20.1. The pg_hba.conf File

Client authentication is controlled by a configuration file, which traditionally is named `pg_hba.conf` and is stored in the database cluster's data directory. (HBA stands for host-based authentication.) A default `pg_hba.conf` file is installed when the data directory is initialized by `initdb`. It is possible to place the authentication configuration file elsewhere, however; see the hba_file configuration parameter.

The general format of the `pg_hba.conf` file is a set of records, one per line. Blank lines are ignored, as is any text after the # comment character. Records cannot be continued across lines. A record is made up of a number of fields which are separated by spaces and/or tabs. Fields can contain white space if the field value is double-quoted. Quoting one of the keywords in a database, user, or address field (e.g., `all` or `replication`) makes the word lose its special meaning, and just match a database, user, or host with that name.

Each record specifies a connection type, a client IP address range (if relevant for the connection type), a database name, a user name, and the authentication method to be used for connections matching these parameters. The first record with a matching connection type, client address, requested database, and user name is used to perform authentication. There is no "fall-through" or "backup": if one record is chosen and the authentication fails, subsequent records are not considered. If no record matches, access is denied.

A record can have one of the seven formats

```
local      database  user  auth-method  [auth-options]
host       database  user  address  auth-method  [auth-options]
hostssl    database  user  address  auth-method  [auth-options]
hostnossl  database  user  address  auth-method  [auth-options]
host       database  user  IP-address  IP-mask  auth-method  [auth-options]
hostssl    database  user  IP-address  IP-mask  auth-method  [auth-options]
hostnossl  database  user  IP-address  IP-mask  auth-method  [auth-options]
```

The meaning of the fields is as follows:

`local`

This record matches connection attempts using Unix-domain sockets. Without a record of this type, Unix-domain socket connections are disallowed.

`host`

This record matches connection attempts made using TCP/IP. `host` records match either SSL or non-SSL connection attempts.

Note: Remote TCP/IP connections will not be possible unless the server is started with an appropriate value for the listen_addresses configuration parameter, since the default behavior is to listen for TCP/IP connections only on the local loopback address `localhost`.

`hostssl`

This record matches connection attempts made using TCP/IP, but only when the connection is made with SSL encryption.

To make use of this option the server must be built with SSL support. Furthermore, SSL must be enabled at server start time by setting the ssl configuration parameter (see Section 18.9 for more information).

`hostnossl`

This record type has the opposite behavior of `hostssl`; it only matches connection attempts made over TCP/IP that do not use SSL.

`database`

Specifies which database name(s) this record matches. The value `all` specifies that it matches all databases. The value `sameuser` specifies that the record matches if the requested database has the same name as the requested user. The value `samerole` specifies that the requested user must be a member of the role with the same name as the requested database. (`samegroup` is an obsolete but still accepted spelling of `samerole`.) Superusers are not considered to be members of a role for the purposes of `samerole` unless they are explicitly members of the role, directly or indirectly, and not just by virtue of being a superuser. The value `replication` specifies that the record matches if a replication connection is requested (note that replication connections do not specify any particular database). Otherwise, this is the name of a specific PostgreSQL database. Multiple database names can be supplied by separating them with commas. A separate file containing database names can be specified by preceding the file name with `@`.

`user`

Specifies which database user name(s) this record matches. The value `all` specifies that it matches all users. Otherwise, this is either the name of a specific database user, or a group name preceded by +. (Recall that there is no real distinction between users and groups in PostgreSQL; a + mark really means "match any of the roles that are directly or indirectly members of this role", while a name without a + mark matches only that specific role.) For this purpose, a superuser is only considered to be a member of a role if they are explicitly a member of the role, directly or indirectly, and not just by virtue of being a superuser. Multiple user names can be supplied by separating them with commas. A separate file containing user names can be specified by preceding the file name with `@`.

`address`

Specifies the client machine address(es) that this record matches. This field can contain either a host name, an IP address range, or one of the special key words mentioned below.

An IP address range is specified using standard numeric notation for the range's starting address, then a slash (`/`) and a CIDR mask length. The mask length indicates the number of high-order bits of the client IP address that must match. Bits to the right of this should be zero in the given IP address. There must not be any white space between the IP address, the `/`, and the CIDR mask length.

Typical examples of an IPv4 address range specified this way are `172.20.143.89/32` for a single host, or `172.20.143.0/24` for a small network, or `10.6.0.0/16` for a larger one. An IPv6 address range might look like `::1/128` for a single host (in this case the IPv6 loopback address) or `fe80::7a31:c1ff:0000:0000/96` for a small network. `0.0.0.0/0` represents all IPv4 addresses, and `::0/0` represents all IPv6 addresses. To specify a single host, use a mask length of 32 for IPv4 or 128 for IPv6. In a network address, do not omit trailing zeroes.

An entry given in IPv4 format will match only IPv4 connections, and an entry given in IPv6 format will match only IPv6 connections, even if the represented address is in the IPv4-in-IPv6 range. Note

that entries in IPv6 format will be rejected if the system's C library does not have support for IPv6 addresses.

You can also write `all` to match any IP address, `samehost` to match any of the server's own IP addresses, or `samenet` to match any address in any subnet that the server is directly connected to.

If a host name is specified (anything that is not an IP address range or a special key word is treated as a host name), that name is compared with the result of a reverse name resolution of the client's IP address (e.g., reverse DNS lookup, if DNS is used). Host name comparisons are case insensitive. If there is a match, then a forward name resolution (e.g., forward DNS lookup) is performed on the host name to check whether any of the addresses it resolves to are equal to the client's IP address. If both directions match, then the entry is considered to match. (The host name that is used in `pg_hba.conf` should be the one that address-to-name resolution of the client's IP address returns, otherwise the line won't be matched. Some host name databases allow associating an IP address with multiple host names, but the operating system will only return one host name when asked to resolve an IP address.)

A host name specification that starts with a dot (`.`) matches a suffix of the actual host name. So `.example.com` would match `foo.example.com` (but not just `example.com`).

When host names are specified in `pg_hba.conf`, you should make sure that name resolution is reasonably fast. It can be of advantage to set up a local name resolution cache such as `nscd`. Also, you may wish to enable the configuration parameter `log_hostname` to see the client's host name instead of the IP address in the log.

This field only applies to `host`, `hostssl`, and `hostnossl` records.

> Users sometimes wonder why host names are handled in this seemingly complicated way, with two name resolutions including a reverse lookup of the client's IP address. This complicates use of the feature in case the client's reverse DNS entry is not set up or yields some undesirable host name. It is done primarily for efficiency: this way, a connection attempt requires at most two resolver lookups, one reverse and one forward. If there is a resolver problem with some address, it becomes only that client's problem. A hypothetical alternative implementation that only did forward lookups would have to resolve every host name mentioned in `pg_hba.conf` during every connection attempt. That could be quite slow if many names are listed. And if there is a resolver problem with one of the host names, it becomes everyone's problem.
>
> Also, a reverse lookup is necessary to implement the suffix matching feature, because the actual client host name needs to be known in order to match it against the pattern.
>
> Note that this behavior is consistent with other popular implementations of host name-based access control, such as the Apache HTTP Server and TCP Wrappers.

`IP-address`
`IP-mask`

These two fields can be used as an alternative to the `IP-address/mask-length` notation. Instead of specifying the mask length, the actual mask is specified in a separate column. For example, `255.0.0.0` represents an IPv4 CIDR mask length of 8, and `255.255.255.255` represents a CIDR mask length of 32.

These fields only apply to `host`, `hostssl`, and `hostnossl` records.

`auth-method`

Specifies the authentication method to use when a connection matches this record. The possible choices are summarized here; details are in Section 20.3.

`trust`

Allow the connection unconditionally. This method allows anyone that can connect to the PostgreSQL database server to login as any PostgreSQL user they wish, without the need for a password or any other authentication. See Section 20.3.1 for details.

`reject`

Reject the connection unconditionally. This is useful for "filtering out" certain hosts from a group, for example a `reject` line could block a specific host from connecting, while a later line allows the remaining hosts in a specific network to connect.

`md5`

Require the client to supply a double-MD5-hashed password for authentication. See Section 20.3.2 for details.

`password`

Require the client to supply an unencrypted password for authentication. Since the password is sent in clear text over the network, this should not be used on untrusted networks. See Section 20.3.2 for details.

`gss`

Use GSSAPI to authenticate the user. This is only available for TCP/IP connections. See Section 20.3.3 for details.

`sspi`

Use SSPI to authenticate the user. This is only available on Windows. See Section 20.3.4 for details.

`ident`

Obtain the operating system user name of the client by contacting the ident server on the client and check if it matches the requested database user name. Ident authentication can only be used on TCP/IP connections. When specified for local connections, peer authentication will be used instead. See Section 20.3.5 for details.

`peer`

Obtain the client's operating system user name from the operating system and check if it matches the requested database user name. This is only available for local connections. See Section 20.3.6 for details.

`ldap`

Authenticate using an LDAP server. See Section 20.3.7 for details.

`radius`

Authenticate using a RADIUS server. See Section 20.3.8 for details.

`cert`

Authenticate using SSL client certificates. See Section 20.3.9 for details.

`pam`

Authenticate using the Pluggable Authentication Modules (PAM) service provided by the operating system. See Section 20.3.10 for details.

`bsd`

Authenticate using the BSD Authentication service provided by the operating system. See Section 20.3.11 for details.

`auth-options`

After the `auth-method` field, there can be field(s) of the form `name=value` that specify options for the authentication method. Details about which options are available for which authentication methods appear below.

In addition to the method-specific options listed below, there is one method-independent authentication option `clientcert`, which can be specified in any `hostssl` record. When set to `1`, this option requires the client to present a valid (trusted) SSL certificate, in addition to the other requirements of the authentication method.

Files included by `@` constructs are read as lists of names, which can be separated by either whitespace or commas. Comments are introduced by `#`, just as in `pg_hba.conf`, and nested `@` constructs are allowed. Unless the file name following `@` is an absolute path, it is taken to be relative to the directory containing the referencing file.

Since the `pg_hba.conf` records are examined sequentially for each connection attempt, the order of the records is significant. Typically, earlier records will have tight connection match parameters and weaker authentication methods, while later records will have looser match parameters and stronger authentication methods. For example, one might wish to use `trust` authentication for local TCP/IP connections but require a password for remote TCP/IP connections. In this case a record specifying `trust` authentication for connections from 127.0.0.1 would appear before a record specifying password authentication for a wider range of allowed client IP addresses.

The `pg_hba.conf` file is read on start-up and when the main server process receives a SIGHUP signal. If you edit the file on an active system, you will need to signal the postmaster (using `pg_ctl reload` or `kill -HUP`) to make it re-read the file.

Tip: To connect to a particular database, a user must not only pass the `pg_hba.conf` checks, but must have the `CONNECT` privilege for the database. If you wish to restrict which users can connect to which databases, it's usually easier to control this by granting/revoking `CONNECT` privilege than to put the rules in `pg_hba.conf` entries.

Some examples of `pg_hba.conf` entries are shown in Example 20-1. See the next section for details on the different authentication methods.

Example 20-1. Example `pg_hba.conf` Entries

```
# Allow any user on the local system to connect to any database with
# any database user name using Unix-domain sockets (the default for local
# connections).
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             all                                     trust

# The same using local loopback TCP/IP connections.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             127.0.0.1/32            trust

# The same as the previous line, but using a separate netmask column
#
# TYPE  DATABASE        USER            IP-ADDRESS      IP-MASK             METHOD
host    all             all             127.0.0.1       255.255.255.255     trust
```

```
# The same over IPv6.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             ::1/128                 trust

# The same using a host name (would typically cover both IPv4 and IPv6).
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             localhost               trust

# Allow any user from any host with IP address 192.168.93.x to connect
# to database "postgres" as the same user name that ident reports for
# the connection (typically the operating system user name).
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    postgres        all             192.168.93.0/24         ident

# Allow any user from host 192.168.12.10 to connect to database
# "postgres" if the user's password is correctly supplied.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    postgres        all             192.168.12.10/32        md5

# Allow any user from hosts in the example.com domain to connect to
# any database if the user's password is correctly supplied.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             .example.com            md5

# In the absence of preceding "host" lines, these two lines will
# reject all connections from 192.168.54.1 (since that entry will be
# matched first), but allow GSSAPI connections from anywhere else
# on the Internet.  The zero mask causes no bits of the host IP
# address to be considered, so it matches any host.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             192.168.54.1/32         reject
host    all             all             0.0.0.0/0               gss

# Allow users from 192.168.x.x hosts to connect to any database, if
# they pass the ident check.  If, for example, ident says the user is
# "bryanh" and he requests to connect as PostgreSQL user "guest1", the
# connection is allowed if there is an entry in pg_ident.conf for map
# "omicron" that says "bryanh" is allowed to connect as "guest1".
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             192.168.0.0/16          ident map=omicron

# If these are the only three lines for local connections, they will
# allow local users to connect only to their own databases (databases
# with the same name as their database user name) except for administrators
# and members of role "support", who can connect to all databases.  The file
# $PGDATA/admins contains a list of names of administrators.  Passwords
# are required in all cases.
#
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   sameuser        all                                     md5
local   all             @admins                                 md5
local   all             +support                                md5

# The last two lines above can be combined into a single line:
local   all             @admins,+support                        md5

# The database column can also use lists and file names:
local   db1,db2,@demodbs  all                                   md5
```

Client Authentication Up User Name Maps

Submit correction

If you see anything in the documentation that is not correct, does not match your experience with the particular feature or requires further clarification, please use this form to report a documentation issue.