

How to Connect

- [Connecting with Node.js](#)
- [Connecting with Python](#)
- [Connecting with PHP](#)
- [Connecting with Go](#)
- [Connecting with Java](#)
- [Connecting with psql](#)
- [Connecting with pgAdmin](#)

Connecting with Node.js

This guide explains how to establish a connection between a Node.js application and a PostgreSQL database using the `pg` package. It walks through the necessary setup, configuration, and execution of a simple SQL query.

Variables

Certain parameters must be provided to establish a successful connection to a PostgreSQL database. Below is a breakdown of each required variable, its purpose, and where to find it. Here's what each variable represents:

Variable	Description	Purpose
<code>USER</code>	PostgreSQL username, from the Elestio service overview page	Identifies the database user who has permission to access the PostgreSQL database.
<code>PASSWORD</code>	PostgreSQL password, from the Elestio service overview page	The authentication key required for the specified <code>USER</code> to access the database
<code>HOST</code>	Hostname for PostgreSQL connection, from the Elestio service overview page	The address of the server hosting the PostgreSQL database.
<code>PORT</code>	Port for PostgreSQL connection, from the Elestio service overview page	The network port is used to connect to PostgreSQL. The default port is <code>5432</code> .
<code>DATABASE</code>	Database Name for PostgreSQL connection, from the Elestio service overview page	The name of the database being accessed. A PostgreSQL instance can contain multiple databases.

These values can usually be found in the Elestio service overview details as shown in the image below, make sure to take a copy of these details and add it to the code moving ahead.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host	postgresql-2p7j1-u7774.vm.elestio.app	
Port	25432	
User	postgres	
Password	*****	Show password
CLI	PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres	Show password

Prerequisites

• Install Node.js and NPM

- Check if Node.js is installed by running:

```
node -v
```

- If not installed, download it from nodejs.org and install.
- Verify npm installation:

```
npm -v
```

• Install the `pg` Package

The `pg` package enables Node.js applications to interact with PostgreSQL. Install it using:

```
npm install pg --save
```

Code

Once all prerequisites are set up, create a new file named `pg.js` and add the following code:

```
const pg = require("pg");
```

```

// Database connection configuration
const config = {
  user: "USER",
  password: "PASSWORD",
  host: "HOST",
  port: "PORT",
  database: "DATABASE",
};

// Create a new PostgreSQL client
const client = new pg.Client(config);

// Connect to the database
client.connect((err) => {
  if (err) {
    console.error("Connection failed:", err);
    return;
  }
  console.log("Connected to PostgreSQL");

  // Run a test query to check the PostgreSQL version
  client.query("SELECT VERSION()", [], (err, result) => {
    if (err) {
      console.error("Query execution failed:", err);
      client.end();
      return;
    }

    console.log("PostgreSQL Version:", result.rows[0]);

    // Close the database connection
    client.end((err) => {
      if (err) console.error("Error closing connection:", err);
    });
  });
});

```

To execute the script, open the terminal or command prompt and navigate to the directory where `pg.js`. Once in the correct directory, run the script with the command

```
node pg.js
```

If the connection is successful, the terminal will display output similar to:

```
Connected to PostgreSQL
PostgreSQL Version: {
  version: 'PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc
(Debian 12.2.0-14) 12.2.0, 64-bit'
}
```

Connecting with Python

This guide explains how to establish a connection between a **Python** application and a **PostgreSQL** database using the `psycopg2-binary` package. It walks through the necessary setup, configuration, and execution of a simple SQL query.

Variables

To connect to a PostgreSQL database, you only need one environment variable — the **connection URI**. This URI contains all the necessary information like username, password, host, port, and database name.

Variable	Description	Purpose
POSTGRES_URL	Full PostgreSQL connection string (from the Elestio service overview page)	Provides all necessary credentials and endpoint details in a single URI format.

The URI will look like this:

```
postgresql://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>
```

You can find the details needed in the URI from the **Elestio service overview** details. Copy and replace the variables carefully in the URI example provided above.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host

postgresql-2p7j1-u7774.vm.elestio.app



Port

25432



User

postgres



Password

Show password



CLI

PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres

Show password



Prerequisites

Install Python

Check if Python is installed by running:

```
python --version
```

If not installed, download it from python.org and install it.

Install `psycopg2-binary` Package

The `psycopg2-binary` package enables Python applications to interact with PostgreSQL. Install it using:

```
pip install psycopg2-binary
```

Code

Once all prerequisites are set up, create a new file named `pg.py` and add the following code and replace the `POSTGRESQL_URI` with actual link or in environment setup as you wish:

```
import psycopg2

def get_db_version():
    try:
        db_connection = psycopg2.connect('POSTGRESQL_URI')
        db_cursor = db_connection.cursor()
        db_cursor.execute('SELECT VERSION()')
        db_version = db_cursor.fetchone()[0]
        return db_version

    except Exception as e:
        print(f"Database connection error: {e}")
        return None

    finally:
        if 'db_cursor' in locals():
            db_cursor.close()
        if 'db_connection' in locals():
            db_connection.close()

def display_version():
    version = get_db_version()
    if version:
        print(f"Connected to PostgreSQL: {version}")

if __name__ == "__main__":
    display_version()
```

To execute the script, open the terminal or command prompt and navigate to the directory where `pg.py`. Once in the correct directory, run the script with the command

```
python pg.py
```

If the connection is successful, the terminal will display output similar to:

```
Connected to PostgreSQL: PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu,
compiled by gcc (Debian 12.2.0-14) 12.2.0, 64-bit
```


Connecting with PHP

This guide explains how to establish a connection between a **PHP** application and a **PostgreSQL** database using the built-in `pdo` extension. It walks through the necessary setup, configuration, and execution of a simple SQL query.

Variables

To connect to a PostgreSQL database, you only need one environment variable — the **connection URI**. This URI contains all the necessary information like username, password, host, port, and database name.

Variable	Description	Purpose
POSTGRES_URL	Full PostgreSQL connection string (from the Elestio service overview page)	Provides all necessary credentials and endpoint details in a single URI format.

The URI will look like this:

```
postgresql://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>
```

You can find the details needed in the URI from the **Elestio service overview** details. Copy and replace the variables carefully in the URI example provided above.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host

postgresql-2p7j1-u7774.vm.elestio.app



Port

25432



User

postgres



Password

Show password



CLI

PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres

Show password



Prerequisites

Install PHP

Check if PHP is installed by running:

```
php -v
```

If not installed, download and install it from <https://www.php.net/downloads.php>.

Code

Once all prerequisites are set up, create a new file named `pg.php` and add the following code and replace the `POSTGRES_URI` with actual link or in environment setup as you wish:

```
<?php

$db_url = "POSTGRES_URI";//Replace with actual URI
$db_parts = parse_url($db_url);
```

```
$dsn = "pgsql:host=${db_parts['host']};port=${db_parts['port']};dbname=Elestio";//Replace with  
your DB name  
$pdo = new PDO($dsn, $db_parts['user'], $db_parts['pass']);  
  
$version = $pdo->query("SELECT VERSION()")->fetchColumn();  
echo $version;
```

To execute the script, open the terminal or command prompt and navigate to the directory where `pg.php`. Once in the correct directory, run the script with the command

```
php pg.php
```

If the connection is successful, the terminal will display output similar to:

```
PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian  
12.2.0-14) 12.2.0, 64-bit
```

Connecting with Go

This guide explains how to establish a connection between a **Go (Golang)** application and a **PostgreSQL** database using the `github.com/lib/pq` driver. It walks through the necessary setup, configuration, and execution of a simple SQL query.

Variables

To connect to a PostgreSQL database, you only need one environment variable — the **connection URI**. This URI contains all the necessary information like username, password, host, port, and database name.

Variable	Description	Purpose
POSTGRES_URL	Full PostgreSQL connection string (from the Elestio service overview page)	Provides all necessary credentials and endpoint details in a single URI format.

The URI will look like this:

```
postgresql://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>
```

You can find the details needed in the URI from the **Elestio service overview** details. Copy and replace the variables carefully in the URI example provided above.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host

postgresql-2p7j1-u7774.vm.elestio.app



Port

25432



User

postgres



Password

Show password



CLI

PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres

Show password



Prerequisites

Install Go

Check if Go is installed by running:

```
go version
```

If not installed, download and install it from <https://go.dev/dl/>.

Install `pq` Package

Install the `pq` driver using:

```
go get github.com/lib/pq
```

Code

Once all prerequisites are set up, create a new file named `main.go` and add the following code, and replace the `POSTGRESQL_URI` with actual link or in environment setup as you wish:

```
package main

import (
    "database/sql"
    "fmt"
    "log"
    "net/url"

    _ "github.com/lib/pq"
)

func getDBConnection(connectionString string) (*sql.DB, error) {
    parsedURL, err := url.Parse(connectionString)
    if err != nil {
        return nil, fmt.Errorf("Failed to parse connection string: %v", err)
    }

    db, err := sql.Open("postgres", parsedURL.String())
    if err != nil {
        return nil, fmt.Errorf("Failed to open database connection: %v", err)
    }

    return db, nil
}

func main() {
    connectionString := "POSTGRESQL_URI"

    db, err := getDBConnection(connectionString)
    if err != nil {
        log.Fatal(err)
    }
    defer db.Close()

    query := "SELECT current_database(), current_user, version()"
    rows, err := db.Query(query)
    if err != nil {
```

```
    log.Fatal("Failed to execute query:", err)
}
defer rows.Close()

for rows.Next() {
    var dbName, user, version string
    if err := rows.Scan(&dbName, &user, &version); err != nil {
        log.Fatal("Failed to scan row:", err)
    }
    fmt.Printf("Database: %s\nUser: %s\nVersion: %s\n", dbName, user, version)
}
}
```

To execute the script, open the terminal or command prompt and navigate to the directory where `main.go`. Once in the correct directory, run the script with the command

```
go run main.go
```

If the connection is successful, the terminal will display output similar to:

```
Database: Elestio
User: postgres
Version: PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc
(Debian 12.2.0-14) 12.2.0, 64-bit
```

Connecting with Java

This guide explains how to establish a connection between a **Java** application and a **PostgreSQL** database using the **JDBC driver**. It walks through the necessary setup, configuration, and execution of a simple SQL query.

Variables

Certain parameters must be provided to establish a successful connection to a PostgreSQL database. Below is a breakdown of each required variable, its purpose, and where to find it. Here's what each variable represents:

Variable	Description	Purpose
<code>USER</code>	PostgreSQL username, from the Elestio service overview page	Identifies the database user who has permission to access the PostgreSQL database.
<code>PASSWORD</code>	PostgreSQL password, from the Elestio service overview page	The authentication key required for the specified <code>USER</code> to access the database
<code>HOST</code>	Hostname for PostgreSQL connection, from the Elestio service overview page	The address of the server hosting the PostgreSQL database.
<code>PORT</code>	Port for PostgreSQL connection, from the Elestio service overview page	The network port is used to connect to PostgreSQL. The default port is <code>5432</code> .
<code>DATABASE</code>	Database Name for PostgreSQL connection, from the Elestio service overview page	The name of the database being accessed. A PostgreSQL instance can contain multiple databases.

These values can usually be found in the Elestio service overview details, as shown in the image below. Make sure to take a copy of these details and add them to the code moving ahead.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host

postgresql-2p7j1-u7774.vm.elestio.app



Port

25432



User

postgres



Password

Show password



CLI

PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres

Show password



Prerequisites

Install Java & JDBC driver

Check if Java is installed by running:

```
java -version
```

If not installed, install it first and then download and install **JDBC** driver from

<https://jdbc.postgresql.org/download/> or if you have Maven installed, run the following command with updated version of the driver:

```
mvn org.apache.maven.plugins:maven-dependency-plugin:2.8:get -Dartifact=org.postgresql:postgresql:42.7.5:jar -Ddest=postgresql-42.7.5.jar
```

Code

Once all prerequisites are set up, create a new file named `Pg.java` and add the following code:

```

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.HashMap;
import java.util.Map;

public class Pg {
    private static class ConnectionConfig {
        private final String host;
        private final String port;
        private final String database;
        private final String username;
        private final String password;

        public ConnectionConfig(String host, String port, String database, String username,
String password) {
            this.host = host;
            this.port = port;
            this.database = database;
            this.username = username;
            this.password = password;
        }

        public String getConnectionUrl() {
            return String.format("jdbc:postgresql://%s:%s/%s?sslmode=require", host, port,
database);
        }

        public boolean isValid() {
            return host != null && !host.isEmpty() &&
                port != null && !port.isEmpty() &&
                database != null && !database.isEmpty();
        }
    }

    private static Map<String, String> parseArguments(String[] args) {
        Map<String, String> config = new HashMap<>();
        for (int i = 0; i < args.length - 1; i++) {

```

```

        String key = args[i].toLowerCase();
        String value = args[++i];
        config.put(key, value);
    }
    return config;
}

private static ConnectionConfig createConfig(Map<String, String> args) {
    return new ConnectionConfig(
        args.get("-host"),
        args.get("-port"),
        args.get("-database"),
        args.get("-username"),
        args.get("-password")
    );
}

private static void validateConnection(Connection connection) throws SQLException {
    try (Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT version()")) {
        if (rs.next()) {
            System.out.println("Database Version: " + rs.getString("version"));
        }
    }
}

public static void main(String[] args) {
    try {
        // Load PostgreSQL driver
        Class.forName("org.postgresql.Driver");

        // Parse and validate configuration
        Map<String, String> parsedArgs = parseArguments(args);
        ConnectionConfig config = createConfig(parsedArgs);

        if (!config.isValid()) {
            System.err.println("Error: Missing required connection parameters (host, port,
database)");
            return;
        }
    }
}

```

```

// Establish connection and validate
try (Connection conn = DriverManager.getConnection(
    config.getConnectionUrl(),
    config.username,
    config.password)) {

    System.out.println("Successfully connected to the database!");
    validateConnection(conn);
}

} catch (ClassNotFoundException e) {
    System.err.println("Error: PostgreSQL JDBC Driver not found");
    e.printStackTrace();
} catch (SQLException e) {
    System.err.println("Database connection error:");
    e.printStackTrace();
}
}
}

```

To execute the script, open the terminal or command prompt and navigate to the directory where `Pg.java`. Once in the correct directory, run the script with the command (Update the variables with actual values acquired from previous steps.)

```

javac Pg.java && java -cp postgresql-42.7.5.jar:. Pg -host HOST -port PORT -database DATABASE
-username avnadmin -password PASSWORD

```

If the connection is successful, the terminal will display output similar to:

```

Successfully connected to the database!
Database Version: PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled
by gcc (Debian 12.2.0-14) 12.2.0, 64-bit

```

Connecting with psql

This guide explains how to connect to a **PostgreSQL** database using the `psql` command-line tool. It walks through the necessary setup, connection process, and execution of a simple SQL query.

Variables

To connect to a PostgreSQL database, you only need one environment variable — the **connection URI**. This URI contains all the necessary information like username, password, host, port, and database name.

Variable	Description	Purpose
POSTGRESQL_URI	Full PostgreSQL connection string (from the Elestio service overview page)	Provides all necessary credentials and endpoint details in a single URI format.

The URI will look like this:

```
postgresql://<USER>:<PASSWORD>@<HOST>:<PORT>/<DATABASE>
```

You can find the details needed in the URI from the **Elestio service overview** details. Copy and replace the variables carefully in the URI example provided above.



postgresql-2p7j1

PostgreSQL

Running

Open terminal

Delete service

Clone this service

Overview

Tools

Backups

Metrics

Monitoring

Logs

Audit

Security

Alerts

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Database Admin

Display your database credentials

Hide DB Credentials

Host postgresql-2p7j1-u7774.vm.elestio.app

Port 25432

User postgres

Password *****

Show password

CLI

PGPASSWORD=***** psql --host=postgresql-2p7j1-u7774.vm.elestio.app --port=25432 --username=postgres

Show password

Prerequisites

While following this tutorial, you will need to have `psql` already installed; if not head over to <https://www.postgresql.org/download/> and download it first.

Connecting to PostgreSQL

Open your terminal and run the following command to connect to your PostgreSQL database using the full connection URI:

```
psql POSTGRES_URI
```

If the connection is successful, you'll see output similar to this. Here it will show you the database you tried to connect to, which in this case is Elestio:

```
psql (17.4, server 16.8 (Debian 16.8-1.pgdg120+1))
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, compression: off, ALPN:
none)
```

Type "help" for help.

Elestio=#

To ensure you're connected correctly, run this command inside the `psql` prompt:

```
SELECT version();
```

You should receive output like the following:

```
version
```

```
-----  
-----
```

```
 PostgreSQL 16.8 (Debian 16.8-1.pgdg120+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian  
12.2.0-14) 12.2.0, 64-bit
```

```
(1 row)
```

Connecting with pgAdmin

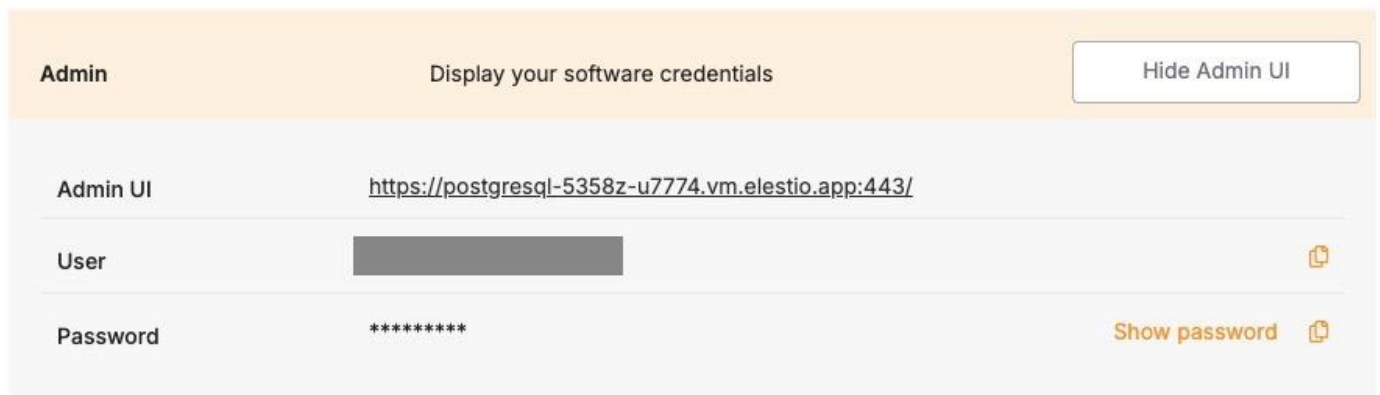
pgAdmin is a widely used graphical interface for PostgreSQL that allows you to manage, connect to, and run queries on your databases with ease.

Variables

To connect using `pgAdmin`, you'll need the following connection parameters. When you deploy a PostgreSQL service on Elestio, you also get a pgAdmin dashboard configured for you to use with these variables. These details are available in the **Elestio service overview page**:

Variable	Description	Purpose
USER	pgAdmin username	Identifies the pgAdmin user with access permission.
PASSWORD	pgAdmin password	Authentication key for the <code>USER</code> .

You can find these values in your Elestio project dashboard under **Admin** section.



Prerequisites

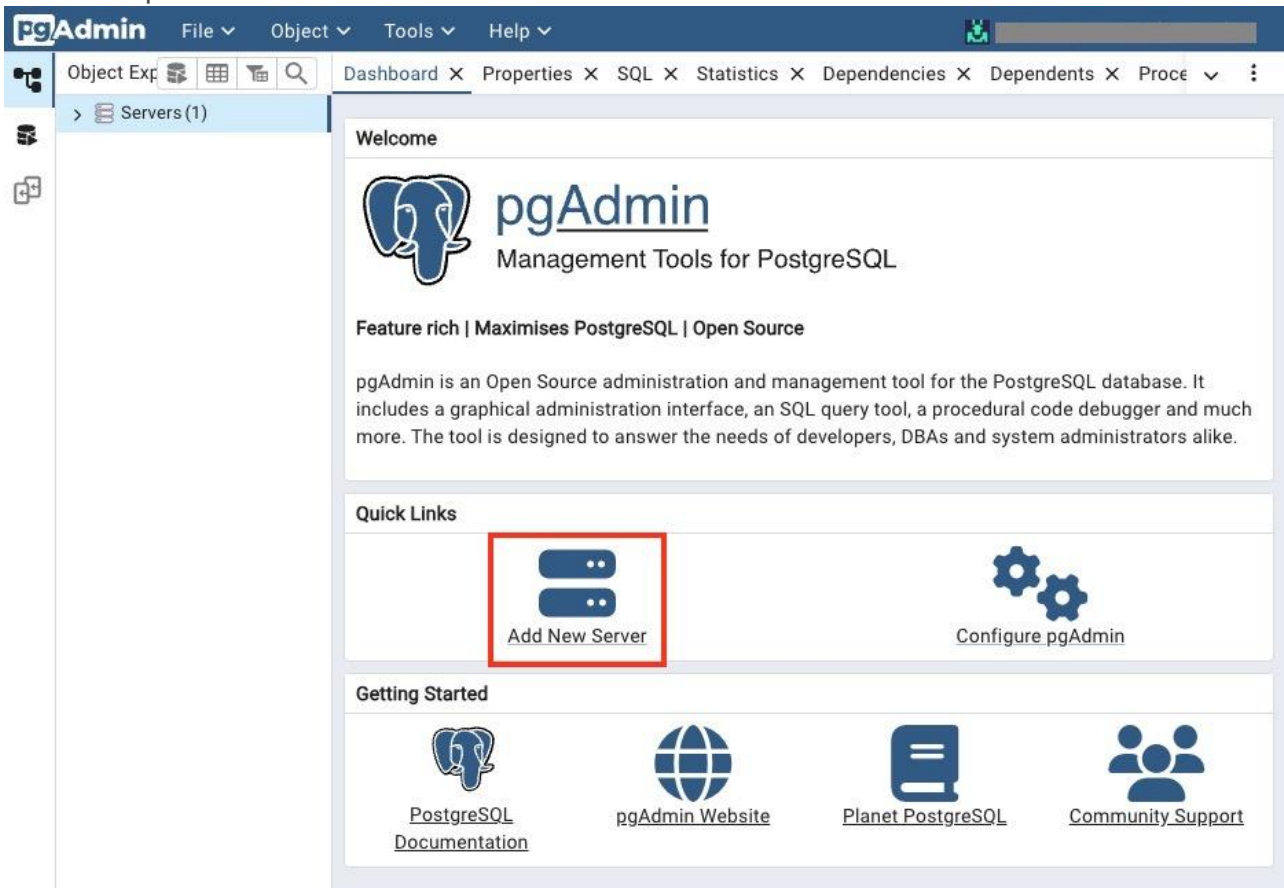
Make sure the **PostgreSQL** service is correctly deployed on Elestio and you are able to access the Admin section like the one in the image above.

Setting Up the Connection

1. Launch **pgAdmin** from the Admin UI URL and log in with the credentials acquired in the steps before.



2. Click on "**Create**" and select "**Server...**" from the dropdown, or find **Add New Server** from the quick links



3. In the **General** tab:
 - Enter a name for your connection (e.g., `Trial pgAdmin Connection`).

Register - Server ×

General | Connection | Parameters | SSH Tunnel | Advanced | Tags

Name: Trial pgAdmin Connection

Server group: Servers

Background:

Foreground:

Connect now?:

Shared?:

Shared Username:

Comments:

4. Go to the **Connection** tab and enter the following details:

- **Host name/address:**
- **Port:**
- **Maintenance database:**
- **Username:**
- **Password:**

Host name/address 

Port

5432

Maintenance database

postgres

Username

trial-user

Kerberos authentication?

Password

Save password?

Role

Service



✕ Close

↺ Reset

💾 Save