Go Driver Examples

The Go MongoDB driver isn't an officially supported driver at the moment, and as such is maintained by the community. It's called mgo.

mgo at the time of this writing, *07-10-15*, supports the following versions of MongoDB:

Go Driver Version	MongoDB 2.4	MongoDB 2.6	MongoDB 3.0
v2	✓	✓	✓

Here are the current versions of *golang* the driver supports:

Go Driver Version	go1.1	go1.2	go1.3	go1.4	go1.5
v2	✓	✓	✓	✓	*untested*

Installation

Installing mgo is simple, using the usual go get procedure:

```
$ go get gopkg.in/mgo.v2
```

Example document

Here’s the example document we’ll be using:

```
{
    "start": "2015-09-02T22:46:30.782Z",
    "end": "2016-09-02T22:46:30.782Z",
    "location": "Texas",
    "official_game": false,
    "winner": "Javi",
    "players": [
        {
            "name": "Javi",
            "decks": [
                "Dinosaurs",
                "Plants"
            ],
            "points": 24,
            "place": 1
        },
        {
            "name": "Seth",
            "decks": [
                "Spies",
                "Zombies"
            ],
            "points": 20,
            "place": 2
        },
        {
            "name": "Dave",
            "decks": [
                "Steampunk",
                "Wizard"
            ],
            "points": 20,
            "place": 2
        },
        {
            "name": "Castro",
            "decks": [
                "Shapeshifters",
                "Ninjas"
            ],
            "points": 18,
            "place": 4
        }
    ]
}
```

Connecting

Warning

When connecting using the MongoDB URI, we highly recommend avoiding usernames or passwords with an @ symbol inside. This can break the URI parsing and cause failures when trying to connect.

Connecting to a replica set:

```go
package main

import (
    "fmt"
    "gopkg.in/mgo.v2"
)

func main() {
    Host := []string{
        "dfw-c9-0.objectrocket.com:12345",
        "dfw-c9-1.objectrocket.com:12345",
    }
    const (
        Username = "YOUR_USERNAME"
        Password = "YOUR_PASSWORD"
        Database = "YOUR_DATABASE_NAME"
        ReplicaSetName = "c74b5276378ed3bd70cba37a3ac45fea"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    Host,
        Username: Username,
        Password: Password,
        Database: Database,
        ReplicaSetName: ReplicaSetName,
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    fmt.Printf("Connected to replica set %v!\n", session.LiveServers())
}
```

Connecting to a sharded instance:

```go
package main

import (
    "fmt"
    "gopkg.in/mgo.v2"
)

func main() {
    const (
        Host     = "iad-mongos0.objectrocket.com:12345"
        Username = "YOUR_USERNAME"
        Password = "YOUR_PASSWORD"
        Database = "YOUR_DATABASE_NAME"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    fmt.Printf("Connected to %v!\n", session.LiveServers())
}
```

Connecting to a sharded instance using SSL:

```go
package main

import (
    "crypto/tls"
    "fmt"
    "gopkg.in/mgo.v2"
    "net"
)

func main() {
    const (
        Host     = "iad-mongos0.objectrocket.com:12345"
        Username = "YOUR_USERNAME"
        Password = "YOUR_PASSWORD"
        Database = "YOUR_DATABASE_NAME"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
        DialServer: func(addr *mgo.ServerAddr) (net.Conn, error) {
            return tls.Dial("tcp", addr.String(), &tls.Config{})
        },
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    fmt.Printf("Connected to %v!\n", session.LiveServers())
}
```

Warning

The below examples are connecting via SSL, which doesn't work with our Replica Set instances. Please adjust accordingly.

Creating a document

Creating and inserting a document:

```go
package main

import (
    "crypto/tls"
    "fmt"
    "gopkg.in/mgo.v2"
    "net"
    "time"
)

type Game struct {
    Winner       string    `bson:"winner"`
    OfficialGame bool      `bson:"official_game"`
    Location     string    `bson:"location"`
    StartTime    time.Time `bson:"start"`
    EndTime      time.Time `bson:"end"`
    Players      []Player  `bson:"players"`
}

type Player struct {
    Name   string    `bson:"name"`
    Decks  [2]string `bson:"decks"`
    Points uint8     `bson:"points"`
    Place  uint8     `bson:"place"`
}

func NewPlayer(name, firstDeck, secondDeck string, points, place uint8) Player {
    return Player{
        Name:   name,
        Decks:  [2]string{firstDeck, secondDeck},
        Points: points,
        Place:  place,
    }
}

func main() {
    const (
        Host       = "iad-mongos0.objectrocket.com:12345"
        Username   = "YOUR_USERNAME"
        Password   = "YOUR_PASSWORD"
        Database   = "YOUR_DATABASE_NAME"
        Collection = "YOUR_COLLECTION_NAME"
    )

    game := Game{
        Winner:       "Dave",
        OfficialGame: true,
        Location:     "Austin",
        StartTime:    time.Date(2015, time.February, 12, 04, 11, 0, 0, time.UTC),
        EndTime:      time.Date(2015, time.February, 12, 05, 54, 0, 0, time.UTC),
        Players: []Player{
            NewPlayer("Dave", "Wizards", "Steampunk", 21, 1),
            NewPlayer("Javier", "Zombies", "Ghosts", 18, 2),
            NewPlayer("George", "Aliens", "Dinosaurs", 17, 3),
            NewPlayer("Seth", "Spies", "Leprechauns", 10, 4),
        },
    }

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
        DialServer: func(addr *mgo.ServerAddr) (net.Conn, error) {
            return tls.Dial("tcp", addr.String(), &tls.Config{})
        },
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    fmt.Printf("Connected to %v\n", session.LiveServers())

    coll := session.DB(Database).C(Collection)
    if err := coll.Insert(game); err != nil {
        panic(err)
    }
    fmt.Println("Document inserted successfully!")
}
```

Reading documents

Finding all documents with a specific field:

```go
package main

import (
    "crypto/tls"
    "fmt"
    "gopkg.in/mgo.v2"
    "gopkg.in/mgo.v2/bson"
    "net"
)

func main() {
    const (
        Host       = "iad-mongos0.objectrocket.com:12345"
        Username   = "YOUR_USERNAME"
        Password   = "YOUR_PASSWORD"
        Database   = "YOUR_DATABASE_NAME"
        Collection = "YOUR_COLLECTION_NAME"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
        DialServer: func(addr *mgo.ServerAddr) (net.Conn, error) {
            return tls.Dial("tcp", addr.String(), &tls.Config{})
        },
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    coll := session.DB(Database).C(Collection)

    // Find the number of games won by Dave
    player := "Dave"
    gamesWon, err := coll.Find(bson.M{"winner": player}).Count()
    if err != nil {
        panic(err)
    }

    fmt.Printf("%s has won %d games.\n", player, gamesWon)
}
```

Updating a document

Updating a document:

```go
package main

import (
    "crypto/tls"
    "fmt"
    "gopkg.in/mgo.v2"
    "gopkg.in/mgo.v2/bson"
    "net"
)

func main() {
    const (
        Host       = "iad-mongos0.objectrocket.com:12345"
        Username   = "YOUR_USERNAME"
        Password   = "YOUR_PASSWORD"
        Database   = "YOUR_DATABASE_NAME"
        Collection = "YOUR_COLLECTION_NAME"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
        DialServer: func(addr *mgo.ServerAddr) (net.Conn, error) {
            return tls.Dial("tcp", addr.String(), &tls.Config{})
        },
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    coll := session.DB(Database).C(Collection)

    // Change the winner for game 55da80 to Seth
    gameId := bson.ObjectIdHex("55da804ea5b2a779329ceb8e")
    newWinner := "Seth"
    update := bson.M{"$set": bson.M{"winner": newWinner}}
    if err := coll.UpdateId(gameId, update); err != nil {
        panic(err)
    }

    fmt.Printf("Winner of game %s updated to %s.\n", gameId, newWinner)
}
```

Deleting a document

Deleting a specific document:

```go
package main

import (
    "crypto/tls"
    "fmt"
    "gopkg.in/mgo.v2"
    "gopkg.in/mgo.v2/bson"
    "net"
)

func main() {
    const (
        Host       = "iad-mongos0.objectrocket.com:12345"
        Username   = "YOUR_USERNAME"
        Password   = "YOUR_PASSWORD"
        Database   = "YOUR_DATABASE_NAME"
        Collection = "YOUR_COLLECTION_NAME"
    )

    session, err := mgo.DialWithInfo(&mgo.DialInfo{
        Addrs:    []string{Host},
        Username: Username,
        Password: Password,
        Database: Database,
        DialServer: func(addr *mgo.ServerAddr) (net.Conn, error) {
            return tls.Dial("tcp", addr.String(), &tls.Config{})
        },
    })
    if err != nil {
        panic(err)
    }
    defer session.Close()

    coll := session.DB(Database).C(Collection)

    // Remove all unofficial games
    info, err := coll.RemoveAll(bson.M{"official_game": false})
    if err != nil {
        panic(err)
    }

    fmt.Printf("%d unofficial game(s) removed!\n", info.Removed)
}
```

Additional reading

If you need more help with *mgo*, here are some links to more documentation:

- mgo GoDoc documentation
- mgo Mailing List
- mgo Github

As always, if you have any questions, please don’t hesitate to reach out to our support team!