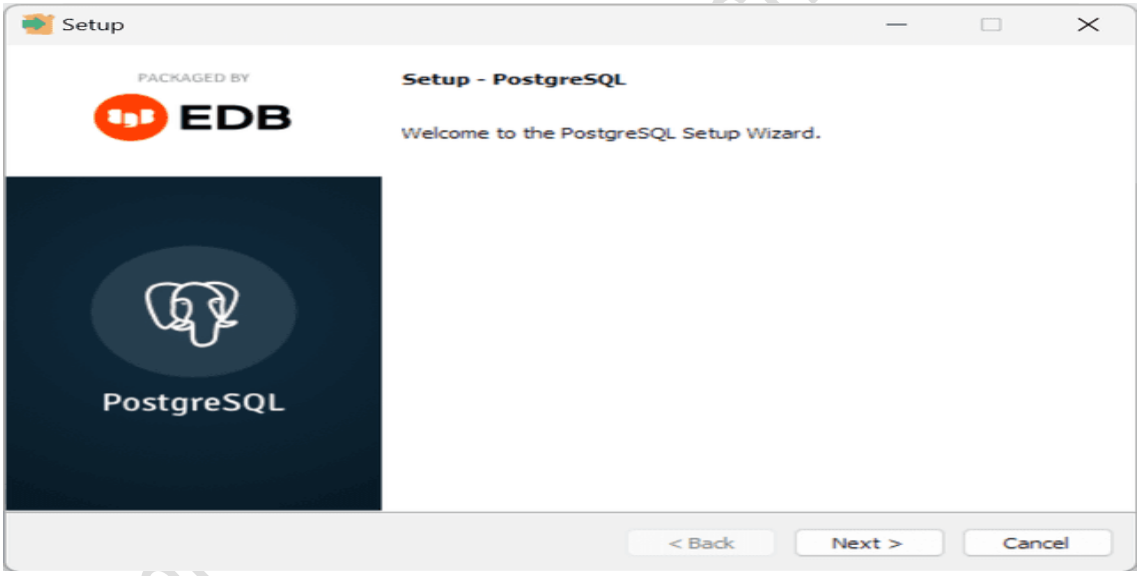
style F fill:#FFF2CC,stroke:#F8B500,stroke-width:2px

Install PostgreSQL on Windows step-by-step

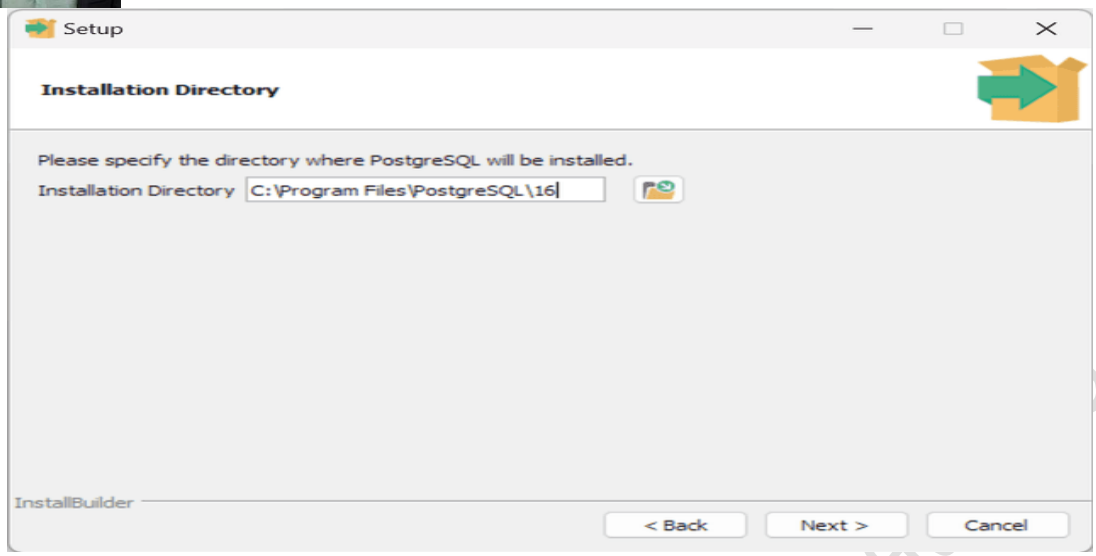
To install PostgreSQL on Windows, you need to have administrator privileges.

Step 1. Double-click on the installer file and an installation wizard will appear and guide you through multiple steps. Hence, you can choose various options based on your preferences for PostgreSQL.

Step 2. Click the Next button



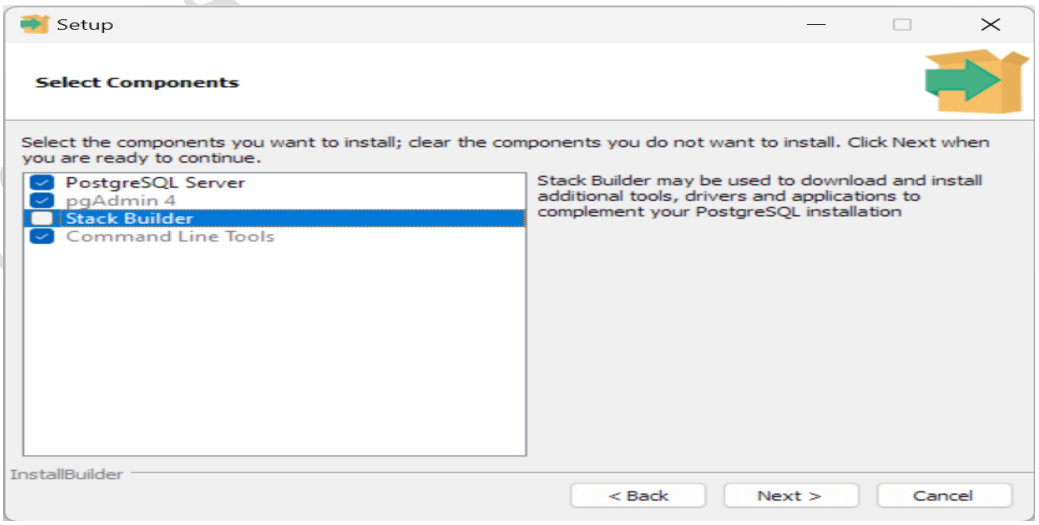
Step 3. Specify the installation directory by either choosing your preferred location or using the default folder suggested by the PostgreSQL installer and click the Next button



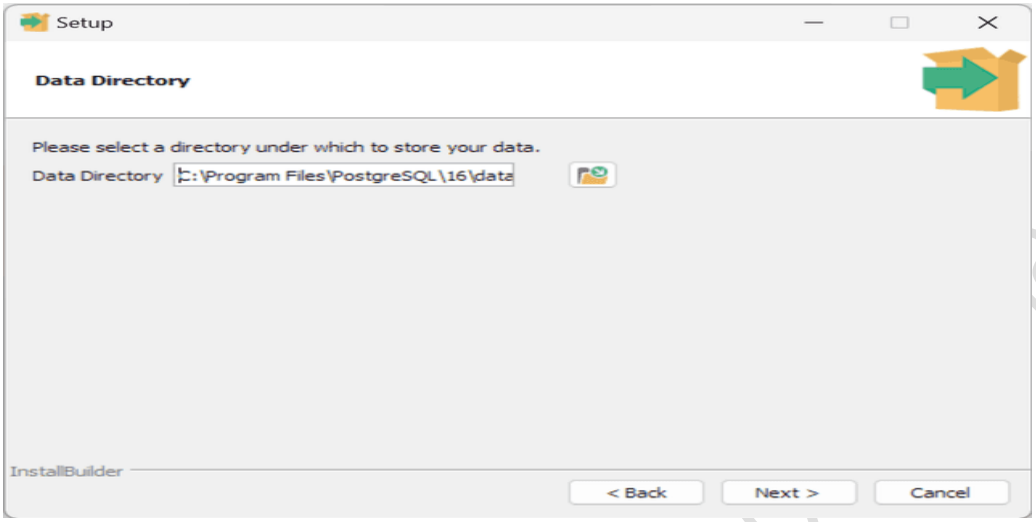
Step 4. Select software components to install:

- **PostgreSQL Server** option allows you to install the PostgreSQL database server
- **pgAdmin 4** option allows you to install the PostgreSQL database GUI management tool.
- **Stack Builder** provides a GUI that allows you to download and install drivers that work with PostgreSQL.
- **Command Line Tools** option allows you to install command-line tools such as psql, pg_restore, and so on. These tools allow you to interact with the PostgreSQL database server using the command-line interface.

For tutorials on this website, you can skip installing Stack Builder. Feel free to uncheck it and click the Next button to proceed to the data directory selection:



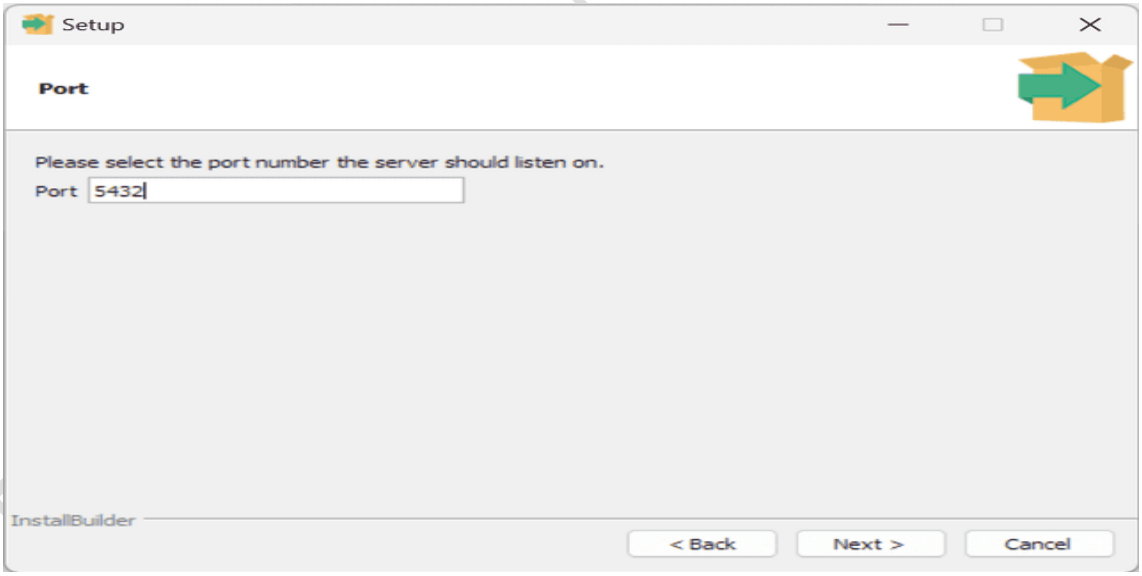
Step 5. Choose the database directory to store the data, or accept the default directory. Click the Next button to proceed to the next step:



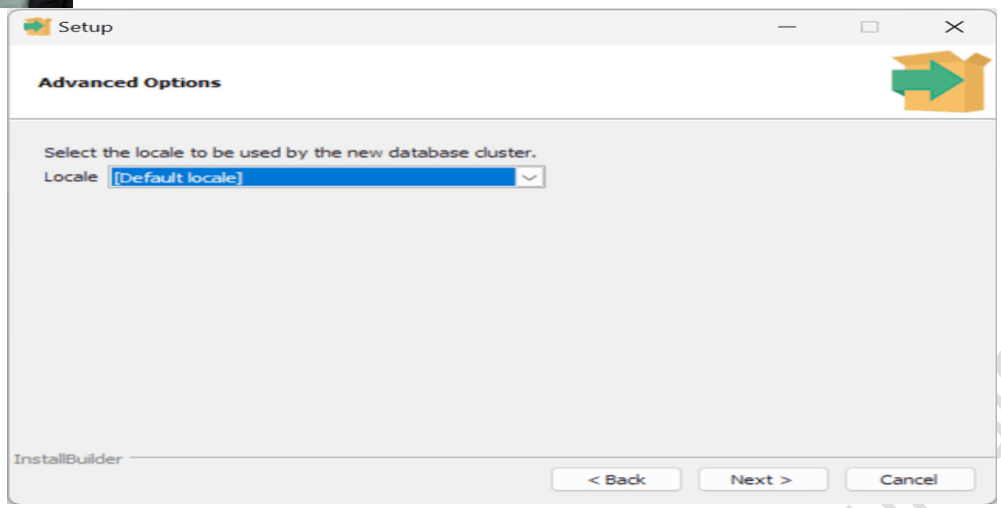
Step 6. Enter the password for the database superuser (postgres).

After entering the password, retype for confirmation, and then click the Next button:

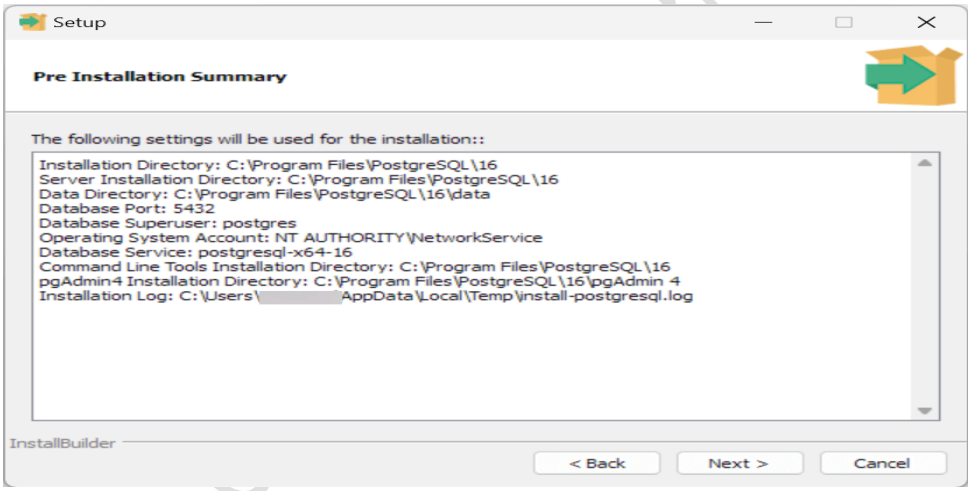
Step 7. Specify a port number on which the PostgreSQL database server will listen. The default port of PostgreSQL is 5432. Ensure that no other applications are using this port.



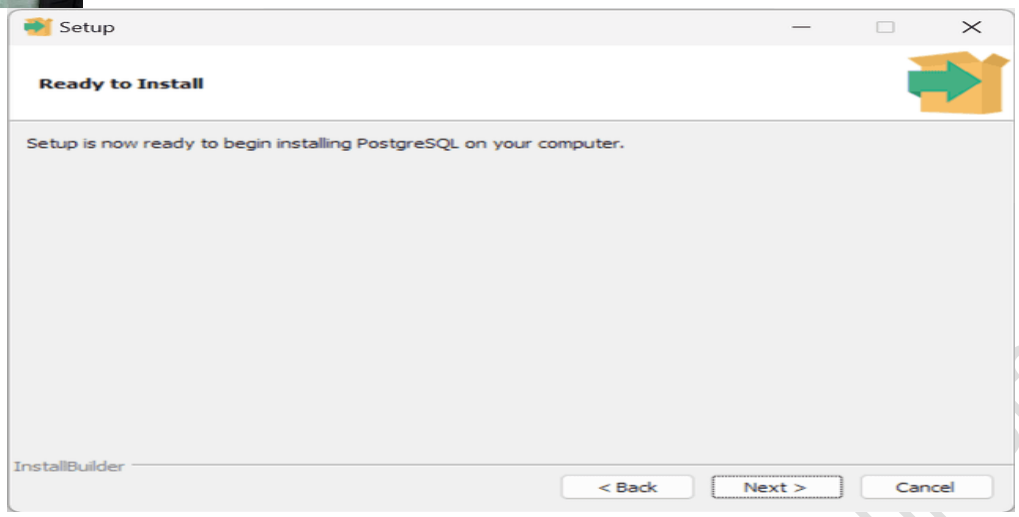
Step 8. Select the default locale for the PostgreSQL server. If you leave it as the default, PostgreSQL will use the operating system locale. Afterward, click the Next button.



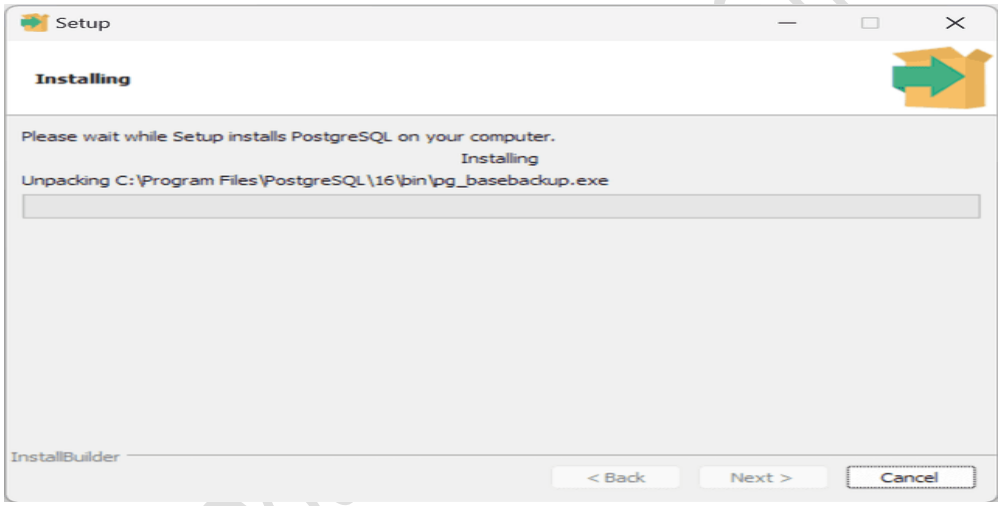
Step 9. The setup wizard will show the summary PostgreSQL information. Review the details, and if everything is correct, click the Next button. Otherwise, click the Back button to adjust the configuration accordingly.



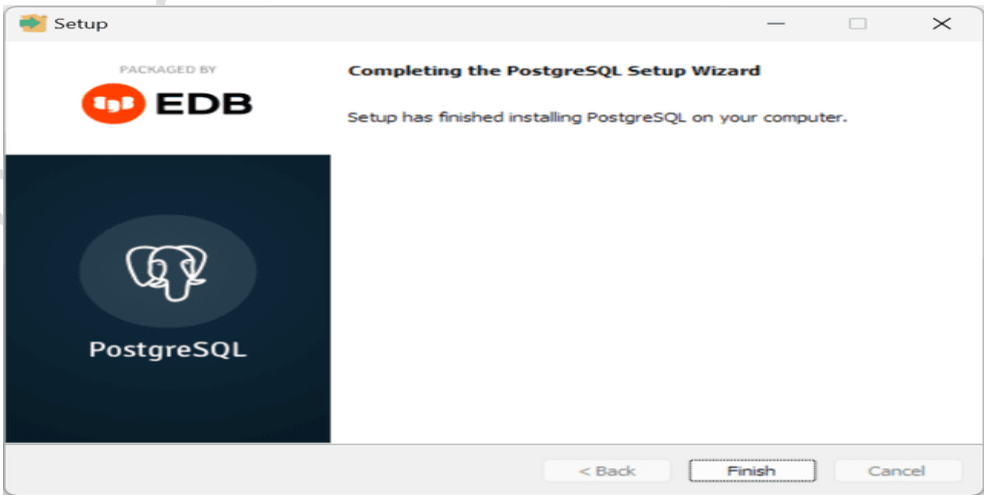
Now, you are ready to install PostgreSQL on your computer. Click the **Next** button to initiate PostgreSQL installation.



The installation may take a few minutes to complete.



Step 10. Click the **Finish** button to complete the PostgreSQL installation.



🛠️ What Is the PATH Environment Variable?

The **PATH** environment variable is a system setting that tells your operating system where to look for executable files. When you type a command in the terminal (like psql or pg_restore), Windows searches through the directories listed in the PATH variable to find the corresponding executable.

By adding PostgreSQL's bin directory to the PATH, you can run PostgreSQL tools from any command prompt window without needing to navigate to the installation folder.

📍 Why Add PostgreSQL's bin Directory to PATH?

Adding the PostgreSQL bin directory to your PATH allows you to:

- Run tools like psql, pg_dump, pg_restore, and createdb from any command line location.
- Avoid typing long paths every time you want to use PostgreSQL utilities.
- Simplify scripting and automation tasks involving PostgreSQL.

📂 Locate the PostgreSQL bin Directory

After installing PostgreSQL, the bin directory is typically located at:

C:\Program Files\PostgreSQL\<version>\bin

Replace <version> with the version number you installed. For example:

C:\Program Files\PostgreSQL\16\bin

This folder contains all the command-line tools provided by PostgreSQL.

🧭 Steps to Add the bin Directory to PATH on Windows

1. Open System Properties

- Press **Win + R** to open the Run dialog.
- Type sysdm.cpl and press **Enter**.
- The **System Properties** window will open.

2. Access Environment Variables

- Click the **Advanced** tab.
- Click the **Environment Variables...** button near the bottom.

3. Choose the PATH Variable

You'll see two sections:

- **User variables for [YourUsername]**
- **System variables**

Choose based on your needs:

- **User PATH**: Affects only your user account.
- **System PATH**: Affects all users on the system.

Select the appropriate **Path** variable and click **Edit**.

4. Add the PostgreSQL bin Directory

- In the **Edit Environment Variable** window, click **New**.
- Paste the path to the PostgreSQL bin directory:
- C:\Program Files\PostgreSQL\16\bin
- Click **OK** to confirm.

5. Finalize Changes

- Click **OK** to close the Environment Variables window.
- Click **OK** again to close the System Properties window.

✅ **Verifying the PATH Update**

To confirm that the PATH update was successful:

1. Open **Command Prompt**.
2. Type:
3. psql --version
4. If the PATH was set correctly, you'll see the PostgreSQL version output.

🧩 **Troubleshooting Tips**

- **Permission Issues**: If you can't edit the System PATH, try editing the User PATH instead.
- **Multiple PostgreSQL Versions**: Ensure the correct version's bin directory is listed first in PATH.
- **Spaces in Path**: Always enclose paths in double quotes when using them in scripts.

Post-Installation and Post-Configuration for PostgreSQL: Hardening and Access Management

After the initial installation of PostgreSQL, a series of crucial post-installation and post-configuration steps are essential to secure your database, manage user access, and enable necessary functionalities like remote connections. These steps transition your raw PostgreSQL installation into a robust, functional, and secure database environment.

1. Setting a Strong Password for the postgres Superuser

The postgres user is the default superuser for your PostgreSQL installation, possessing all administrative privileges. By default, on Linux systems, this user might be configured for peer or trust authentication for local connections, meaning no password is required when connecting as the system user postgres. This is highly insecure for production or even development environments where multiple users might access the system.

Action: Set a robust password for the postgres database superuser.

1. **Access the PostgreSQL Command-Line Interface (psql):** You need to connect to the psql shell as the postgres system user, which typically has elevated privileges to connect to the database without a password initially.

```
sudo -u postgres psql
```

   - sudo -u postgres: Executes the following command (psql) as the postgres system user. This is a common and secure way to gain initial access to the PostgreSQL database as the postgres superuser on Linux.

2. **Set the Password for the postgres Database User:** Within the psql prompt, execute the ALTER USER command. It's highly recommended to use ENCRYPTED PASSWORD for better security, although PASSWORD (which implies SCRAM-SHA-256 by default in modern PostgreSQL versions) is also secure. ENCRYPTED PASSWORD explicitly uses the configured password encryption method (e.g., SCRAM-SHA-256 or MD5).

```
ALTER USER postgres WITH ENCRYPTED PASSWORD 'your_secure_password';
```

   - **ALTER USER postgres**: Targets the database user named postgres.
   - **WITH ENCRYPTED PASSWORD 'your_secure_password'**: Assigns a new password. **Crucially, replace 'your_secure_password' with a genuinely strong, unique password.** A strong password combines uppercase and

lowercase letters, numbers, and special characters, and is of considerable length.

- **Password Storage:** Modern PostgreSQL versions (10+) default to SCRAM-SHA-256 for password hashing, which is significantly more secure than MD5. If your password_encryption setting in postgresql.conf is still md5, then ENCRYPTED PASSWORD will use MD5. It's recommended to update password_encryption to scram-sha-256 if it's not already.

3. **Exit psql:**

\q

Or simply press Ctrl + D.

2. Configuring Client Authentication (pg_hba.conf)

The pg_hba.conf (Host-Based Authentication) file is the cornerstone of PostgreSQL's client authentication system. It dictates which hosts can connect to your PostgreSQL database, which users they can connect as, which databases they can access, and what authentication methods are required. Correctly configuring this file is paramount for security.

Location:

- **Linux (Ubuntu/Debian):** Typically /etc/postgresql/<version>/main/pg_hba.conf (e.g., /etc/postgresql/16/main/pg_hba.conf).
- **Windows:** Usually within the data directory, such as C:\Program Files\PostgreSQL\<version>\data\pg_hba.conf.

Action: Modify pg_hba.conf to enforce password authentication for local connections.

1. **Open pg_hba.conf for Editing:** Use a text editor with administrative privileges.

On Linux

sudo nano /etc/postgresql/16/main/pg_hba.conf

(Replace 'nano' with your preferred editor, and '16' with your PostgreSQL version)

2. **Modify Authentication for Local Connections (specifically for postgres user):** Look for lines related to local connections or connections from 127.0.0.1 (localhost). You might find entries that look like:
3. # TYPE DATABASE USER ADDRESS METHOD
4. local all postgres peer

Or

host all all 127.0.0.1/32 trust

Change the authentication method for the postgres user on local or 127.0.0.1/32 to md5 (or scram-sha-256 for newer PostgreSQL versions if supported by your clients). md5 (and especially scram-sha-256) requires a password.

Example Modified Entry:

TYPE DATABASE USER ADDRESS METHOD

local all postgres md5

For IPv4 local connections:

host all postgres 127.0.0.1/32 md5

For IPv6 local connections:

host all postgres ::1/128 md5

- **local**: Refers to connections made via Unix domain sockets (most common for local connections on Linux).
- **host**: Refers to connections made via TCP/IP.
- **all**: Applies to all databases or all users, depending on its column.
- **postgres**: Specifically targets the postgres user.
- **md5**: Specifies that the client must provide an MD5-hashed password.
- **scram-sha-256**: (Recommended for PostgreSQL 10+) Specifies SCRAM-SHA-256 password authentication, which is more secure than MD5. Ensure your client libraries support it.

Understanding pg_hba.conf Lines: Each line in pg_hba.conf defines a rule: [CONNECTION_TYPE] [DATABASE] [USER] [ADDRESS] [AUTHENTICATION_METHOD] [OPTIONS] Rules are processed in order. The first rule that matches the connection attempt is used.

5. **Save Changes and Restart PostgreSQL Service:** For pg_hba.conf changes to take effect, you *must* restart the PostgreSQL service.

On Linux (systemd-based systems like Ubuntu 16.04+)

sudo systemctl restart postgresql

On older Linux systems (SysVinit)

sudo /etc/init.d/postgresql restart

On Windows (via Services Manager or command line)

Open Services (services.msc), find "postgresql-<version>", and restart it.

Or from an elevated Command Prompt:

net stop postgresql-<version>

net start postgresql-<version>

3. Creating Additional Users and Databases (Recommended Best Practice)

It's a critical security and operational best practice to avoid using the postgres superuser for routine application or development tasks. Instead, create dedicated users with the minimum necessary privileges for specific applications or individuals.

Action: Connect as postgres and create new users and databases.

1. **Connect to psql as the postgres superuser:**

psql -U postgres

Or, if you've set up your system to require a password for postgres:

psql -U postgres -h localhost

(It will prompt for password)

2. **Create New Database Users:** Grant only the necessary privileges. Avoid giving SUPERUSER or CREATEDB/CREATEROLE privileges unless absolutely required.

CREATE USER myappuser WITH ENCRYPTED PASSWORD 'a_strong_password_for_app';

-- Example with additional roles (optional)

-- CREATE USER myreadonlyuser WITH ENCRYPTED PASSWORD 'another_strong_password';

-- GRANT SELECT ON ALL TABLES IN SCHEMA public TO myreadonlyuser;

- **myappuser**: Replace with a descriptive username for your application or user.
- **ENCRYPTED PASSWORD '...'**: Again, use a strong password.

3. **Create New Databases and Assign Ownership:** Create databases for your applications and assign ownership to the newly created users.

CREATE DATABASE myappdb OWNER myappuser;

-- Example for a testing database

-- CREATE DATABASE testdb OWNER myappuser;

- **myappdb**: Replace with your desired database name.
- **OWNER myappuser**: Assigns myappuser as the owner of myappdb, giving them full control over objects within that database.

4. **Exit psql:**

\q

4. Enabling Remote Connections (If Needed)

By default, PostgreSQL is configured to only accept connections from localhost (the machine it's running on) for security reasons. If your application or another client needs to connect to the database from a different machine, you must configure PostgreSQL to listen on external network interfaces.

Action: Modify postgresql.conf and pg_hba.conf to allow remote connections.

1. **Edit postgresql.conf:** This file controls the global configuration parameters of your PostgreSQL server.

Location: Same directory as pg_hba.conf (e.g., /etc/postgresql/<version>/main/postgresql.conf on Linux).

On Linux

sudo nano /etc/postgresql/16/main/postgresql.conf

Find and modify listen_addresses: Uncomment the listen_addresses line (remove the # at the beginning) and set its value.

What IP address(es) to listen on; '*' means all IPv4 and IPv6 interfaces

#listen_addresses = 'localhost' # comma-separated list of addresses;

defaults to 'localhost'; use '*' for all

listen_addresses = '*'

- **listen_addresses = '*'**: This setting tells PostgreSQL to listen for connections on all available network interfaces (IPv4 and IPv6). This is the

simplest way to enable remote connections but also the least secure as it opens up the server to the entire network.

- **listen_addresses = '192.168.1.100'**: For better security, specify a comma-separated list of exact IP addresses that PostgreSQL should listen on. This limits the attack surface.

2. **Adjust pg_hba.conf for Remote Connections:** Even after setting listen_addresses, connections will be denied unless pg_hba.conf explicitly permits them. You need to add rules that match the incoming remote connections.

sudo nano /etc/postgresql/16/main/pg_hba.conf

Add new host entries for remote access:

TYPE DATABASE USER ADDRESS METHOD

Allow myappuser to connect to myappdb from any host in the 192.168.1.0/24 subnet, requiring a password

host myappdb myappuser 192.168.1.0/24 md5

Allow the postgres superuser to connect from a specific management IP, requiring a password

host all postgres 10.0.0.5/32 md5

More permissive (use with caution!): Allow any user to connect to any database from any IP

host all all 0.0.0.0/0 md5

- **ADDRESS**:
  - 192.168.1.0/24: Allows connections from any IP address within the 192.168.1.x subnet.
  - 10.0.0.5/32: Allows connections only from the specific IP address 10.0.0.5. The /32 indicates a single host.
  - 0.0.0.0/0: Allows connections from *any* IPv4 address. **Use with extreme caution and only if absolutely necessary, combined with strong passwords and firewall rules.**

- **METHOD**: Always use md5 or scram-sha-256 for remote connections. Never use trust for remote connections in production.

3. **Save Changes and Restart PostgreSQL Service:** Both postgresql.conf and pg_hba.conf changes require a service restart.

sudo systemctl restart postgresql

5. Firewall Configuration (If Enabling Remote Connections)

Even if PostgreSQL is configured to listen on external interfaces and pg_hba.conf permits remote connections, your operating system's firewall can block incoming connection attempts. You must explicitly open the PostgreSQL port (default: 5432) in your firewall rules.

Action: Open port 5432 in your system's firewall.

- **Linux (using ufw - Uncomplicated Firewall, common on Ubuntu):**

sudo ufw allow 5432/tcp

sudo ufw enable # If ufw is not already enabled

sudo ufw status verbose

- ufw allow 5432/tcp: Allows incoming TCP connections on port 5432.
- For enhanced security, you can restrict the source IP:

sudo ufw allow from 192.168.1.0/24 to any port 5432

- **Windows (Windows Defender Firewall):**

1. Open "Windows Defender Firewall with Advanced Security" (search in Start Menu).
2. In the left pane, click "Inbound Rules."
3. In the right pane, click "New Rule..."
4. Select "Port," then "Next."
5. Select "TCP," enter 5432 for "Specific local ports," then "Next."
6. Select "Allow the connection," then "Next."
7. Choose when the rule applies ("Domain," "Private," "Public" - select as appropriate for your network).

8. Give the rule a name (e.g., "PostgreSQL Inbound 5432") and a description, then "Finish."

Edit postgresql.conf

Now edit the postgresql.conf file...

```
sudo vim /etc/postgresql/9.*/main/postgresql.conf
```

... and add the following line:

```
listen_addresses = '*'
```

Restart The Service

Now restart PostgreSQL for the changes to take effect.

```
sudo /etc/init.d/postgresql restart
```

Test It

To check that everything went correctly, try to make a connection to your local server by using your servers IP as below:

```
psql -U postgres -h [Server IP]
```

You should be prompted for a password, which only if you enter correctly, you will gain access.

Summary of Post-Installation Security and Management:

By meticulously following these steps, you will transform your basic PostgreSQL installation into a more secure, accessible, and manageable database server. Remember to always apply the principle of least privilege: grant users only the permissions they absolutely need. Regularly review your pg_hba.conf and postgresql.conf files, and ensure your firewall rules align with your security posture.

How to Allow Remote Connection to PostgreSQL Database using psql

🌐 Enabling Remote TCP/IP Connections in PostgreSQL

By default, PostgreSQL is configured to accept connections **only from the local machine**. If you attempt to connect remotely using tools like psql, you may encounter the following error:

bash

psql: could not connect to server: Connection refused

Is the server running on host "192.168.102.1" and accepting

TCP/IP connections on port 5432?

To resolve this and allow remote connections, you need to modify two key configuration files on the PostgreSQL server:

1. pg_hba.conf – Controls client authentication.
2. postgresql.conf – Controls server listening behavior.

🛠️ Step 1: Modify pg_hba.conf to Allow Remote Clients

📍 Location of pg_hba.conf

Typical path:

/var/lib/pgsql/data/pg_hba.conf

Or for newer versions:

/etc/postgresql/<version>/main/pg_hba.conf

📄 Default Configuration

conf

IPv4 local connections:

host all all 127.0.0.1/32 trust

IPv6 local connections:

host all all ::1/128 ident

These entries only allow connections from localhost.

📝 Add Remote Access Entry

To allow a specific client IP (e.g., 192.168.101.20) or a subnet (e.g., 192.168.101.0/24), add:

conf

host all all 192.168.101.0/24 trust

🔐 **Note**: The trust method allows connections without a password. For production environments, use md5 or scram-sha-256 for secure authentication.

💻 **Script to Append Entry**

```
echo "host    all    all    192.168.101.0/24    trust" | sudo tee -a /var/lib/pgsql/data/pg_hba.conf
```

🛠️ **Step 2: Update postgresql.conf to Listen on All IPs**

📍 **Location of postgresql.conf**

Typical path:

/var/lib/pgsql/data/postgresql.conf

Or:

/etc/postgresql/<version>/main/postgresql.conf

🔍 **Check Current Setting**

```
grep listen_addresses /var/lib/pgsql/data/postgresql.conf
```

You'll likely see:

conf

```
listen_addresses = 'localhost'
```

📝 **Modify to Accept Remote Connections**

Change it to:

conf

```
listen_addresses = '*'
```

Or specify a particular IP:

conf

```
listen_addresses = '192.168.102.1'
```

💻 **Script to Update Setting**

```
sudo sed -i "s/^#listen_addresses = 'localhost'/listen_addresses = '*'/g" /var/lib/pgsql/data/postgresql.conf
```

```
sudo sed -i "s/^listen_addresses = 'localhost'/listen_addresses = '*'/g" /var/lib/pgsql/data/postgresql.conf
```

🔄 Step 3: Restart PostgreSQL Service

After making changes, restart the PostgreSQL service to apply them:

```
sudo systemctl restart postgresql
```

Or for older systems:

```
sudo service postgresql restart
```

✅ Step 4: Verify Remote Connection

From the client machine, test the connection:

```
psql -U postgres -h 192.168.102.1
```

If everything is configured correctly, you should be prompted for a password (unless using trust), and then connected to the database.

🔐 Security Considerations

- Use **firewall rules** to restrict access to PostgreSQL port (default: 5432).
- Avoid using trust in production; prefer md5 or scram-sha-256.
- Consider using **SSL** for encrypted connections (hostssl in pg_hba.conf).
