Webinar: Getting Started with MongoDB and Ruby

Brandon Black

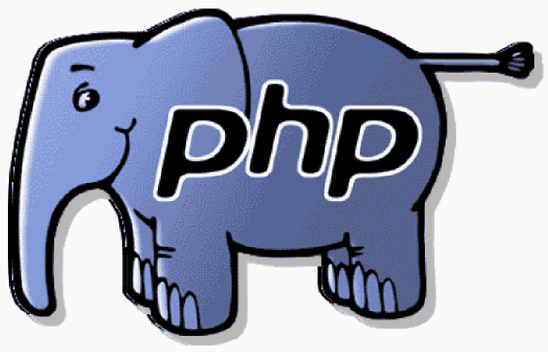
Software Engineer, MongoDB
@brandonmblack

mongoDB


C++


Microsoft®
.NET


ERLANG


php

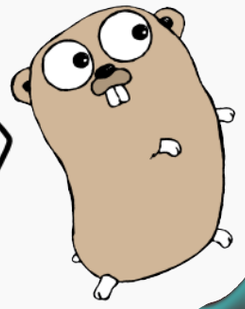

Scala


C

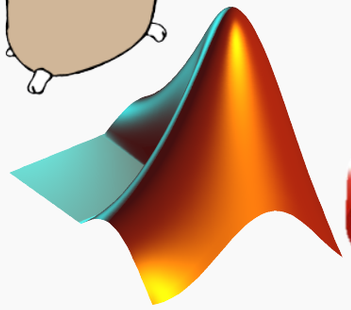

nodeJS



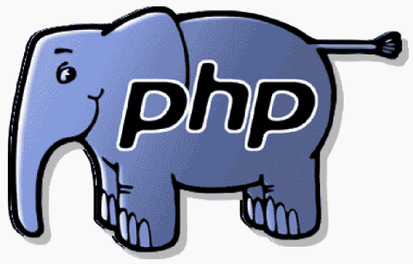












mongoDB

+

“I believe people want to express themselves when they program. They don't want to fight with the language. Programming languages must feel natural to programmers. I tried to make people enjoy programming and concentrate on the fun and creative part of programming when they use Ruby.”

— Yukihiro “Matz” Matsumoto, Creator of Ruby (2001)

What is MongoDB?

mongoDB

MongoDB is a _____ database

- Document
- Open source
- High performance
- Horizontally scalable
- Full featured

Document Database

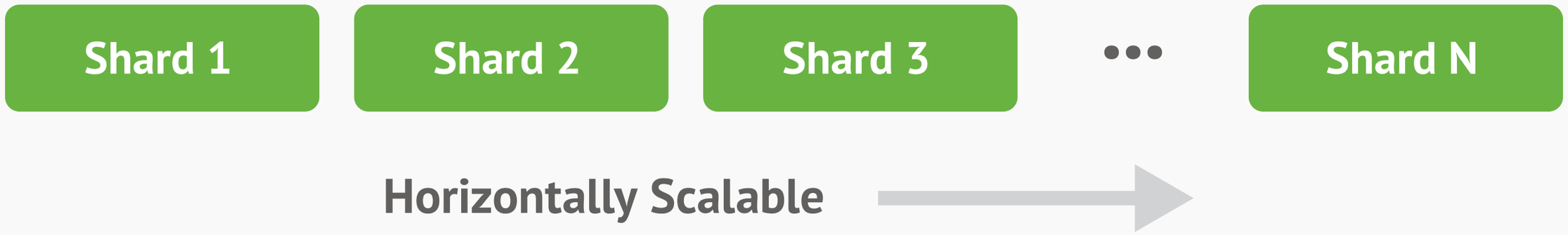
- Not for .PDF & .DOC files
- A document is essentially an associative array
- Document = JSON object
- Document = PHP Array
- Document = Python Dict
- Document = Ruby Hash

Open-Source

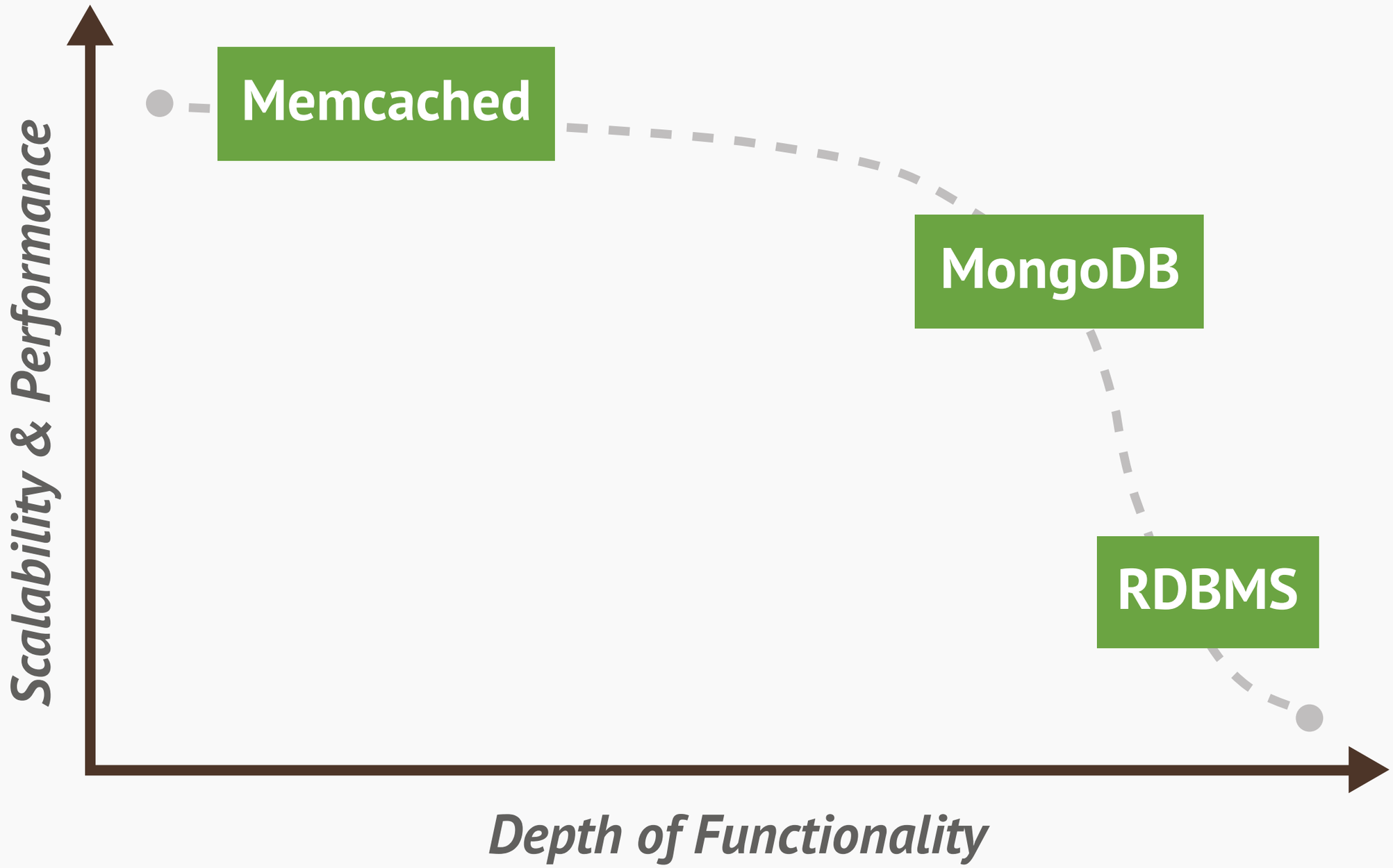
- MongoDB is an open source project
- Available on GitHub
- Licensed under the AGPL
- Started & sponsored by MongoDB, Inc. (10gen)
- Commercial licenses available
- Contributions welcome

High Performance

- Written in C++
- Extensive use of memory-mapped files
  i.e. read-through write-through memory caching.
- Runs nearly everywhere
- Data serialized as BSON (fast parsing)
- Full support for primary & secondary indexes
- Document model = less work


Shard 1
Shard 2
Shard 3
Shard N
Horizontally Scalable

Data Landscape



Full Featured

- Ad Hoc queries
- Real time aggregation
- Rich query capabilities
- Strongly consistent
- Geospatial features
- Native support for most programming languages
- Flexible schema (not schema-less!)

Installation & Setup

1 Download

From website:

http://www.mongodb.org/downloads

Package manager:

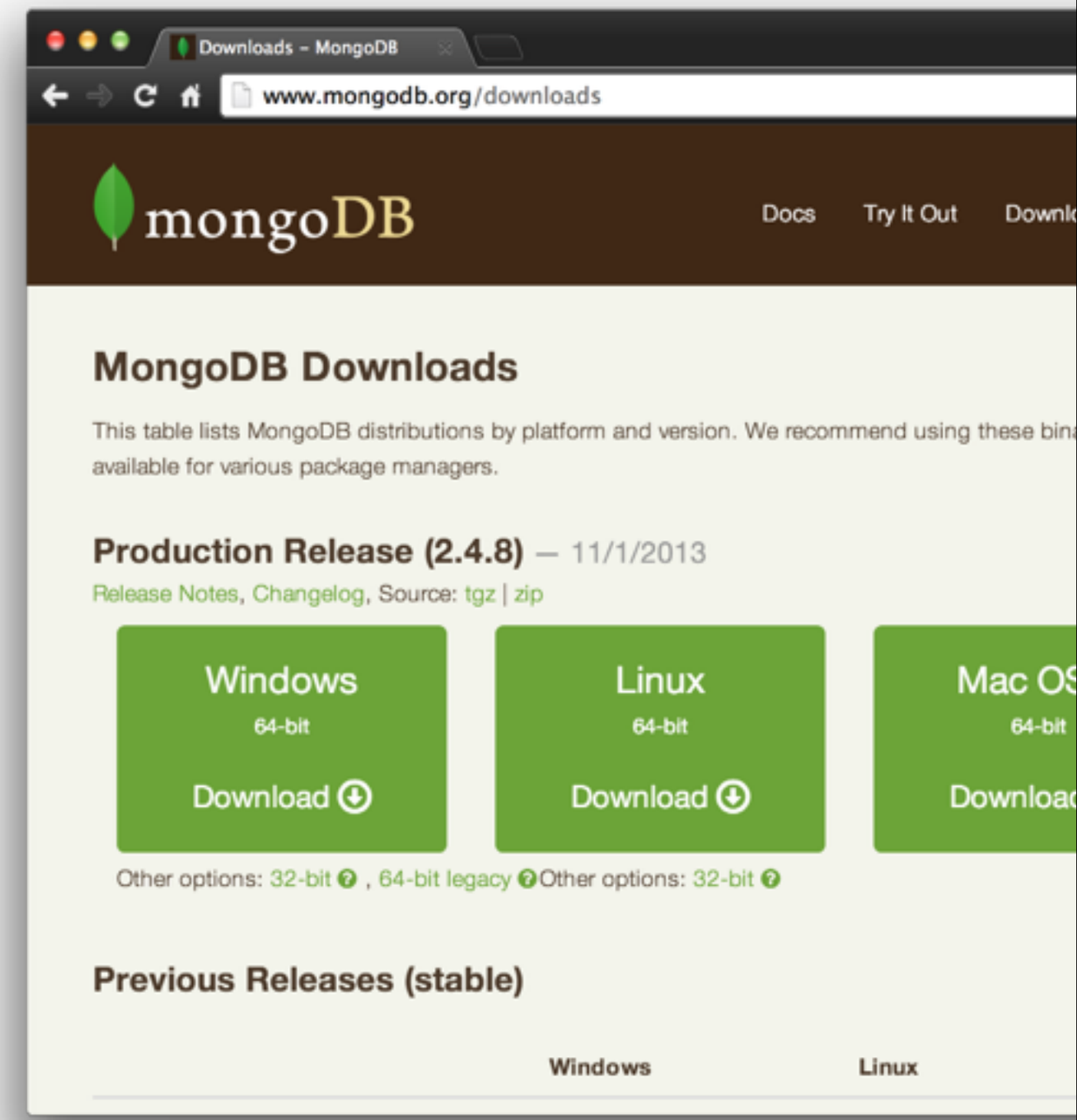
```
sudo apt-get install mongodb-10gen
brew install mongodb
```

2 Startup MongoDB

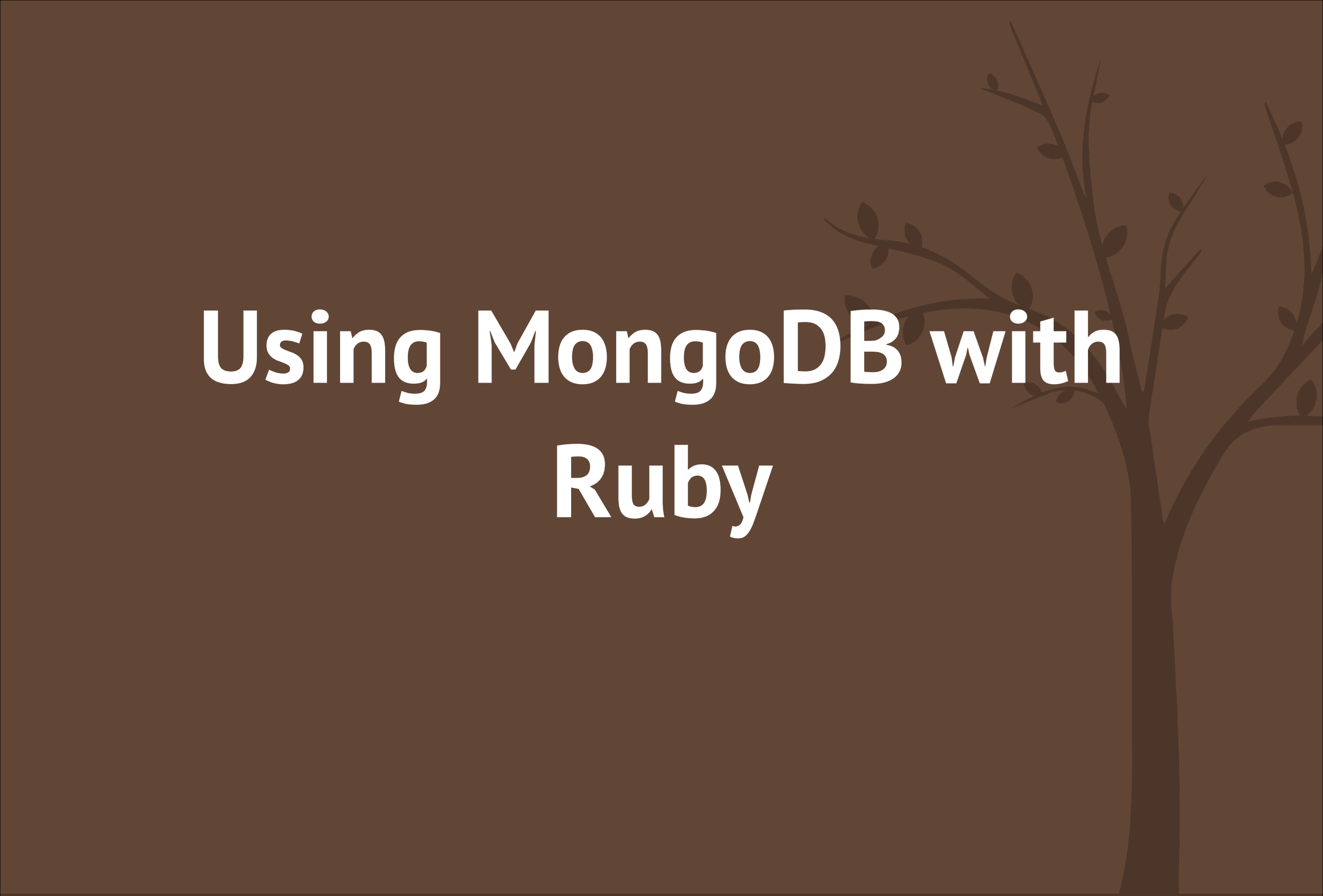
```
mongod –dbpath /path/to/data
```

3 Connect with the `mongo` Shell

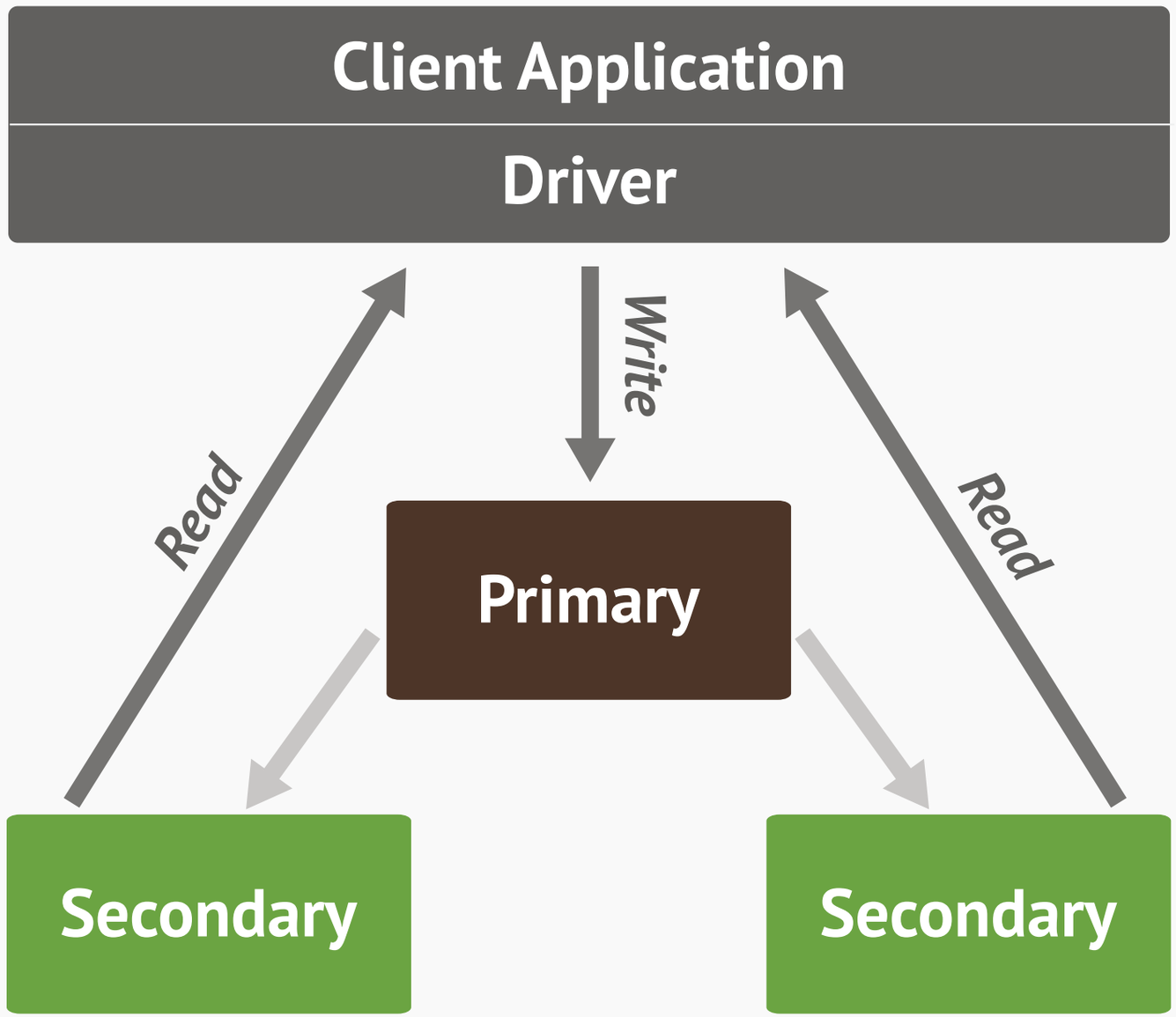
Or in our case... IRB/Pry!



Using MongoDB with Ruby



Using the Ruby Driver



Using the Ruby Driver

```
# install gem and native extensions (optional)
gem install mongo
gem install bson_ext
```

```
require 'mongo'
include Mongo

# connecting to the database
client = MongoClient.new # defaults to localhost:27017
client = MongoClient.new('host1.example.com')
client = MongoClient.new('host1.example.com', 27017)


# using configuration options
opts   = { :pool_size => 5, :pool_timeout => 5 }
client = MongoClient.new('host1.example.com', 27017, opts)
```

Using the Ruby Driver

```
seeds = ['h1.example.com:27017',
         'h2.example.com:27017',
         'h3.example.com:27017']

# connecting to a replica set
client = MongoReplicaSetClient.new(seeds)
client = MongoReplicaSetClient.new(seeds, :read => :secondary)

# connecting to a sharded cluster
client = MongoShardedClient.new(seeds)
client = MongoShardedClient.new(seeds, :read => :secondary)

# using a connection string
ENV['MONGODB_URI'] = 'mongodb://host1:27017?ssl=true'
client = MongoClient.new
```

Using the Ruby Driver

```
# using a database
db = client.db('blog')
db = client['blog']

client.drop_database('blog')
client.database_names

# using a collection
coll = db['posts']

coll.drop
db.collection_names
```

Terminology

RDBMS		MongoDB
Table, View	➜	Collection
Row	➜	Document
Index	➜	Index
Join	➜	Embedded Document
Foreign Key	➜	Reference
Partition	➜	Shard

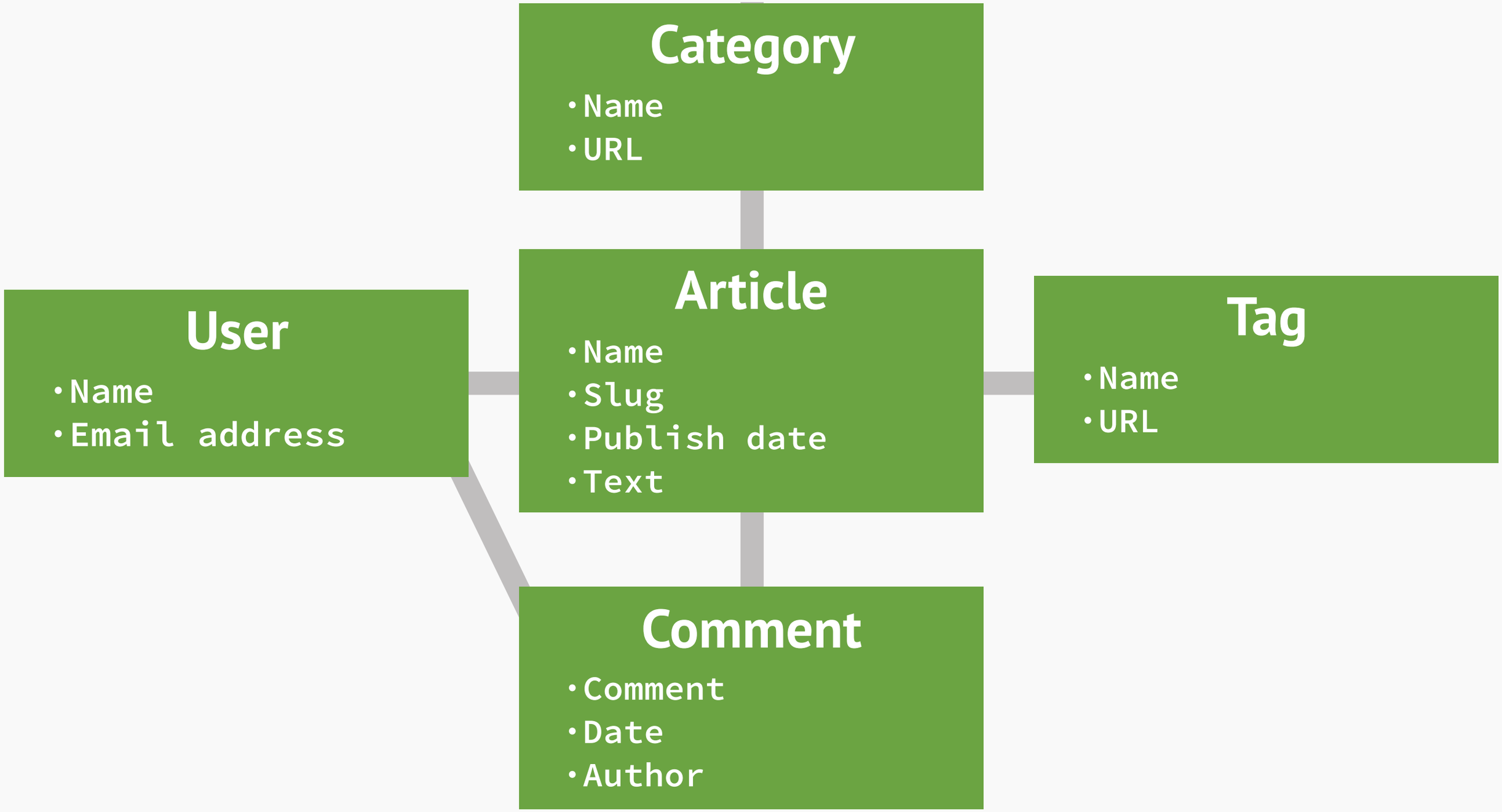
Building a Blog

Models/Entities:

- Users (Authors)
- Articles
- Comments
- Tags/Categories

In a relational application
we would start by defining
our schema.

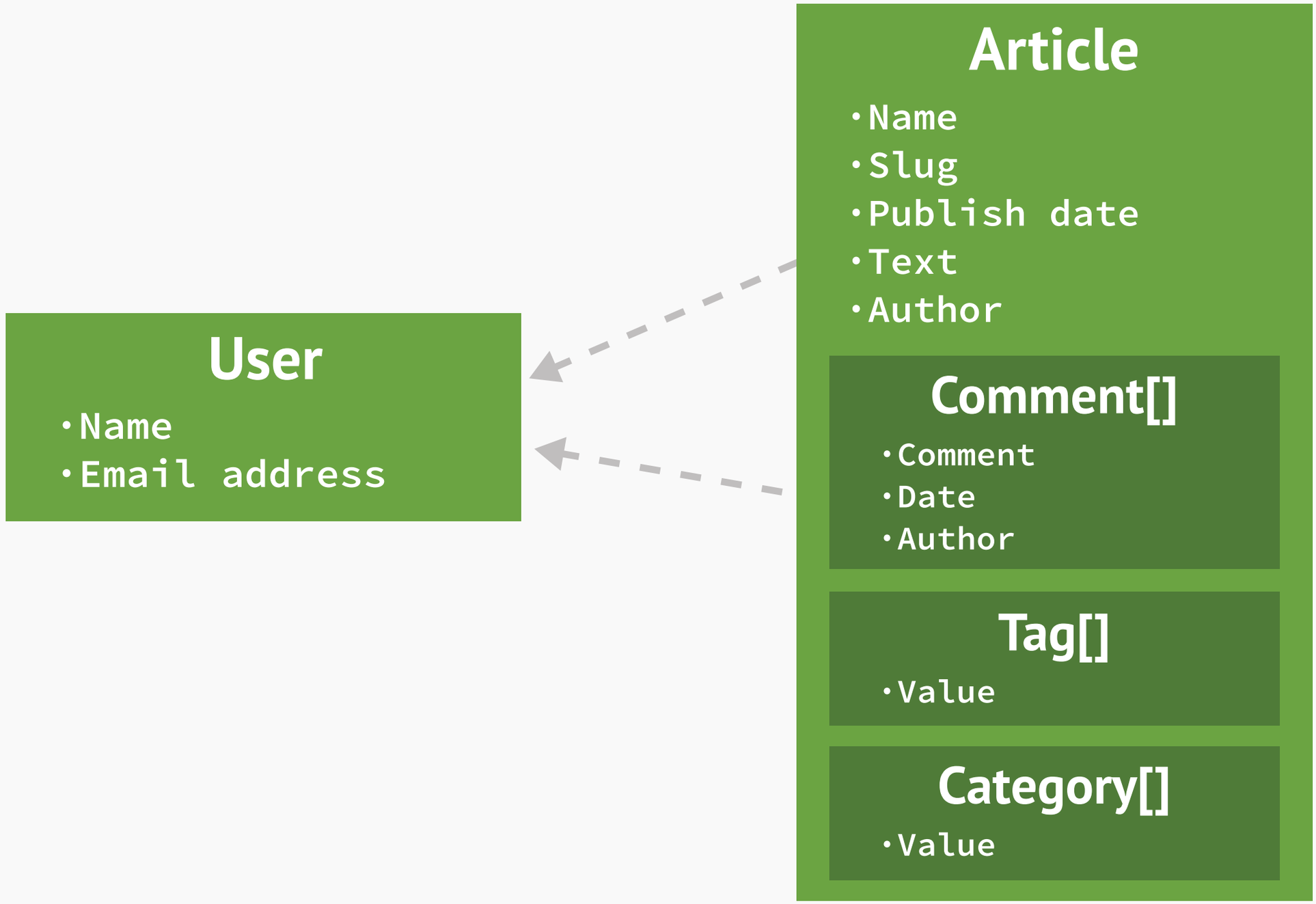
Typical Relational Diagram



In MongoDB we start building and we let the schema evolve as we go.

Like Ruby, it has a natural and enjoyable feeling to it!

MongoDB Diagram



Inserting a New Document

```
# example document
author = {
  :username => 'brandonblack',
  :first    => 'brandon',
  :last     => 'black'
}

# inserting into my blog database
client['blog']['users'].insert(author)
```

No database or collection creation required!

Inserting a New Document

```ruby
# example document
article = {
  :title    => 'Hello World'
  :username => 'brandonblack',
  :tags     => ['ruby', 'getting started'],
  :comments => [ # embedded docs ]
}
```

```ruby
# inserting into my blog database
client['blog']['articles'].insert(article)
```

No database or collection creation required!

Finding Documents

```ruby
coll = client['blog']['users']

# finding a single document
coll.find_one
coll.find_one({ :username => 'brandonblack' })
coll.find_one({ :username => 'brandonblack' }, { :first => 1})

coll = client['blog']['articles']

# finding multiple documents (using cursors)
cursor = coll.find({ :username => 'brandonblack' }, :limit => 10)
cursor.each do |article|
  puts article['title']
end
```

Updating and Deleting Documents

```
# updating a document
article = { :title => 'Hello Ruby' }
coll.update({ '_id' => article_id }, article)
coll.update({ '_id' => article_id },
            {'$set' => {:title => 'Hello Ruby'}})

# deleting a single document
coll.remove({ '_id' => article_id })

# delete multiple documents
coll.remove({ 'username' => 'brandonblack' })

# delete all documents in a collection
coll.remove
```

Indexes

```ruby
# running explain on a query
coll.find({ :username => 'brandonblack' }).explain

# showing collection indexes
coll.index_information

# adding an index
coll.ensure_index({ :username => 1 })
coll.ensure_index({ :username => Mongo::ASCENDING })

coll.ensure_index({ :username => -1 })
coll.ensure_index({ :username => Mongo::DESCENDING })

# adding a special index types
coll.ensure_index({ :loc => Mongo::GEO2D })
coll.ensure_index({ :title => Mongo::TEXT })

# droping an index
coll.drop_index('username_1')

# dropping all indexes for a collection
coll.drop_indexes
```

ODMs (Object Document Mappers)

Mongoid (http://mongoid.org/)

Mongo Mapper (http://mongomapper.com/)

Mongoid Example

```
# rails setup
rails g mongoid:config

# non-rails setup
Mongoid.load!("path/to/your/mongoid.yml", :production)

# document examples
class Article
  include Mongoid::Document
  field :title, type: String
  embeds_many :comments
end

class Comment
  include Mongoid::Document
  field :comment_text, type: String
  embedded_in :article
end
```

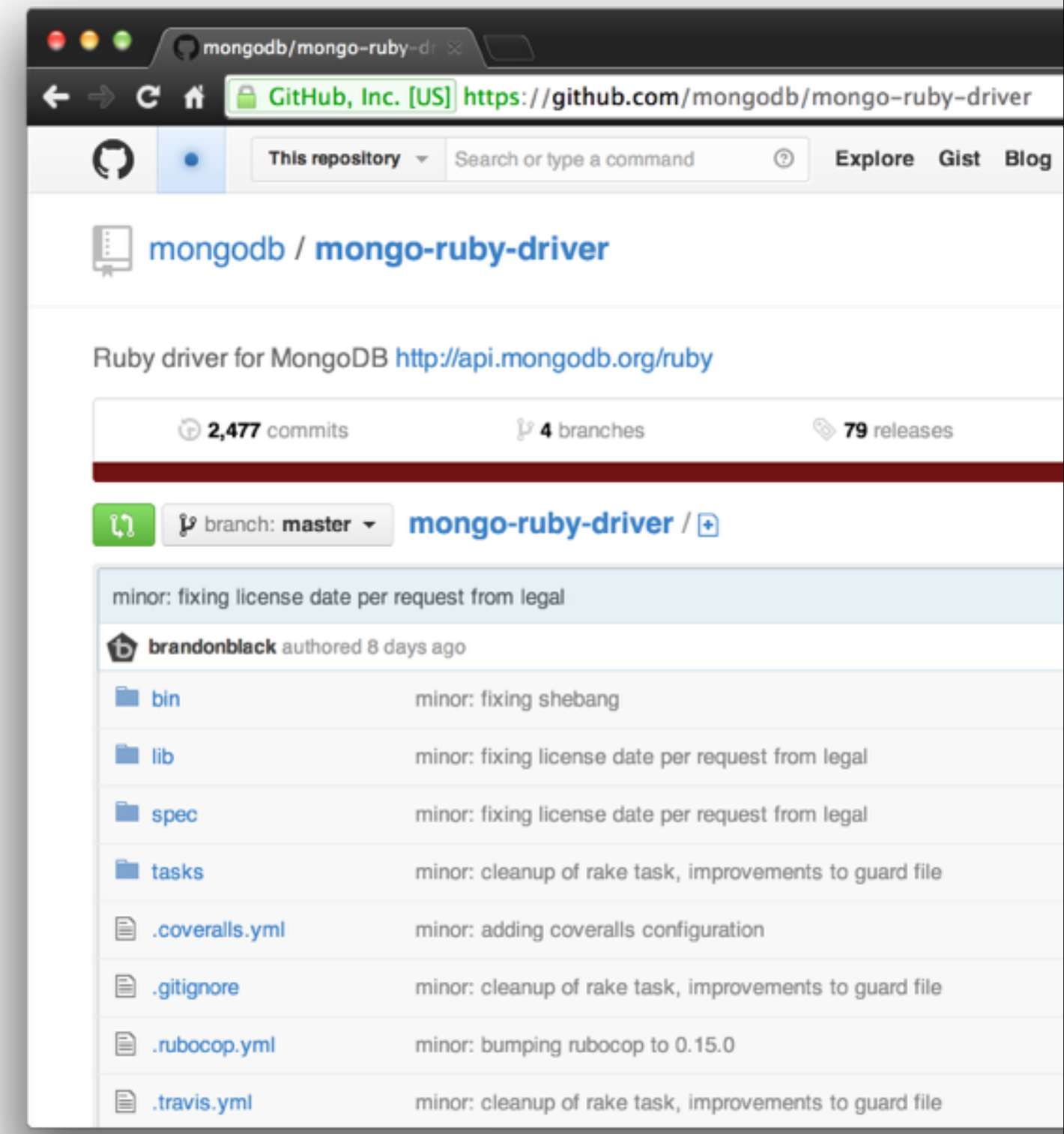
What Next?

mongoDB

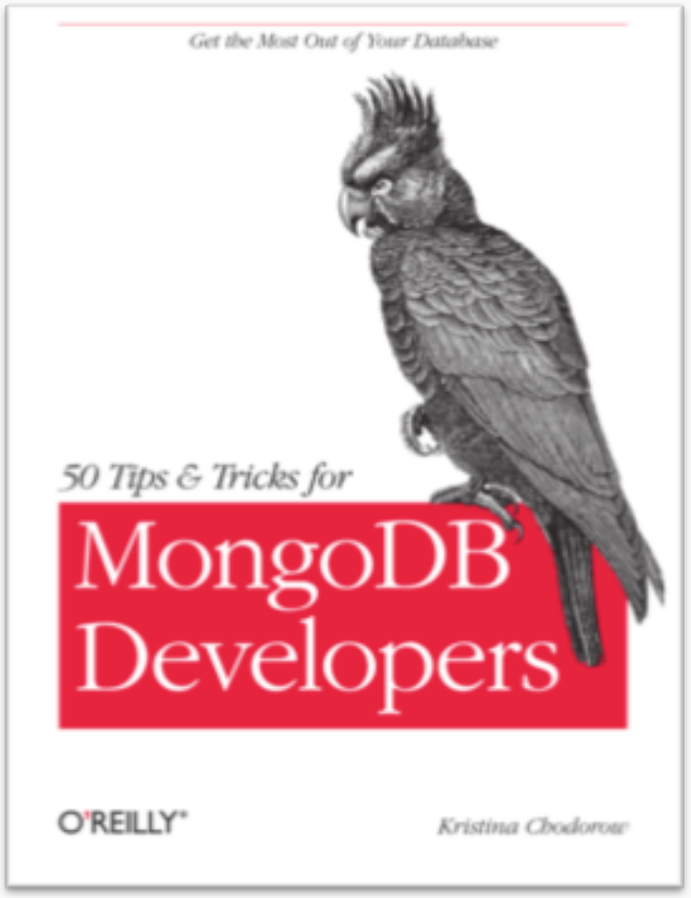
We've covered a lot of ground quickly.

Come Find Us!

- Github
- Stack Overflow
- MongoDB User Group



Further Reading



More Ways to Learn

Free Online Courses

http://education.mongodb.com/

Events & Webinars

http://www.mongodb.com/events

Presentations

http://www.mongodb.com/presentations

Thank you!

Brandon Black

Software Engineer, MongoDB
@brandonmblack

mongoDB