MongoDB

Hands-on test drive

What is MongoDB?

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

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"
}
```

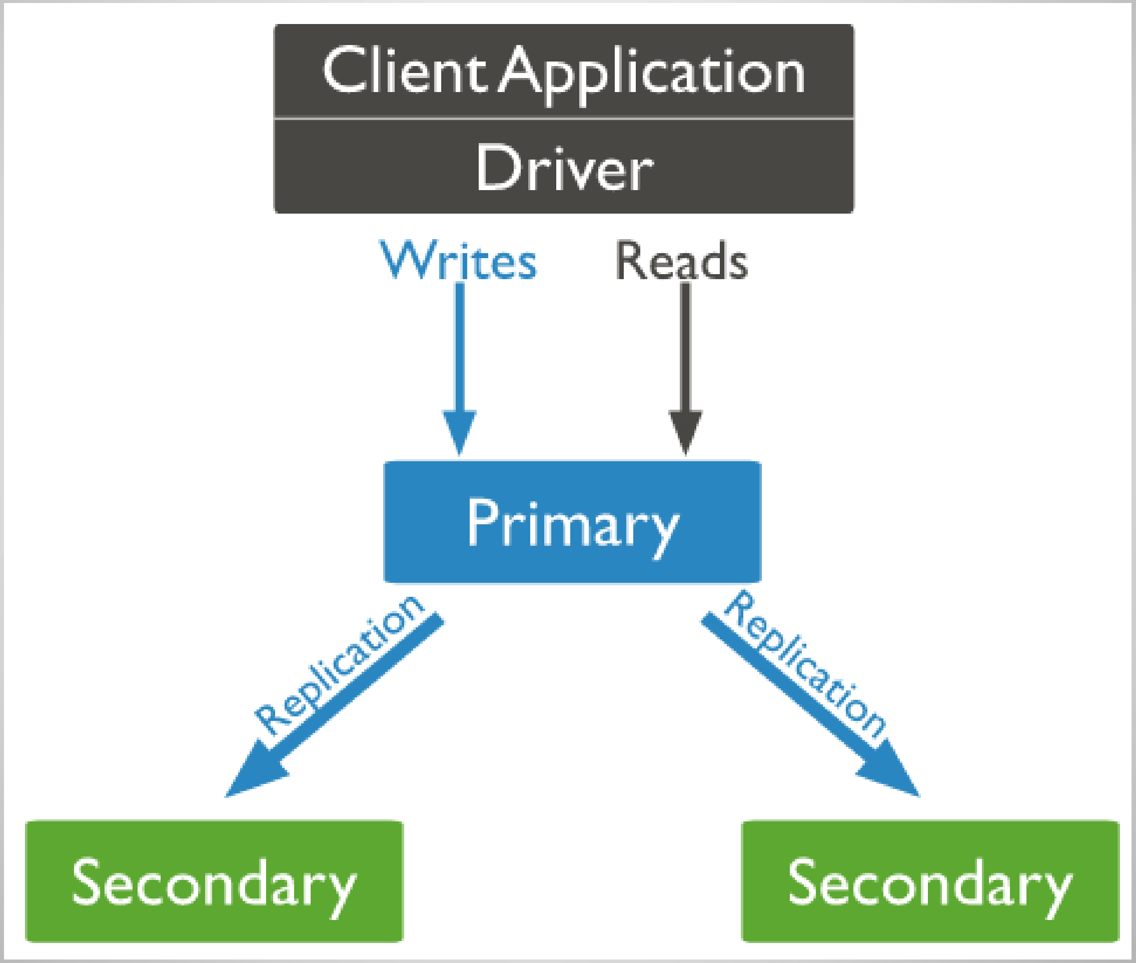
Stores BSON Documents

```
{
	"_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

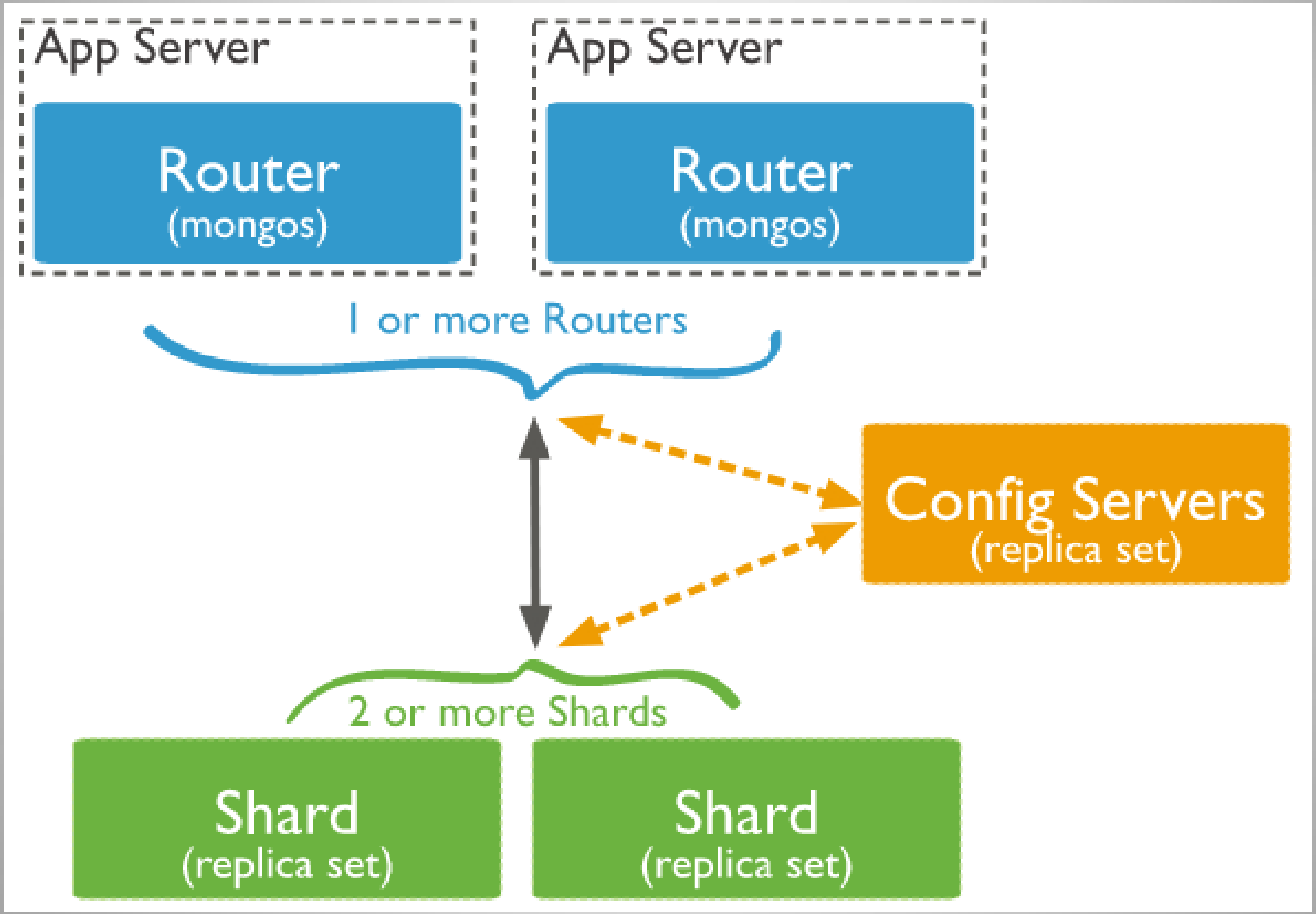
```
{
    "_id" : ObjectId("5932cdb9c72a3fb0c0ecbd75"),
    "title" : "The Incredibles",
    "plot" : "A family of undercover superheroes, while trying to live the quiet suburban life, are forced into action to save the world.",
    "actors" : [ "Craig T. Nelson", "Samuel L. Jackson" ],
    "details"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

High Availability: Replica Sets



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

Horizontal Scaling: Sharding



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

Getting started: install mongo

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 <db name>
```

List the collections in the database:

```
show collections
```

Querying Documents

db.<collection>.find()

– queries everything

Check to see if you have the movies data inserted:

```
db.movies.find()
```

Should display 12 documents.

Pipeline Operators

Find can be followed with pipeline operators in the shell:

.pretty()

.limit(rows)

.skip(rows)

.sort({“key”: order**})**

order:1 for ascending, -1 for descending

Querying Documents

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

- Selector and projection look like javascript objects
- The simplest selector just matches object fields
- Projection takes document fields and either displays or removes from the result: 1=display, 0=remove

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

- Returns the plot for Batman
- Also returns the _id field

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( { title: { $gt: “Spider” } })
```

Querying Documents

Conjunctions can be used with an array:

```
$or
$and

db.movies.find(
    {$or: [ { title: “Batman” },
            { title: “The Incredibles” } ] } )
```

Querying Documents

You can reference embedded documents with dot notation:

```
db.movies.find({“details.rated”: “PG”})
```

Querying Exercise

1. Find all movie titles in our database with the actor Ian McKellen
2. List the names of directors for all movies
3. List the title and imdbRating for movies since 2000 or that have a rating greater than 8

Example:
```
db.movies.find( { imdbRating: { $gt: 8 } },
                {title: 1, _id: 0})
```

Operators: $lt, $lte, $gt, $gte, $ne

Solutions

1. Find all movie titles in our database with the actor Ian McKellen

```
db.movies.find({actors: "Ian McKellen"}, {title: 1, _id: 0})
```

Solutions

1. Find all movie titles in our database with the actor Ian McKellen

```
db.movies.find({actors: "Ian McKellen"}, {title: 1, _id: 0})
```

2. List the names of directors for all movies

```
db.movies.find({},{"details.director":1, _id: 0})
```

Solutions

1. Find all movie titles in our database with the actor Ian McKellen

```
db.movies.find({actors: "Ian McKellen"}, {title: 1, _id: 0})
```

2. List the names of directors for all movies

```
db.movies.find({},{"details.director":1, _id: 0})
```

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

```
db.movies.find({$or: [{"details.year": {$gte: 2000}}, {imdbRating: {$gt:
8}}]},  {title: 1, imdbRating: 1, _id: 0})
```

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)
```

Updates

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

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

Array Operators

$pop	Removes the first or last item of an array.
$push	Adds an item to an array.
$addToSet	Adds elements to an array only if they do not already exist in the set.

Updates Exercises

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

2. Add “Jack Palance” to the actors for Batman.

3. Add “Action” to the genre for “The Incredibles”.

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

Solutions

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

```
db.movies.update({title:"Batman"},{$set: {boxOffice: “$251,188,924”}})
```

Solutions

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

db.movies.update({title:"Batman"},{$set: {boxOffice: “$251,188,924”}})

2. Add “Jack Palance” to the actors for Batman.

db.movies.update({title:"Batman"},{$push: {actors: "Jack Palance"}})

Solutions

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

```
db.movies.update({title:"Batman"},{$set: {boxOffice: “$251,188,924”}})
```

2. Add “Jack Palance” to the actors for Batman.

```
db.movies.update({title:"Batman"},{$push: {actors: "Jack Palance"}})
```

3. Add “Action” to the genre for “The Incredibles”.

```
db.movies.update({title:“The Incredibles"}, {$push: {"details.genre": “Action"}})
```

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: {selectors},
          update:{operators});
```

- Select the row to update and then update in a single operation

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( “movies",
  { validator: { $and:
      [ { title: { $type: "string" } },
        { plot: { $type:  "string"} },
        {imdbRating: {$lte: 10 } } ]
   } } )
```

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;
  });
 });
```