Configuring PostgreSQL for Network Communication

It is not common or advisable to have a database (like PostgreSQL) publicly accessible over the internet. Instead, databases are typically confined to a local network (LAN) or even localhost on the same machine running the web server. Where remote access is required, it should be done securely, such as through a VPN or SSH tunnel.

The following two PostgreSQL configuration files need to be configured to allow incoming connections from outside the device running the PostgreSQL server:

* **`postgresql.conf`**
* **`pg\_hba.conf`**

## Configuring `postgresql.conf`

***Path: ` /etc/postgresql/[version]/main/postgresql.conf`***

The `postgresql.conf` file is the primary configuration file for PostgreSQL. It contains settings that control the overall behaviour of the PostgreSQL server, such as memory usage, logging, connection limits, and network settings.

**The `postgresql.conf` file contains a `listen\_addresses` field. This field specifies the IP addresses the PostgreSQL server should listen on for incoming connections.**

***\*This is NOT the IP addresses of hosts connecting to the PostgreSQL server. This is the IP address/es assigned to the host (device) running the PostgreSQL server from any of its network interfaces.***

For example, if the host is connected to two different networks, but you only want the PostgreSQL server to listen for incoming connections from one of these networks, you would assign the IP address assigned to the host in this network to `listen\_addresses`.

For example, if I have connected two of my hosts (my MacBook and HP laptop) using a crossover cable, I am running PostgreSQL on my MacBook, and I only want the HP laptop to be able to access the PostgreSQL server, I would add the following IPv4 address to `listen\_addresses`.

**192.168.0.200**

This is the MacBook's IP address on the interface `en10`. The reason that the IP address is used rather than the interface name is that in certain network configurations, a host can have multiple IP addresses on the same physical interface (such as with VLANs).

The following is a section from the `postgresql.conf` file after a fresh installation:

#------------------------------------------------------------------------------

# CONNECTIONS AND AUTHENTICATION

#------------------------------------------------------------------------------

# - Connection Settings -

#listen\_addresses = 'localhost' # what IP address(es) to listen on;

# comma-separated list of addresses;

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

# (change requires restart)

port = 5432 # (change requires restart)

max\_connections = 100 # (change requires restart)

#reserved\_connections = 0 # (change requires restart)

#superuser\_reserved\_connections = 3 # (change requires restart)

unix\_socket\_directories = '/var/run/postgresql' # comma-separated list of directories

As can be seen, the `listen\_addresses` field is commented out by default. In this case, the default behaviour of PostgreSQL is to only listen to incoming connections on the loopback address (i.e., `127.0.0.1`, or simply `localhost`).

In this instance I am configuring the PostgreSQL server running on the Raspberry Pi to accept all incoming connections from hosts in the home LAN. The Raspberry Pi has an IP address of `192.168.0.20` in this network, so this should be added to the `listen\_addresses` (and the field should be uncommented).

#------------------------------------------------------------------------------

# CONNECTIONS AND AUTHENTICATION

#------------------------------------------------------------------------------

# - Connection Settings -

listen\_addresses = 'localhost,192.168.0.20' # what IP address(es) to listen on;

# comma-separated list of addresses;

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

Alternatively, if you want the PostgreSQL server to listen for incoming connections on all IP addresses assigned to the host on all of its network interfaces, you can assign **`\*`** to `listen\_addresses`, resulting in the following configuration:

#------------------------------------------------------------------------------

# CONNECTIONS AND AUTHENTICATION

#------------------------------------------------------------------------------

# - Connection Settings -

listen\_addresses = '\*' # what IP address(es) to listen on;

# comma-separated list of addresses;

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

## Configuring `pg\_hba.conf`

***Path: `/etc/postgresql/[version]/main/pg\_hba.conf`***

The `pg\_hba.conf` file (HBA stands for Host-Based Authentication) is used to control client authentication. It defines which users can connect to which databases, from which hosts, and which authentication methods should be used.

The `pg\_hba.conf` file is organised into lines, where each line defines a rule for allowing or denying access based on various criteria. Here are the key fields that make up each rule, in order:

*\*There are more detailed notes on this in the `pg\_hba.conf` file itself. These notes are shortened for conciseness.*

1. **Connection Type (`TYPE`)**

This rule specifies the type of connection.

Common values include:

* **`local`**: A connection over a Unix socket (local to the machine).
* **`host`**: A TCP/IP socket (encrypted or not).
* **`hostssl`**: A TCP/IP socket that is SSL-encrypted.
* **`hostnossl`**: A TCP/IP socket that is not SSL-encrypted.

1. **Database (`DATABASE`)**

Specifies the database(s) to which the rule applies.

Common values include:

* + **`all`**: Applies to all databases.
  + **Specific database name**: Applies to a particular database.
  + **`sameuser`**: Allows the user to connect only to a database with the same name as their username.
  + **`samerole`**: Allows the user to connect to any database for which they have a role with the same name.

1. **User (`USER`)**

Defines the PostgreSQL user(s) the rule applies to.

Common values include:

* **`all`**: Applies to all users.
* **Specific username**: Applies to a particular user or users.

1. **Address (`ADDRESS`)**

Specifies the client IP address or address range from which the connection is allowed or denied.

This field is only used for `host`, `hostssl`, and `hostnossl` connection types.

Format:

* A single IPv4 address: `192.168.1.50`
* An IP address range using CIDR notation: `192..168.1.0/24`
* For IPv6: `2001:db8::/32`
* `all` for allowing connections from any IP address (less secure).

1. **Authentication Method (`METHOD`)**

Specifies the authentication method used to verify the user's identity.

Common methods include:

* **`trust`**: Allows connections without a password (not secure).
* **`md5`**: Requires an MD5-hashed password for authentication.
* **`password`**: Requires a plaintext password (not secure over unencrypted connections).
* **`peer`**: Uses the operating system username for authentication (commonly used with Unix sockets).
* **`scram-sha-256**`: Uses the more secure SCRAM-SHA-256 hashing for passwords.
* **`ident`**: Uses an external service to verify the client's identity based on their IP address.

The format of an entry in the `pg\_hba.conf` file is demonstrated below:

# TYPE DATABASE USER ADDRESS METHOD

local all all peer

host all all 192.168.0.0/24 md5

**The first entry**:

* Applies to local (Unix socket) connections to all databases from all users. It states that they must use the operating system's username for authentication.

**The second entry**:

* Applies to all unencrypted TCP/IP connections to all databases from all users. It only allows connections from IP addresses within the `192.168.0.0/24` subnet and requires an MD5-hashed password for authentication.

**Placement in the `pg\_hba.conf` File**:  
Each rule you define in the `pg\_hba.conf` file is written on a new line, following this structure. The order of the lines matters because PostgreSQL processes these rules in a sequence, from top to bottom. The first rule that matches a connection attempt is applied, and no further rules are checked for that connection.

### How PostgreSQL Authentication is Handled

The client user does not need to manually encrypt or hash the password before sending it to the server. The client software (e.g., `psql` or a PostgreSQL driver in an application) handles this automatically.

When you configure the client to connect to a PostgreSQL server, you typically provide the password in plaintext. The client software takes care of processing this password according to the method specified by the server in `pg\_hba.conf`.

### When to Use Weaker Authentication Methods

The additional computing load of using `scram-sha-256` is minimal, especially on modern hardware, and is unlikely to have a noticeable impact on performance, even in a busy environment. Therefore, unless there is a specific reason to use a weaker method, it is recommended to use `scram-sha-256` for all connections, including those from your LAN.

An example of a scenario where you might need to use a weaker encryption method is to facilitate compatibility with an older client that doesn't support `scram-sha-256`.

### Choosing Between `host` and `hostssl`

#### `hostssl`

Using `hostssl` specifies that connections should be made over TCP/IP and MUST use SSL/TLS encryption. This encrypts the entire communication channel between the client and the server, ensuring that even if the data is intercepted, it cannot be easily read or tampered with.

Using `hostssl` is particularly recommended if your LAN has wireless connections (e.g., a Wi-Fi network), or is shared with other devices/users that might not be fully trusted.

For `hostssl`, additional configuration is required in the `postgresql.conf` file, as well as setting up certificates.

1. SSL certificates must be generated - this can be done using the `openssl` utility.
2. SSL can be enabled by setting the following parameters in `postgresql.conf`:

ssl = on

ssl\_cert\_file = '/path/to/server.crt'

ssl\_key\_file = '/path/to/server.key'

Since this database does not contain any sensitive information and is currently being used for learning purposes only, it is safe to allow unencrypted connections for now. Therefore, the following rule will be added to `pg\_hba.conf`:

host all all 192.168.0.0/24 scram-sha-256

## Connecting to PostgreSQL Server

Once all the above configurations have been applied, you can connect to the server from the remote host using the following command:

psql -h <host> -p <port> -U <username> -d <database>

By default, `psql` will ask for a password if the server requires password authentication.