

# Connecting with Go

This guide explains how to establish a connection between a Go application and a Redis database using the go-redis package. It walks through the necessary setup, configuration, and execution of a simple Redis command.

## Variables

Certain parameters must be provided to establish a successful connection to a Redis 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
HOST	Redis hostname, from the Elestio service overview page	The address of the server hosting your Redis instance.
PORT	Port for Redis connection, from the Elestio service overview page	The network port used to connect to Redis. The default port is 6379.
PASSWORD	Redis password, from the Elestio service overview page	The authentication key required to connect securely to Redis.

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.



redis-aiont

Redis

Cluster

Running

Open terminal

Delete cluster

Add node

Overview

Nodes

Backups

Audit

Termination protection

Disabled. VM can be powered off and terminated.

Protection deactivated



Auto-Failover

Enabled. In case of failure, the cluster will automatically attempt to recover

Auto-Failover activated



Nodes

2 Nodes: 1 Primary, 1 Replica

Add node

Database Admin

Display your database credentials

Hide DB Credentials

Host

redis-aiont-u7774.vm.elestialio.app



Port

26379



User

default



Password

\*\*\*\*\*

Show password



CLI

redis-cli -h redis-aiont-u7774.vm.elestialio.app -p 26379 --user default --pass '\*\*\*\*\*'

Show password



# Prerequisites

## Install Go

Check if Go is installed by running:

```
go version
```

If not installed, download it from [golang.org](https://golang.org) and install.

## Install the go-redis Package

The go-redis package enables Go applications to interact with Redis. Install it using:

```
go get github.com/redis/go-redis/v9
```

# Code

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

```
package main

import (
    "context"
    "fmt"
    "time"

    "github.com/redis/go-redis/v9"
)

func main() {
    opt := &redis.Options{
        Addr:      "HOST:PORT",
        Password:  "PASSWORD",
        DB:        0,
    }

    rdb := redis.NewClient(opt)
    ctx, cancel := context.WithTimeout(context.Background(), 5*time.Second)
    defer cancel()

    err := rdb.Set(ctx, "testKey", "Hello Redis", 0).Err()
    if err != nil {
        fmt.Println("Redis operation failed:", err)
        return
    }

    val, err := rdb.Get(ctx, "testKey").Result()
    if err != nil {
        fmt.Println("Redis operation failed:", err)
        return
    }

    fmt.Println("Connected to Redis")
    fmt.Println("Retrieved value:", val)

    if err := rdb.Close(); err != nil {
        fmt.Println("Error closing connection:", err)
    }
}
```

```
}  
}
```

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

```
go run redis.go
```

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

```
Connected to Redis  
Retrieved value: Hello Redis
```

---

Revision #2

Created 19 May 2025 13:17:24 by kaiwalya

Updated 19 May 2025 13:44:34 by kaiwalya