MongoDB

Hands-on test drive

What is MongoDB?

- No SQL
- High Performance
- High Availability
- Horizontal Scaling
- Document Store

Stores BSON Documents


```
{
    "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),
    "title" : "The Incredibles",
    "plot" : "A family of undercover superheroes, while trying to live the quiet suburban life, are forced into action to save the world.",
    "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],
    "details" : {
        "year" : "2004",
        "rated" : "PG",
        "genre" : "Animation",
        "director" : "Brad Bird"
    },
    "imdbRating" : "8.0"
}
```

Stores BSON Documents


```
{
    "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),
    "title" : "The Incredibles",
    "plot" : "A family of undercover superheroes, while trying to live the quiet suburban life, are forced into action to save the world.",
    "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],
    "details" : {
        "year" : "2004",
        "rated" : "PG",
        "genre" : "Animation",
        "director" : "Brad Bird"
    },
    "imdbRating" : "8.0"
}
```

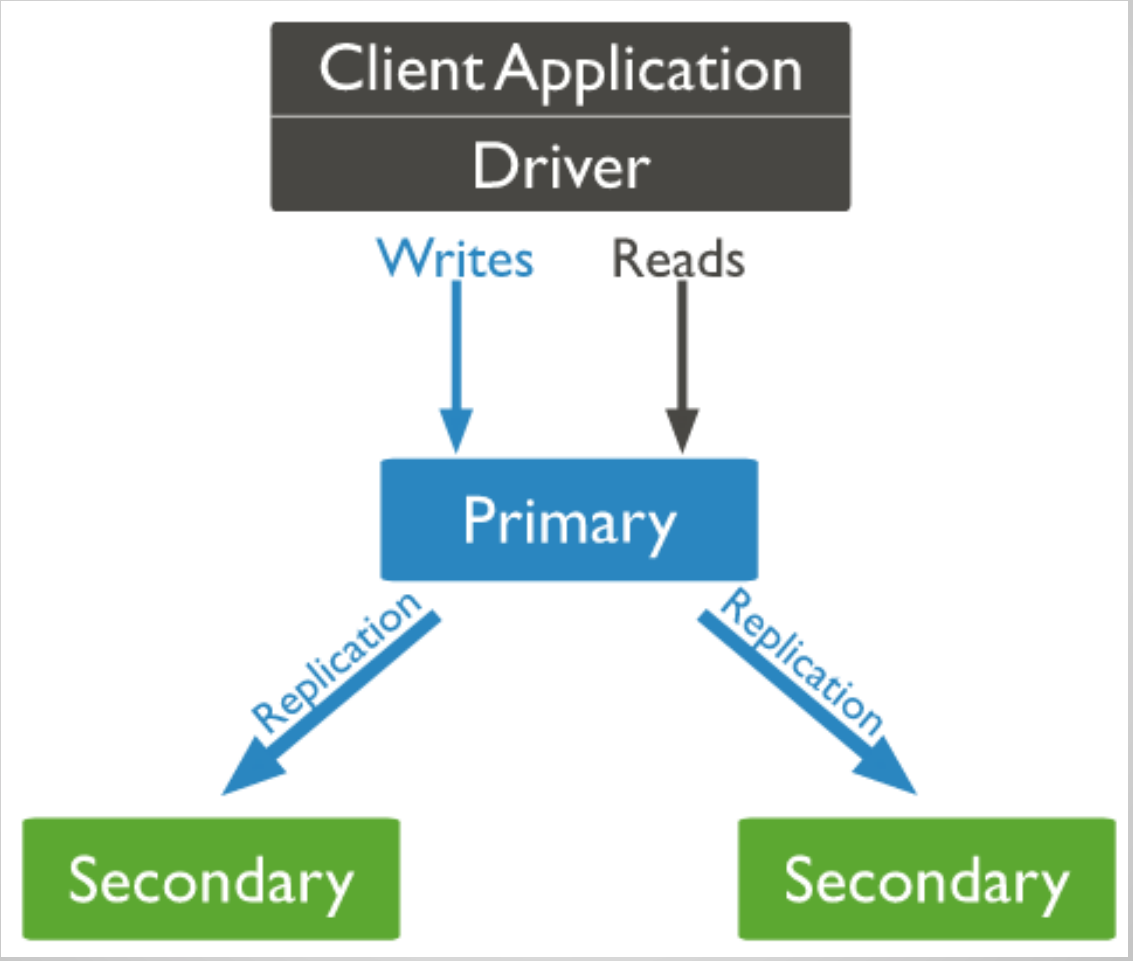
Stores BSON Documents


```
{
    "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),
    "title" : "The Incredibles",
    "plot" : "A family of undercover superheroes, while trying to live the quiet suburban life, are forced into action to save the world.",
    "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],
    "details" : {
        "year" : "2004",
        "rated" : "PG",
        "genre" : "Animation",
        "director" : "Brad Bird"
    },
    "imdbRating" : "8.0"
}
```

Stores BSON Documents

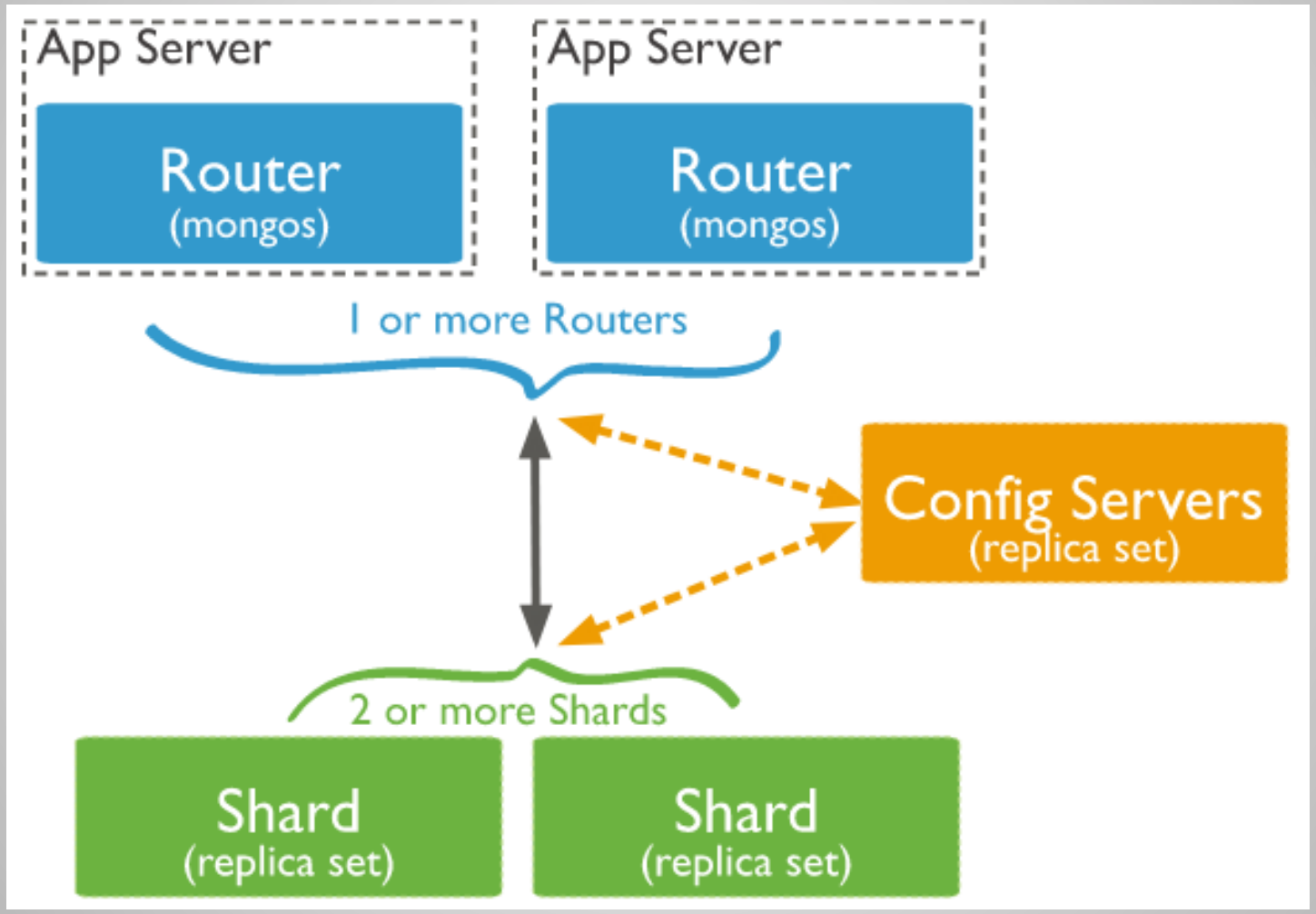

```
{
    "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),
    "title" : "The Incredibles",
    "plot" : "A family of undercover superheroes, while trying to live the quiet suburban life, are forced into action to save the world.",
    "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],
    "details" : {
        "year" : "2004",
        "rated" : "PG",
        "genre" : "Animation",
        "director" : "Brad Bird"
    },
    "imdbRating" : "8.0"
}
```

Replica Sets



https://docs.mongodb.com/manual/replication/

Sharding



https://docs.mongodb.com/manual/sharding/

Getting started: install mongo

Install Mongo

- https://docs.mongodb.com/manual/administration/install-community/
- Create the default directory for the database:

  `md /data/db`

  (don't forget to use \ for windows)

Start Mongo

- Start the server:

  `mongod`
- Start the command shell (from Intellij Terminal helps):

  `mongo`
- In the shell, display the mongo version running:

  `db.version()`

Try Out The Shell

Some Simple Commands

List your databases:

```
show dbs
```

Start using a database (or create one) :

```
use exercises
```

List the collections in the database:

```
show collections
```

Getting started: insert data

Grab slides and run insert script:

- From Git:
  https://github.com/dvanvali/mongoDbPresentation
- From the Wiki:
  <fill me in>
- Insert data from the script
  ```
  cd mongoDbPresentation
  mongo moviesForMongo.js
  ```
- Check to see that data is there
  ```
  $ mongo
  > db.movies.find().count()
  12
  ```

Querying Documents

db.<collection>.find()

- queries everything

db.<collection>.find(selector, projection)

- Selects documents and limits fields selected

db.movies.find({title: “The Incredibles”}, {plot: 1})

- adding .pretty() will format the output

Selectors

Search value can also be an operator:

$lt	less than
$lte	less than or equal to
$gt	greater than
$gte	greater than or equal to
$ne	not equal

```
db.movies.find( { imdbRating: { $gt: 8 } })
```

Querying Documents

Conjunctions can be used with an array:

$or

$and

```
db.<collection>find(
        {$or: [ { key1: val1 }, { key1: val1 } ] } )
```

Querying Exercise

Find all movies in our database with the actor Ian McKellen

List the names of directors for all movies

List the title and imdbRating for movies since 2000 or that have a rating great than 8

Pipeline Operators

Find can be followed with pipeline operators in the shell. Aggregation can be used outside the shell:

.pretty()

.limit(rows)

.skip(rows)

.sort({“key”: order})

order:1 for ascending, -1 for descending

Updates

Save: create or overwrite a document:

```
db.<collection>.save(document)
```

Update: a portion of a document:

```
db.<collection>.update(criteria, { op : portion })
```

Remove: remove documents:

```
db.<collection>.remove(criteria,  optional_limit)
```

```
> db.movies.aggregate([
      {$group: {_id: "$details.year",
                  average: {$avg: "$imdbRating"}}},
      {$sort: {_id: 1}} ])
{ "_id" : "1977", "average" : 8.7 }
{ "_id" : "1989", "average" : 7.6 }
{ "_id" : "2000", "average" : 7.4 }
{ "_id" : "2001", "average" : 8.8 }
{ "_id" : "2002", "average" : 7.3 }
{ "_id" : "2004", "average" : 7.55 }
{ "_id" : "2005", "average" : 5.7 }
{ "_id" : "2008", "average" : null }
{ "_id" : "2012", "average" : 8 }
{ "_id" : "2017", "average" : 8.4 }
>
```

Updates

db.movies.update(selector, document with optional operator)

```
db.movies.update({title: "The Incredibles"},
        {$set: {boxOffice: "$261,441,092"}})
```

Array Operators

$addToSet	Adds elements to an array only if they do not already exist in the set.
$pop	Removes the first or last item of an array.
$pull	Removes all array elements that match a specified query.
$push	Adds an item to an array.

Updates Exercises

Add a boxOffice of value of $251,188,924 To Batman.

Add “Jack Palance” to the actors for Batman.

Add “Action” to the genre for “The Incredibles”. Check to see it now has three genres.

Other Operators

$inc	**Increments the value of the field by the specified amount.**
$mul	**Multiplies the value of the field by the specified amount.**
$unset	**Removes the specified field from a document.**
$rename	**Renames a field.**

Atomic Updates

```
> db.movies.findAndModify( {query: selector},
        update:{operators});
```

Validation

- Occurs during insert and update
- A selector that must pass
- Can be set to different levels
  - Strict – will not allow a validation failure
  - Moderate – only forces failure if the document is already valid

Validation

```
db.createCollection( "contacts",
   { validator:
      { $or:
         [ { phone: { $type: "string" } },
            { email: { $regex: /@mongodb\.com$/ } },
            { status: { $in: [ "Unknown", "Incomplete" ] } } ]
      }
   }
)
```

MEAN example

- MongoDB
- Express
- Angular
- Node

Angular view

```
<div ng-repeat="todo in todos | orderBy: '!highPriority'”
  …
  <div class="col-md-1 col-sm-1 col-xs-1">
    <span ng-class="{'glyphicon glyphicon-star':
                                todo.highPriority,
                            'glyphicon glyphicon-star-empty':
                                !todo.highPriority}”
      ng-click="updatePriority($event, todo)">
    </span>
  </div>
```

Angular Controller

```
$scope.updatePriority = function($event, _t) {
  var cbk = !_t.highPriority;
  todosFactory.updateTodo({
    _id: _t._id,
    highPriority: cbk,
    isCompleted: _t.isCompleted,
    todo: _t.todo
  }).then(function(data) {
    if (data.data.ok) {
      _t.highPriority = cbk;
    } else {
      alert('Oops something went wrong!');
    }
  });
};
```

Express Server

```
router.put('/api/todos', function(req, res) {
  var id = req.body._id;
  delete req.body["_id"];
  db.todos.update({
    _id: mongojs.ObjectId(id)
  }, req.body, {}, function(err, data) {
    res.json(data);
    req.body["_id"] = id;
  });
});
```

Express Server

```
router.put('/api/todos', function(req, res) {
  var id = req.body._id;
  delete req.body["_id"];
  db.todos.update({
    _id: mongojs.ObjectId(id)
  }, req.body, {}, function(err, data) {
    res.json(data);
    req.body["_id"] = id;
  });
});
```