HOW IT'S CHANGING

GO AND MONGO

I AM...

- Dj Walker-Morgan
- Currently:
  - Technical Content Curator at Compose* (part of IBM)
    - Compose does databases
- @codepope on Twitter
- codepope@gmail.com

FIRST THERE WAS...

MGO

MGO – THE ORIGINAL GO DRIVER

- Originally the only game in town
- Very Go idiomatic
- Adopted by MongoDB for their tooling
- Lets look at connecting, reading and writing

CONNECTING WITH A SELF-SIGNED TLS CERTIFICATE WITH MGO - 1

```
connectionString, present := os.LookupEnv("COMPOSE_MONGODB_URL")

if !present {
        log.Fatal("Need to set COMPOSE_MONGODB_URL environment variable")
}

roots := x509.NewCertPool()

if ca, err := ioutil.ReadFile(os.Getenv("PATH_TO_MONGODB_CERT")); err == nil {
        roots.AppendCertsFromPEM(ca)
}

tlsConfig := &tls.Config{}
tlsConfig.RootCAs = roots

trimmedConnectionString := strings.TrimSuffix(connectionString, "?ssl=true")
dialInfo, err := mgo.ParseURL(trimmedConnectionString)

if err != nil {
        log.Fatal(err)
}

...
```

CONNECTING WITH A SELF-SIGNED TLS CERTIFICATE WITH MGO - 2

```
...

dialInfo.DialServer = func(addr *mgo.ServerAddr) (net.Conn, error) {
        return tls.Dial("tcp", addr.String(), tlsConfig)
}

session, err = mgo.DialWithInfo(dialInfo)

if err != nil {
        log.Fatal(err)
}

defer session.Close()
```

READING WITH MGO

```
// Reading

type item struct {
        ID bson.ObjectId `bson:"_id,omitempty"`
        Word string `bson:"word"`
        Definition string `bson:"definition"`
}

c := session.DB("grand_tour").C("words")

var items []item

err := c.Find(nil).Sort("word").All(&items)

if err != nil {
        return err
}
```

WRITING WITH MGO

```
// Writing

type item struct {
        ID bson.ObjectId `bson:"_id,omitempty"`
        Word string `bson:"word"`
        Definition string `bson:"definition"`
}

c := session.DB("grand_tour").C("words")

newItem := item{Word: word), Definition: definition)}

err := c.Insert(newItem)

if err != nil {
        return err
}
```

THE MGO PROBLEM

- Author stopped using MongoDB
  - Last update was August 2016
  - Huge thanks to Gustavo Niemeyer
- original mgo works but bit rot is inevitable
  - MongoDB has a major update each year now
- Forks appeared to fill the gap
  - Many applied various PRs which had built up over time.

THE MGO FORKS

- Github says there's 554 forks but...
- The `globalsign/mgo` fork is the *blessed* community fork
- Changes include
  - MongoDB 3.2, 3.4 and 3.6 feature support
  - Easier SSL/TLS
  - See https://github.com/globalsign/mgo#changes
- Use this if you want to use mgo

ENTER

MONGODB

THE MONGODB GO DRIVER

- Announced January 2018
- Built to conform with other MongoDB drivers
  - Tracks the ongoing development of the driver spec
- Less Go idiomatic
- More MongoDB current
- So how's it going?

THE MONGODB DRIVER – ALPHA 10

- Functional but still alpha
- Now implementing OP_MSG
  - OP_MSG simplifies the MongoDB protocol
- Lots of work happening on BSON codec redesign
- Plenty of outstanding issues

CONNECTING WITH A SELF-SIGNED TLS CERTIFICATE WITH MONGODB-GO

```
var client *mongo.Client

connectionString, present := os.LookupEnv("COMPOSE_MONGODB_URL")

if !present {
        log.Fatal("Need to set COMPOSE_MONGODB_URL environment variable")
}

certpath, certavail := os.LookupEnv("PATH_TO_MONGODB_CERT")

var err error

if certavail {
        client, err = mongo.NewClientWithOptions(connectionString, mongo.ClientOpt.SSLCaFile(certpath))
} else {
        client, err = mongo.NewClient(connectionString)
}

if err != nil {
        log.Fatal(err)
}

err = client.Connect(nil)

if err != nil {
        log.Fatal(err)
}

defer client.Disconnect(nil)
```

READING WITH MONGODB-GO

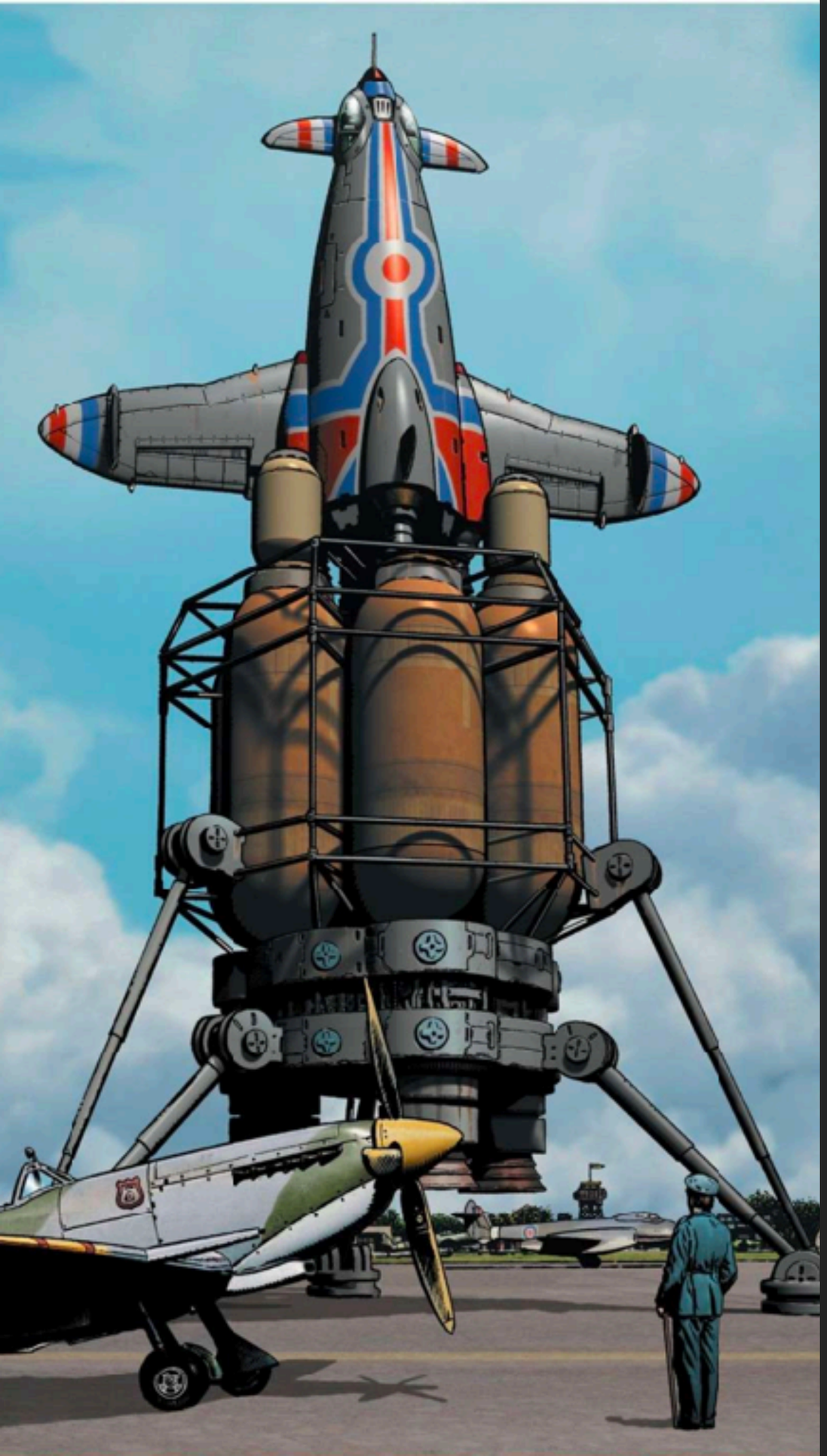
```
// Reading

type item struct {
        ID bson.ObjectId `bson:"_id,omitempty"`
        Word string `bson:"word"`
        Definition string `bson:"definition"`
}

c := client.Database("grand_tour").Collection("words")

sort, err := mongo.Opt.Sort(bson.NewDocument(bson.EC.Int32("word", 1)))

if err != nil {
        log.Fatal("Sort error ", err)
}

cur, err := c.Find(nil, nil, sort)

if err != nil {
        log.Fatal("Cursor error",err)
}

defer cur.Close(context.Background())

var items []item

for cur.Next(nil) {
        item := item{}
        err := cur.Decode(&item)
        if err != nil {
                log.Fatal("Decode error ", err)
        }
        items = append(items, item)
}

if err := cur.Err(); err != nil {
        log.Fatal("Cursor error ", err)
}
```

WRITING WITH MONGODB-GO

```
// Writing

c := client.Database("grand_tour").Collection("words")

newItemDoc := bson.NewDocument(bson.EC.String("word", word), bson.EC.String("definition", definition))

_, err := c.InsertOne(nil, newItemDoc)

if err != nil {
       log.Fatal(err)
}
```

THE MONGODB DRIVER STATE OF PLAY

- Connections easier
- Reading is now cursor centric by default
  - But then unlike examples, real code will tend to use cursors
- Writing is simple though compiling your own BSON documents can be tedious



WHAT NOW FOR

GO AND MONGO

WHICH TO PICK?

- If a long time scale...
  - Consider tracking MongoDB driver
- Otherwise...
  - Go with the globalsign/mgo fork
  - But...
    - Abstract your data access so you can move in future

RESOURCES

- mgo
  - https://github.com/globalsign/mgo#changes
- mongodb
  - https://github.com/mongodb/mongo-go-driver
- Compose examples
  - https://github.com/compose-grandtour/golang