**How to Allow Remote Connection to PostgreSQL Database using psql**

by Ramesh Natarajan on February 5, 2014

When you install PostgreSQL, by default connection to the database using TCP/IP is not allowed.

When you try to connect from a client to a remote PostgreSQL database using psql command, you might get “psql: could not connect to server: Connection refused” error message.

In the following example, from a client machine, we are trying to connect to a PostgreSQL database that is running on 192.168.102.1 server. As you see from the output, it clearly says that the remote PostgreSQL database is not accepting connection.

# psql -U postgres -h 192.168.102.1

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 enable TCP/IP connection for PostgreSQL database, you need to follow the two steps mentioned below.

**1. Modify pg\_hba.conf to add Client Authentication Record**

On the PostgreSQL database server, by default, you’ll notice the following records towards the end of the /var/lib/pgsql/data/pg\_hba.conf. As indicated below, it accepts connections only from the localhost.

# IPv4 local connections:

host all all 127.0.0.1/32 trust

# IPv6 local connections:

host all all ::1/128 ident

Add the following line to the pg\_hba.conf server. This will allow connection from “192.168.101.20” ip-address (This is the client in our example). If you want to allow connection from multiple client machines on a specific network, specify the network address here in the CIDR-address format.

# vi /var/lib/pgsql/data/pg\_hba.conf

host all all 192.168.101.20/24 trust

The following are various client authentication record format supported in the pg\_hba.conf file. We are using the #2 format from this list.

* local database user authentication-method [authentication-option]
* host database user CIDR-address authentication-method [authentication-option]
* hostssl database user CIDR-address authentication-method [authentication-option]
* hostnossl database user CIDR-address authentication-method [authentication-option]

Instead of “CIDR-address” format, you can also specify the ip-address and the network mask in separate fields using the following record format.

* host database user IP-address IP-mask authentication-method [authentication-option]
* hostssl database user IP-address IP-mask authentication-method [authentication-option]
* hostnossl database user IP-address IP-mask authentication-method [authentication-option]

**2. Change the Listen Address in postgresql.conf**

On the PostgreSQL database server, by default, the listen address will be localhost in the postgresql.conf file as shown below.

# grep listen /var/lib/pgsql/data/postgresql.conf

listen\_addresses = 'localhost'

Modify this line and give \*. If you have multiple interfaces on the server, you can also specify a specific interface to be listened.

# grep listen /var/lib/pgsql/data/postgresql.conf

listen\_addresses = '\*'

**3. Test the Remote Connection**

Now, login to the client machine 192.168.101.20, and perform the psql remote connection to the PostgreSQL database server (192.168.102.1) as shown below. This time, it should work.

# psql -U postgres -h 192.168.102.1

Welcome to psql 8.1.11 (server 8.4.18), the PostgreSQL interactive terminal.

postgres=#

Also, if you don’t want to specify the hostname in the command line parameter every time, you can setup the remote PostgreSQL database ip-address in PGHOST environment variable name as shown below.

# export PGHOST=192.168.102.1

# psql -U postgres

Welcome to psql 8.1.11 (server 8.4.18), the PostgreSQL interactive terminal.

postgres=#