devart

How To PostgreSQL Tools

How to Configure PostgreSQL for Remote Connections: A Beginner’s Guide

By Helena Alexander February 22, 2023 5281 0

Configuring PostgreSQL for remote connections is essential for accessing data from different locations. By default, PostgreSQL only allows connections from the local machine, which can be a limitation in many situations.

Remote access to a PostgreSQL database is necessary for applications that require accessing the data from different geographical locations or for teams that are working in different parts of the world. It can also be crucial for troubleshooting issues that cannot be resolved locally or when the database administrator is working remotely. Configuring PostgreSQL for remote access enables greater flexibility, accessibility, and collaboration, which can improve productivity and efficiency in multiple use cases.

In this article, we will walk through the step-by-step process of configuring PostgreSQL for remote access, including setting up the PostgreSQL server, configuring the firewall, modifying PostgreSQL configuration files, and testing the connection with the help of an advanced Postgres GUI tool – dbForge Studio for PostgreSQL. Whether you are a database administrator, developer, or user, this article will provide you with the knowledge and tools to enable remote access to your PostgreSQL database.

Contents

Introduction to remote database connections

Remote database connections allow users to access data from a database server that is located on a different machine or network. Remote access is crucial for organizations with distributed teams or with users in different geographic locations who need to access and work with the same data. It is also important for businesses that require real-time data access for critical decision-making. Remote database connections can be established through different protocols and technologies, such as TCP/IP, SSH, or VPN, and can be configured for various database management systems, including PostgreSQL. In this era of global connectivity, remote access to databases is becoming increasingly important for organizations of all sizes.

Ports play a vital role in establishing remote database connections. A port is a communication endpoint that enables data to be sent and received between different devices. When setting up a remote connection, the database server needs to listen on a specific port to accept incoming connections from remote clients. The default port used for PostgreSQL is 5432. Other database management systems, such as MySQL and Microsoft SQL Server, have their own default port numbers that can be changed to ensure secure remote access. Ports can also be used to configure firewalls to allow or block incoming traffic to the database server. When configuring remote database connections, it is essential to ensure that the appropriate ports are open and that they are secured to prevent unauthorized access or data breaches.

Securing remote connections to PostgreSQL

Securing remote connections to a PostgreSQL database is essential to prevent unauthorized access or attacks. Here are some ways to secure remote connections to PostgreSQL:

2. Implement IP whitelisting: Configure the PostgreSQL server to accept connections only from trusted IP addresses or networks. This can be done by configuring the firewall or by using the pg_hba.conf file in PostgreSQL.
3. Use VPN: Use a Virtual Private Network (VPN) to establish a secure connection between the client and the server. VPNs provide an additional layer of security by encrypting the entire communication channel.
4. Use SSH Tunnels: SSH tunnels can be used to encrypt and forward remote connections to a PostgreSQL server. This provides secure access to the server without exposing it to the public network.
5. Use strong authentication: Use strong passwords and implement multi-factor authentication for all user accounts. This can prevent brute force attacks and unauthorized access to the server.

By implementing these security measures, remote connections to a PostgreSQL database can be secured, ensuring the safety and privacy of sensitive data.

How to allow remote connections to PostgreSQL on Linux

Prerequisites

In this instructional guide, we will be discussing how to enable remote connections to a PostgreSQL server installed on Ubuntu. To follow along with the steps provided, you will need to have the following installed on your system:

- Ubuntu 20.04
- PostgreSQL server version 12

Please ensure that you have these requirements installed before proceeding with the guide.

Enable remote access to Postgres

To allow remote access to a PostgreSQL 12 server on Ubuntu 20.04, you need to follow the steps below:

1. Modify the PostgreSQL configuration file

Open the PostgreSQL configuration file "postgresql.conf" using your preferred text editor. The file is typically located in the /etc/postgresql/12/main directory. To open the file from the Linux Terminal, execute: `sudo nano /etc/postgresql/12/main/postgresql.conf`

Then, find the line `#listen_addresses = 'localhost'` and uncomment it (remove the # character at the beginning of the line).

Next, change the value of "listen_addresses" to "*". This allows PostgreSQL to listen on all available IP addresses. Alternatively, you can specify a specific IP address or a range of IP addresses that are allowed to connect to the server.

2. Modify the pg_hba.conf file

Take the following section:

```
 # IPv4 local connections:
host    all             all             127.0.0.1/32            md5
```

And modify it this way:

```
 # IPv4 local connections:
host    all             all             0.0.0.0/0            md5
```

3. Allow port 5432 through the firewall

To enable traffic on port 5432 through the firewall, execute the following command: `sudo ufw allow 5432/tcp`

4. Restart PostgreSQL

Run the following command to restart PostgreSQL: `sudo service postgresql restart`

After completing these steps, you should be able to connect to the PostgreSQL server from a remote machine using a PostgreSQL client. However, please note that allowing remote access to a PostgreSQL server can bear a security risk, so it is recommended to use secure passwords, encryption, and firewall rules to protect your system.

How to allow remote connections to PostgreSQL on Windows

By default, when installing PostgreSQL, you get it configured to only accept connections from the local machine. This is a security measure to prevent unauthorized access to the database server. However, in some cases, you may need to allow connections from remote hosts. Let us look at how to do this.

To allow remote connections to PostgreSQL on Windows, follow these steps:

1. Edit the PostgreSQL configuration file

1.1 Open the PostgreSQL installation directory and locate the `postgresql.conf` file. By default, the configuration file for PostgreSQL (version 14) is located at C:\Program Files\PostgreSQL\14\data\postgresql.conf.

1.2 Make a backup of the file before making any changes.

1.3 Open the `postgresql.conf` file in a text editor.

Note: You can also use a specific IP address for the PostgreSQL server to listen on.

2. Edit the pg_hba.conf file to allow remote connections

2.1 Open the PostgreSQL installation directory and locate the `pg_hba.conf` file. By default, it is located at C:\Program Files\PostgreSQL\14\data\pg_hba.conf (for PostgreSQL 14).

2.2 Make a backup of the file before making any changes.

2.3 Open the `pg_hba.conf` file in a text editor.

2.4 Add a new line at the end of the file to allow remote connections. The line should have the following format:

```
host all all 0.0.0.0/0 md5
```

This line allows connections from any IP address (0.0.0.0/0) and requires a password for authentication (md5).

2.5 Restart PostgreSQL (for example, from the Computer Management console).

3. Configure the Windows Firewall to allow incoming connections to PostgreSQL

3.1 Launch **Windows Control Pane**.

3.2 Open Windows Defender Firewall.

3.3 Click **Advanced settings** on the left-hand side of the window.

3.4 Click **Inbound Rules** on the left-hand side of the window.

3.5 Click **New Rule** on the right-hand side of the window.

3.6 Select **Port** as the type of rule and click **Next**.

3.7 Select **TCP** as the protocol and enter *5432* as the port number. Click **Next**.

3.8 Select **Allow the connection** and click **Next**.

3.9 Select the network types for which the rule should apply (*Domain*, *Private*, or *Public*). Click **Next**.

3.10 Enter a name and description for the rule and click **Finish**.

After completing these steps, remote connections to PostgreSQL should be allowed on the Windows machine.

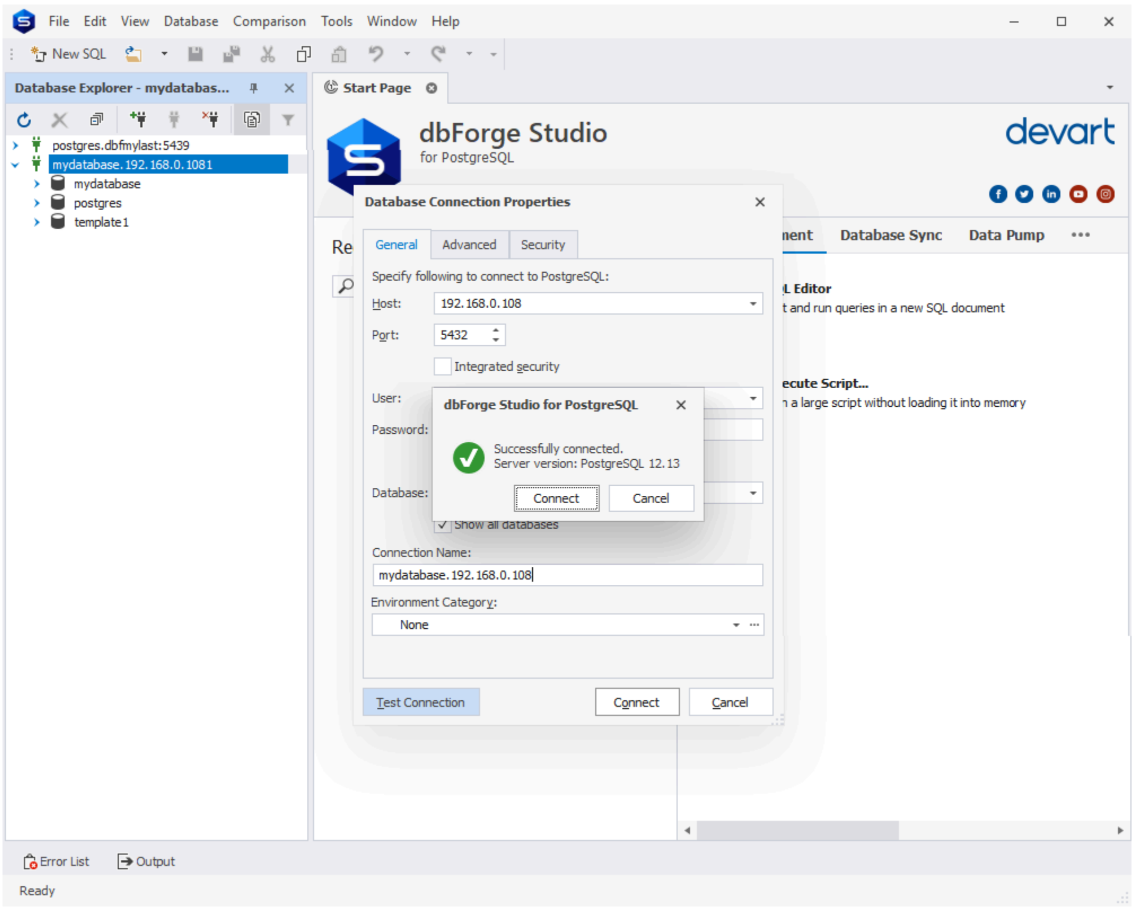
Let us look at how you can connect to the PostgreSQL server and then manage connections in one of the best PostgreSQL clients – dbForge Studio for PostgreSQL.

1. On the **Database** menu, select **New Connection**.

2. On the **Database Connection Properties** > **General** tab, specify the connection details:

- **Host:** Provide the host name
- **Port:** Provide the port number
- **User and Password:** Enter respective user credentials. By default, the password is saved automatically. If you don't want to save the password, clear the **Allow saving password** checkbox.
- **Database:** Enter the name of a PostgreSQL database you want to connect to or select it from the drop-down list.
- Optional: **Connection Name:** The connection name is generated automatically from the host name. If you want to create a distinctive name for the connection, type the new name.

Click **Test Connection** to verify the connection details you have provided. Click **Connect** to connect to a PostgreSQL server.



postgres.dbfmylast:5439
mydatabase.192.168.0.1081
mydatabase
postgres
template1

Database Connection Properties

Specify following to connect to PostgreSQL:
Host: 192.168.0.108
Port: 5432
Integrated security
User:
Password:
Database:
Show all databases
Connection Name: mydatabase.192.168.0.108
Environment Category: None

dbForge Studio for PostgreSQL
Successfully connected.
Server version: PostgreSQL 12.13

SQL Editor
...t and run queries in a new SQL document
...ecute Script...
...n a large script without loading it into memory

administration. Here are some of the advantages of using it in daily work:

User-friendly interface: dbForge Studio for PostgreSQL has a clean interface that is easy to use, even for those who are new to PostgreSQL database management.

Advanced PostgreSQL functionality: The Studio provides a wide range of advanced features, including code completion, database management, PostgreSQL data export and import, SQL editing, debugging, and profiling.

Efficient data and schema compare and sync: The IDE allows its users to easily compare PostgreSQL databases, get comprehensive information on all differences, and generate clear and accurate SQL synchronization scripts to deploy changes.

Easy database management: With dbForge Studio for PostgreSQL, you can easily create, edit, and delete tables, views, procedures, and other database objects.

Advanced SQL editing: The tool boasts a mighty PostgreSQL editor with advanced features, including code highlighting, code formatting, and auto-completion, which help to improve the efficiency and quality of your work.

Instant test data generation: dbForge Studio incorporates a powerful Data Generator allowing users to create realistic test data in just a few clicks.

Customizable code snippets: The IDE provides a library of customizable code snippets for frequently used SQL commands, which can help reduce the time and effort required to write code.

Overall, dbForge Studio for PostgreSQL is a powerful and versatile tool that provides many benefits that make it a popular choice for database developers.

To see for yourself how dbForge Studio for PostgreSQL can help you manage your databases more efficiently, we invite you to download and evaluate our free trial. Simply download free trial of dbForge Studio for PostgreSQL.

With the free trial, you'll have full access to all the features of the Studio for PostgreSQL for 30 days, so you can try out all the functionality and see how it fits into your workflow. We're confident that you'll love the product, and we look forward to hearing your feedback.