

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 |
|----------------|--|---|
| 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

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

```

■

Revision #2

Created 31 March 2025 09:10:22 by kaiwalya

Updated 31 March 2025 09:33:27 by kaiwalya