Joshua Catoe

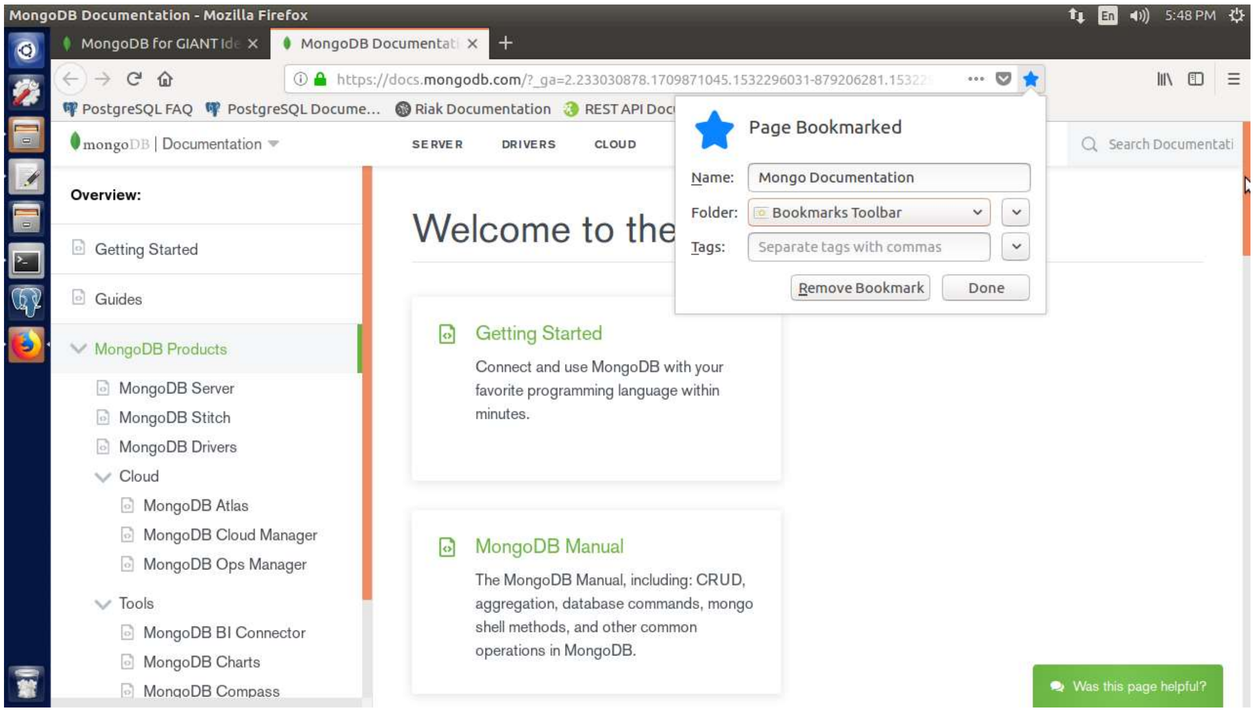
07/29/18

MongoDB

Day 1

Find:

1. Bookmark the online MongoDB documentation.



Page Bookmarked

Name: Mongo Documentation

Folder: Bookmarks Toolbar

Tags: Separate tags with commas

Getting Started

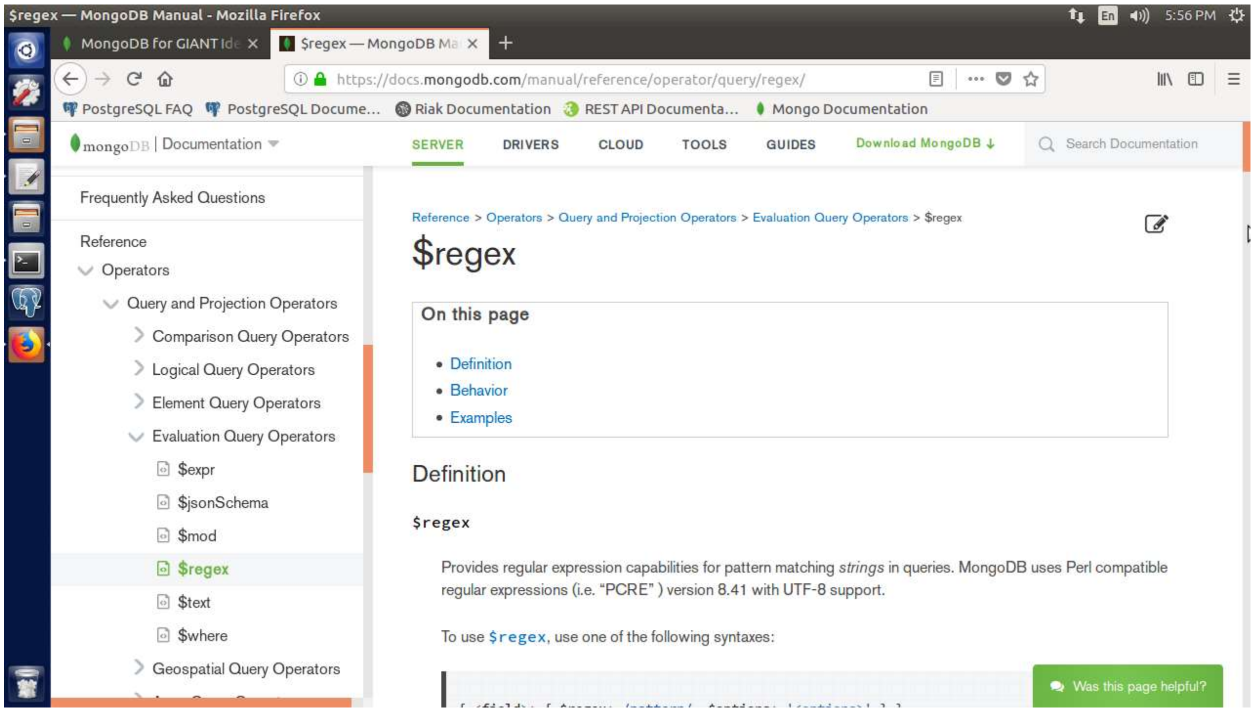
Connect and use MongoDB with your favorite programming language within minutes.

MongoDB Manual

The MongoDB Manual, including: CRUD, aggregation, database commands, mongo shell methods, and other common operations in MongoDB.

The MongoDB documentation can be found by going to www.mongodb.com and clicking on "DOCS" in the top left corner.

2. Look up how to construct regular expressions in Mongo.



$regex

Definition

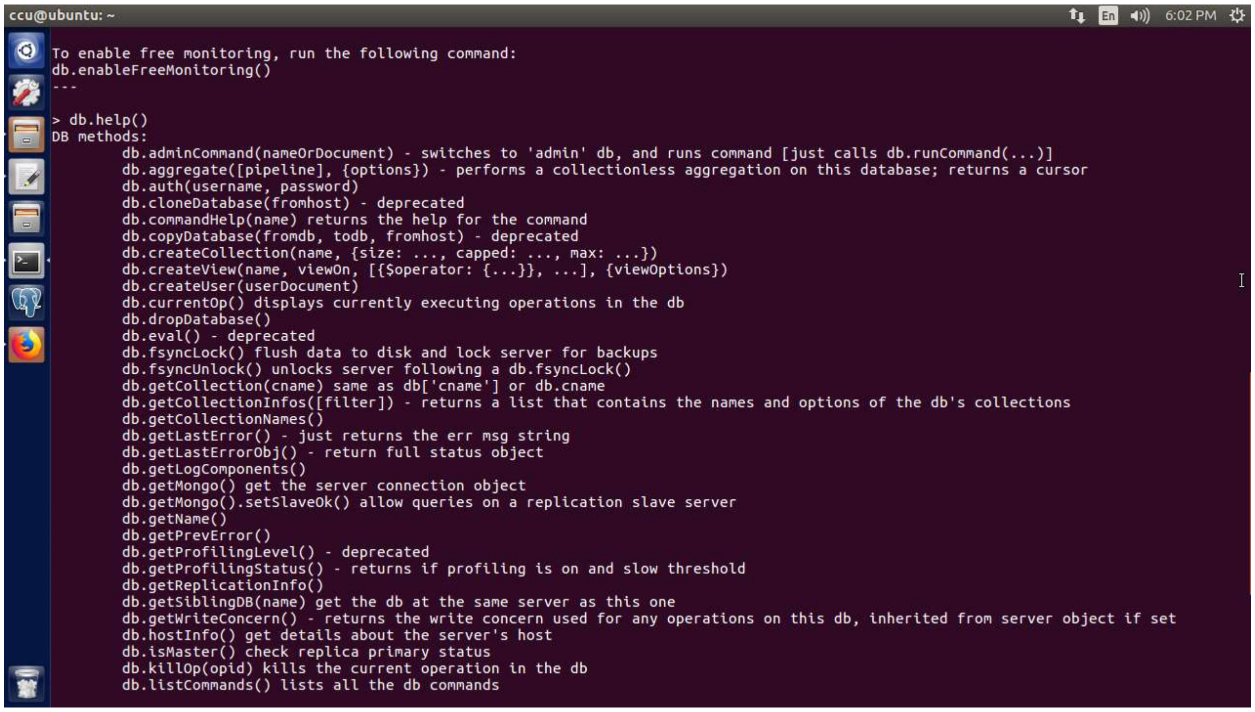
`$regex`

Provides regular expression capabilities for pattern matching *strings* in queries. MongoDB uses Perl compatible regular expressions (i.e. "PCRE") version 8.41 with UTF-8 support.

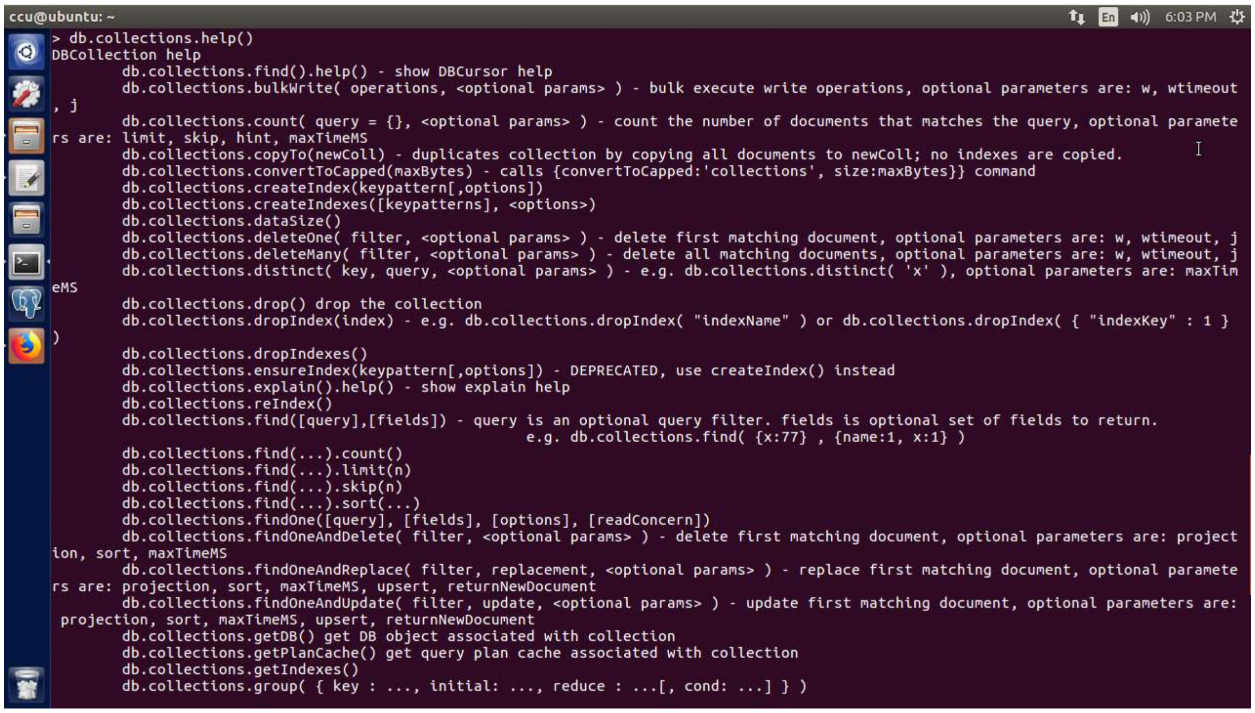
To use `$regex`, use one of the following syntaxes:

The MongoDB documentation contains a section specifically for **$regex**. It can be found by going to the main docs page and clicking on MongoDB Manual -> Reference -> Operators -> Query and Projection Operators -> Evaluation Query Operators -> $regex

3. Acquaint yourself with command-line db.help() and db.collections.help() output.



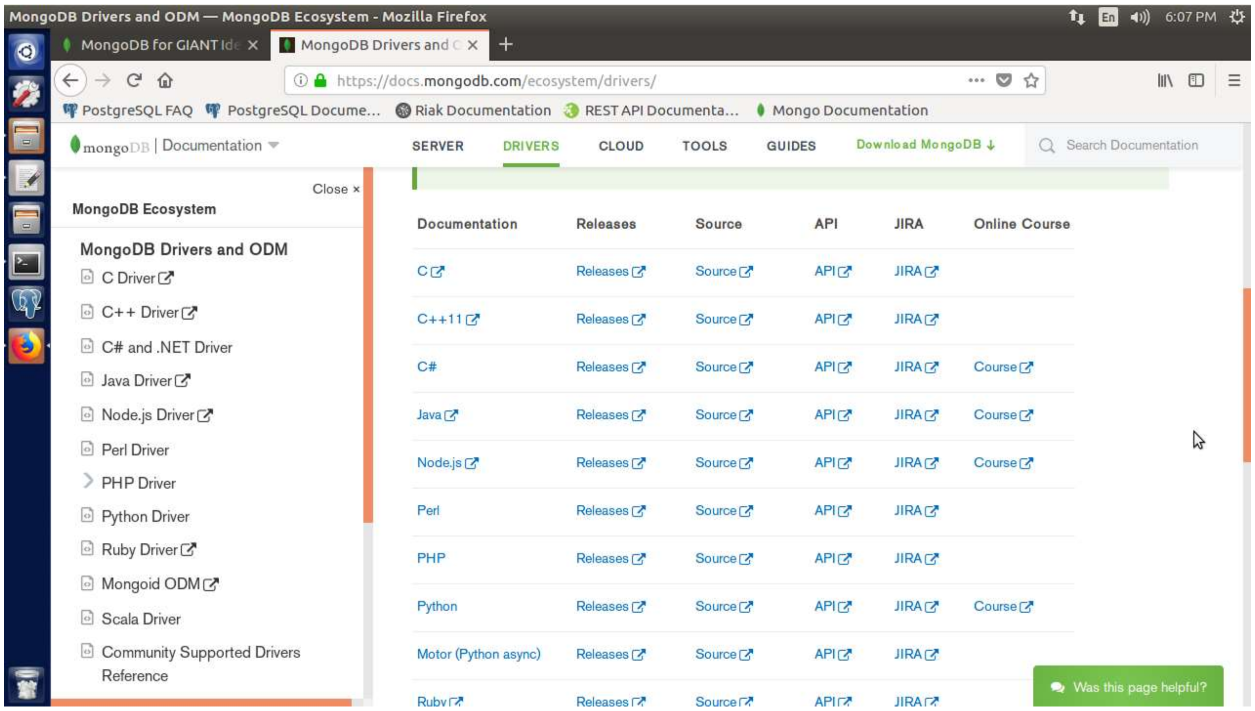
```
To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.help()
DB methods:
        db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls db.runCommand(...)]
        db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this database; returns a cursor
        db.auth(username, password)
        db.cloneDatabase(fromhost) - deprecated
        db.commandHelp(name) returns the help for the command
        db.copyDatabase(fromdb, todb, fromhost) - deprecated
        db.createCollection(name, {size: ..., capped: ..., max: ...})
        db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
        db.createUser(userDocument)
        db.currentOp() displays currently executing operations in the db
        db.dropDatabase()
        db.eval() - deprecated
        db.fsyncLock() flush data to disk and lock server for backups
        db.fsyncUnlock() unlocks server following a db.fsyncLock()
        db.getCollection(cname) same as db['cname'] or db.cname
        db.getCollectionInfos([filter]) - returns a list that contains the names and options of the db's collections
        db.getCollectionNames()
        db.getLastError() - just returns the err msg string
        db.getLastErrorObj() - return full status object
        db.getLogComponents()
        db.getMongo() get the server connection object
        db.getMongo().setSlaveOk() allow queries on a replication slave server
        db.getName()
        db.getPrevError()
        db.getProfilingLevel() - deprecated
        db.getProfilingStatus() - returns if profiling is on and slow threshold
        db.getReplicationInfo()
        db.getSiblingDB(name) get the db at the same server as this one
        db.getWriteConcern() - returns the write concern used for any operations on this db, inherited from server object if set
        db.hostInfo() get details about the server's host
        db.isMaster() check replica primary status
        db.killOp(opid) kills the current operation in the db
        db.listCommands() lists all the db commands
```



```
> db.collections.help()
DBCollection help
        db.collections.find().help() - show DBCursor help
        db.collections.bulkWrite( operations, <optional params> ) - bulk execute write operations, optional parameters are: w, wtimeout
, j
        db.collections.count( query = {}, <optional params> ) - count the number of documents that matches the query, optional paramete
rs are: limit, skip, hint, maxTimeMS
        db.collections.copyTo(newColl) - duplicates collection by copying all documents to newColl; no indexes are copied.
        db.collections.convertToCapped(maxBytes) - calls {convertToCapped:'collections', size:maxBytes}} command
        db.collections.createIndex(keypattern[,options])
        db.collections.createIndexes([keypatterns], <options>)
        db.collections.dataSize()
        db.collections.deleteOne( filter, <optional params> ) - delete first matching document, optional parameters are: w, wtimeout, j
        db.collections.deleteMany( filter, <optional params> ) - delete all matching documents, optional parameters are: w, wtimeout, j
        db.collections.distinct( key, query, <optional params> ) - e.g. db.collections.distinct( 'x' ), optional parameters are: maxTim
eMS
        db.collections.drop() drop the collection
        db.collections.dropIndex(index) - e.g. db.collections.dropIndex( "indexName" ) or db.collections.dropIndex( { "indexKey" : 1 }
)
        db.collections.dropIndexes()
        db.collections.ensureIndex(keypattern[,options]) - DEPRECATED, use createIndex() instead
        db.collections.explain().help() - show explain help
        db.collections.reIndex()
        db.collections.find([query],[fields]) - query is an optional query filter. fields is optional set of fields to return.
                                              e.g. db.collections.find( {x:77} , {name:1, x:1} )
        db.collections.find(...).count()
        db.collections.find(...).limit(n)
        db.collections.find(...).skip(n)
        db.collections.find(...).sort(...)
        db.collections.findOne([query], [fields], [options], [readConcern])
        db.collections.findOneAndDelete( filter, <optional params> ) - delete first matching document, optional parameters are: project
ion, sort, maxTimeMS
        db.collections.findOneAndReplace( filter, replacement, <optional params> ) - replace first matching document, optional paramete
rs are: projection, sort, maxTimeMS, upsert, returnNewDocument
        db.collections.findOneAndUpdate( filter, update, <optional params> ) - update first matching document, optional parameters are:
 projection, sort, maxTimeMS, upsert, returnNewDocument
        db.collections.getDB() get DB object associated with collection
        db.collections.getPlanCache() get query plan cache associated with collection
        db.collections.getIndexes()
        db.collections.group( { key : ..., initial: ..., reduce : ...[, cond: ...] } )
```

For both **db.help()** and **db.collections.help()**, they just have to be entered into the mongo shell directly to display their contents.

4. Find a Mongo driver in your programming language of choice (Ruby, Java, PHP, and so on).



MongoDB Ecosystem

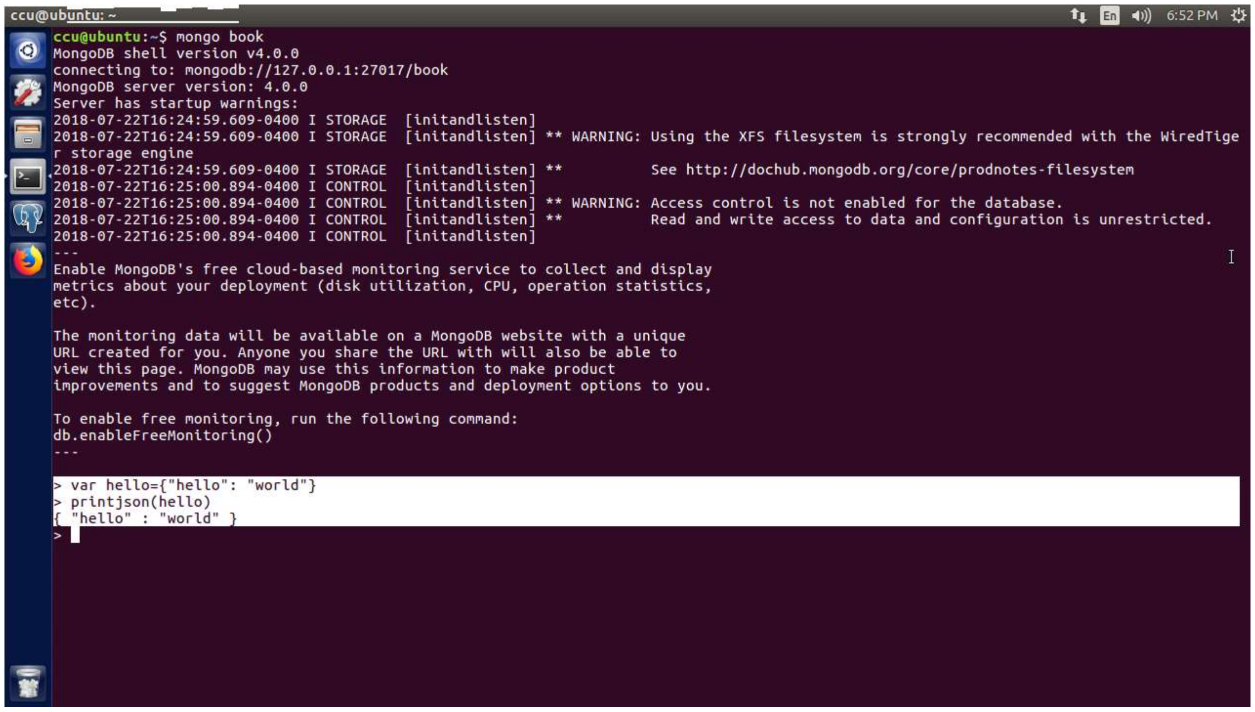
MongoDB Drivers and ODM
- C Driver
- C++ Driver
- C# and .NET Driver
- Java Driver
- Node.js Driver
- Perl Driver
- PHP Driver
- Python Driver
- Ruby Driver
- Mongoid ODM
- Scala Driver
- Community Supported Drivers Reference

Documentation	Releases	Source	API	JIRA	Online Course
C	Releases	Source	API	JIRA	
C++11	Releases	Source	API	JIRA	
C#	Releases	Source	API	JIRA	Course
Java	Releases	Source	API	JIRA	Course
Node.js	Releases	Source	API	JIRA	Course
Perl	Releases	Source	API	JIRA	
PHP	Releases	Source	API	JIRA	
Python	Releases	Source	API	JIRA	Course
Motor (Python async)	Releases	Source	API	JIRA	
Ruby	Releases	Source	API	JIRA	

The Mongo documentation has a list of all drivers available by clicking the “DRIVERS” tab at the top of any page. My language of choice would be C, which happens to be the first listed.

Do:

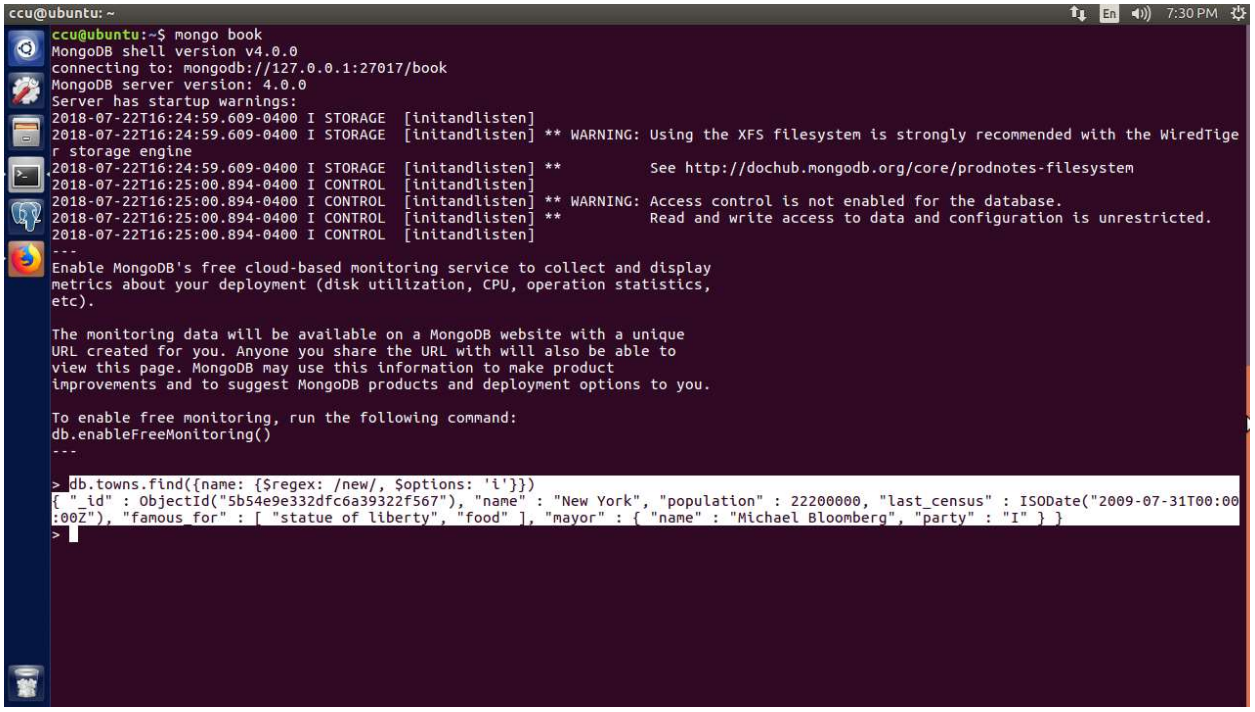
1. Print a JSON document containing {"hello": "world"}.



```
ccu@ubuntu:~$ mongo book
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017/book
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten]
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTige
r storage engine
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> var hello={"hello": "world"}
> printjson(hello)
{ "hello" : "world" }
>
```

For this, I stored a JSON inside a variable called **hello** and then printed the contents with the **printjson()** command. At first, I tried the regular **print()**, but it gave me the type of contents, that being **{object Object}**, instead of the actual contents, so **printjson()** is required to view JSON contents.

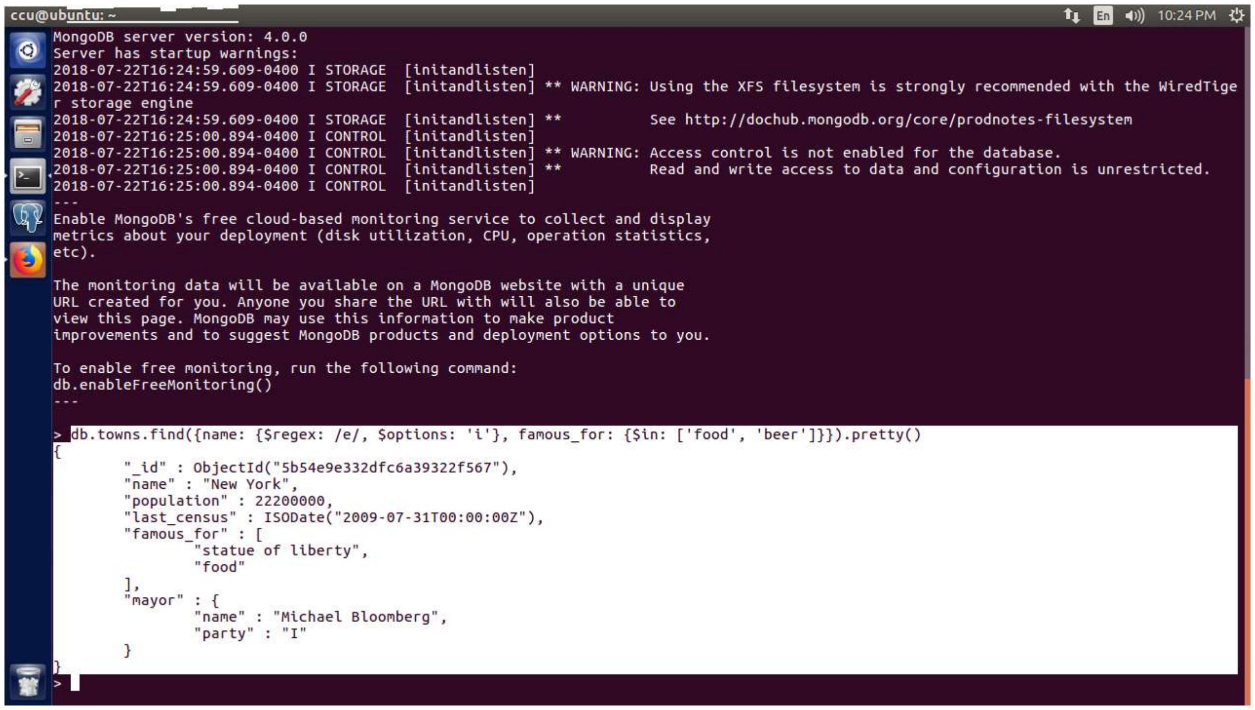
2. Select a town via a case-insensitive regular expression containing the word *new*.



```
ccu@ubuntu:~$ mongo book
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017/book
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten]
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTige
r storage engine
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.towns.find({name: {$regex: /new/, $options: 'i'}})
{ "_id" : ObjectId("5b54e9e332dfc6a39322f567"), "name" : "New York", "population" : 22200000, "last_census" : ISODate("2009-07-31T00:00
:00Z"), "famous_for" : [ "statue of liberty", "food" ], "mayor" : { "name" : "Michael Bloomberg", "party" : "I" } }
> 
```

I had to visit the $regex page in the Mongo docs to determine the correct syntax. Then, I searched for the word **new** with the option **‘i’** for case insensitivity.

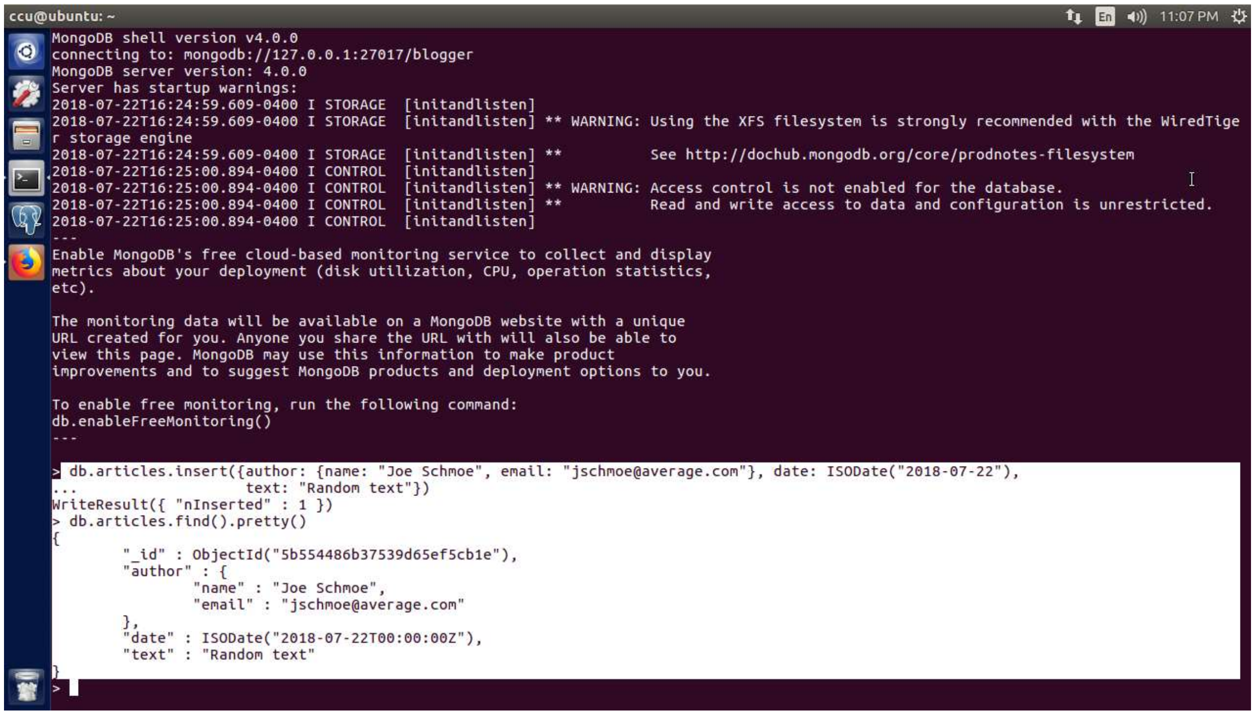
3. Find all cities whose names contain an *e* and are famous for food or beer.



```
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten]
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTige
r storage engine
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.towns.find({name: {$regex: /e/, $options: 'i'}, famous_for: {$in: ['food', 'beer']}}).pretty()
{
        "_id" : ObjectId("5b54e9e332dfc6a39322f567"),
        "name" : "New York",
        "population" : 22200000,
        "last_census" : ISODate("2009-07-31T00:00:00Z"),
        "famous_for" : [
                "statue of liberty",
                "food"
        ],
        "mayor" : {
                "name" : "Michael Bloomberg",
                "party" : "I"
        }
}
>
```

This problem’s solution is the same as the previous, except I had to add the **famous_for** criteria, and change the regular expression to search for only an e in the **name field**. The **famous_for** portion required an **$or** operation, but I chose instead to use the **$in** operator; it works exactly as it does in SQL, **$or**-ing over a list of values instead of each one separately. I also used the **.pretty()** command to make the output more readable.

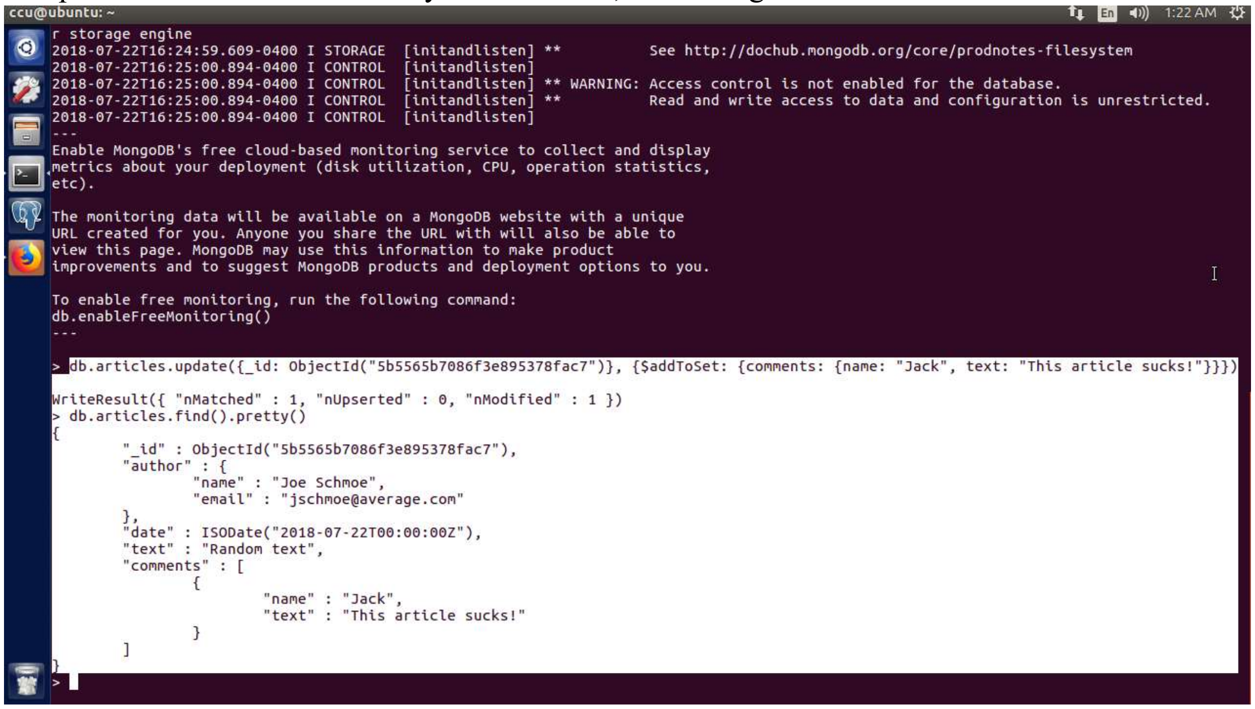
4. Create a new database named *blogger* with a collection named *articles*- insert a new article with an author name and email, creation date, and text.



```
ccu@ubuntu: ~                                                        11:07 PM
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017/blogger
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten]
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTige
r storage engine
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.articles.insert({author: {name: "Joe Schmoe", email: "jschmoe@average.com"}, date: ISODate("2018-07-22"),
...                     text: "Random text"})
WriteResult({ "nInserted" : 1 })
> db.articles.find().pretty()
{
        "_id" : ObjectId("5b554486b37539d65ef5cb1e"),
        "author" : {
                "name" : "Joe Schmoe",
                "email" : "jschmoe@average.com"
        },
        "date" : ISODate("2018-07-22T00:00:00Z"),
        "text" : "Random text"
}
> 
```

I used the **mongo blogger** command to create the database and the **.insert()** command to create the **articles** collection with an article inside of it. I then used **.find()** to make sure the article was there.

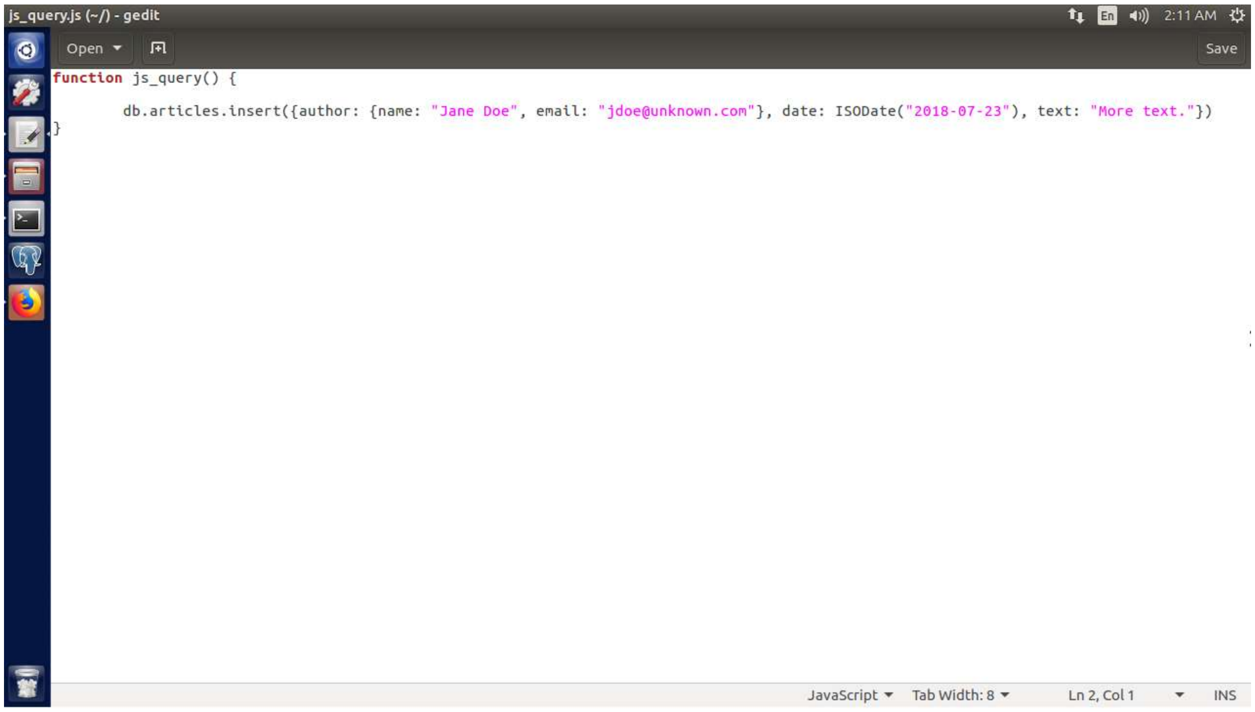
5. Update the article with an array of comments, containing a comment with an author and text.



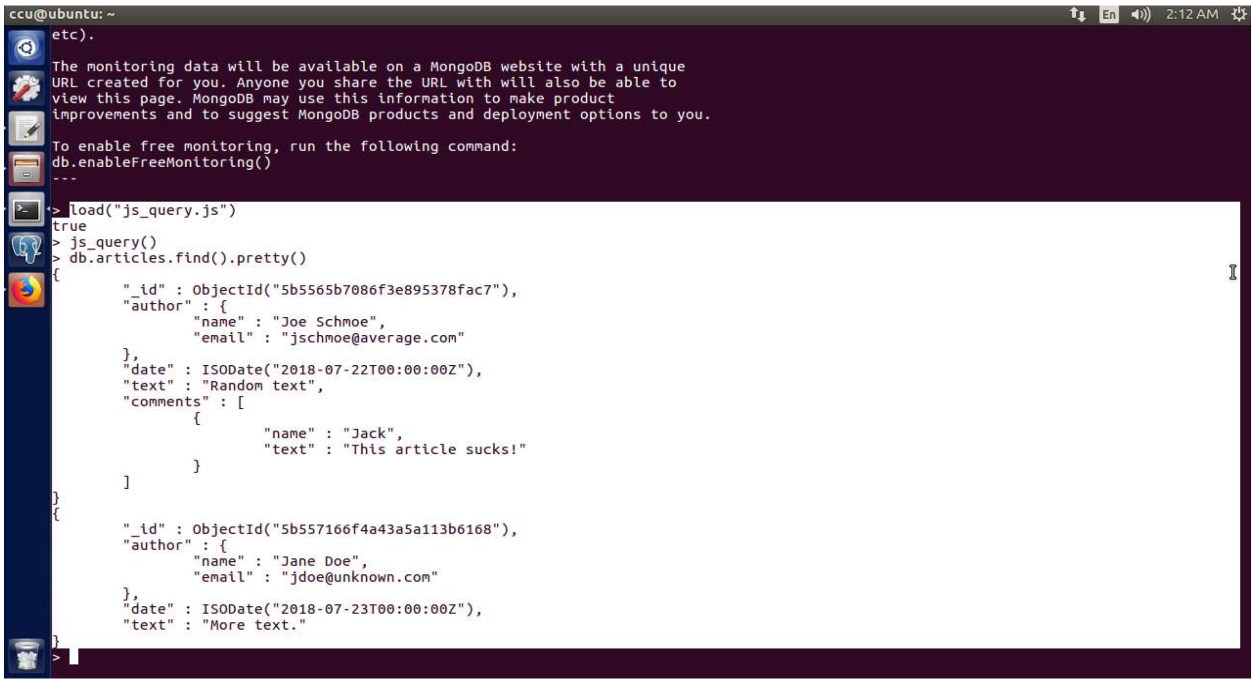
```
ccu@ubuntu: ~                                                              En  1:22 AM
r storage engine
2018-07-22T16:24:59.609-0400 I STORAGE  [initandlisten] **          See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.articles.update({_id: ObjectId("5b5565b7086f3e895378fac7")}, {$addToSet: {comments: {name: "Jack", text: "This article sucks!"}}})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.articles.find().pretty()
{
        "_id" : ObjectId("5b5565b7086f3e895378fac7"),
        "author" : {
                "name" : "Joe Schmoe",
                "email" : "jschmoe@average.com"
        },
        "date" : ISODate("2018-07-22T00:00:00Z"),
        "text" : "Random text",
        "comments" : [
                {
                        "name" : "Jack",
                        "text" : "This article sucks!"
                }
        ]
}
> 
```

For this problem, used the **.update()** command, but since the book specifically called for an array, I used **$addToSet** to add the comments instead of **$set**. I also tried **$push** to create the array, but if a singular value is used, like the above comment, it creates multiple copies of the value, so **$addToSet** seemed to be the best option.

6. Run a query from an external JavaScript file.



```
function js_query() {

        db.articles.insert({author: {name: "Jane Doe", email: "jdoe@unknown.com"}, date: ISODate("2018-07-23"), text: "More text."})
}
```



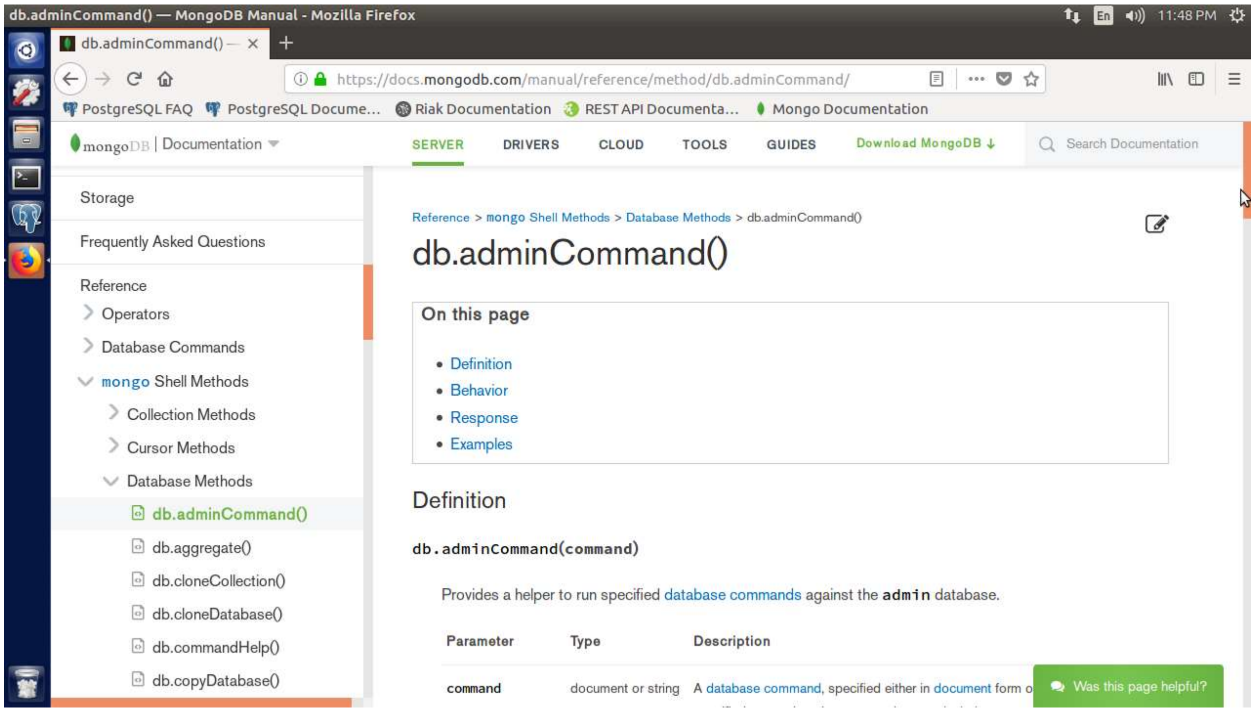
```
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> load("js_query.js")
true
> js_query()
> db.articles.find().pretty()
{
        "_id" : ObjectId("5b5565b7086f3e895378fac7"),
        "author" : {
                "name" : "Joe Schmoe",
                "email" : "jschmoe@average.com"
        },
        "date" : ISODate("2018-07-22T00:00:00Z"),
        "text" : "Random text",
        "comments" : [
                {
                        "name" : "Jack",
                        "text" : "This article sucks!"
                }
        ]
}
{
        "_id" : ObjectId("5b557166f4a43a5a113b6168"),
        "author" : {
                "name" : "Jane Doe",
                "email" : "jdoe@unknown.com"
        },
        "date" : ISODate("2018-07-23T00:00:00Z"),
        "text" : "More text."
}
> 
```

The book, as far as I can tell, did not mention running external files, so I had to look it up. In Mongo, you can use the **load()** function to point to a **.js** file that you want to use, so I created a file named **js_query** and wrote a simple function that adds a new article in the **articles** collection in the **blogger** database. I didn't realize this at first, but the **load()** function does not actually run the file, it loads it into Mongo, so that you can run the function. In this case, I called the function **js_query()**.

Day 2

Find:

1. A shortcut for admin commands.



Reference > mongo Shell Methods > Database Methods > db.adminCommand()

db.adminCommand()

On this page

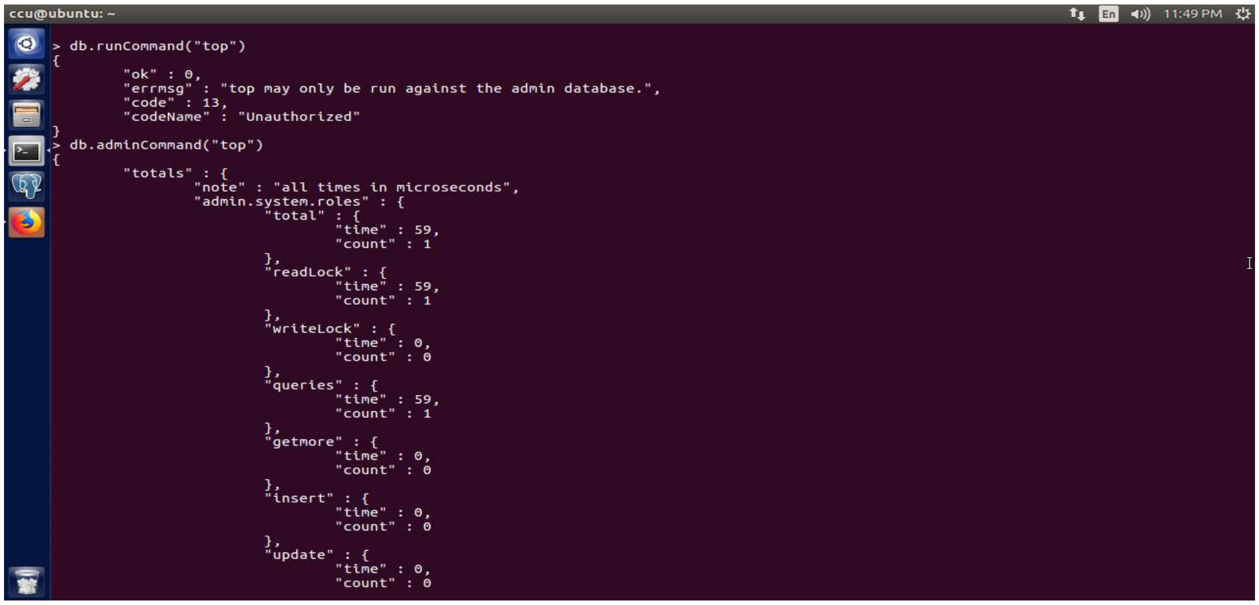
- Definition
- Behavior
- Response
- Examples

Definition

`db.adminCommand(command)`

Provides a helper to run specified database commands against the `admin` database.

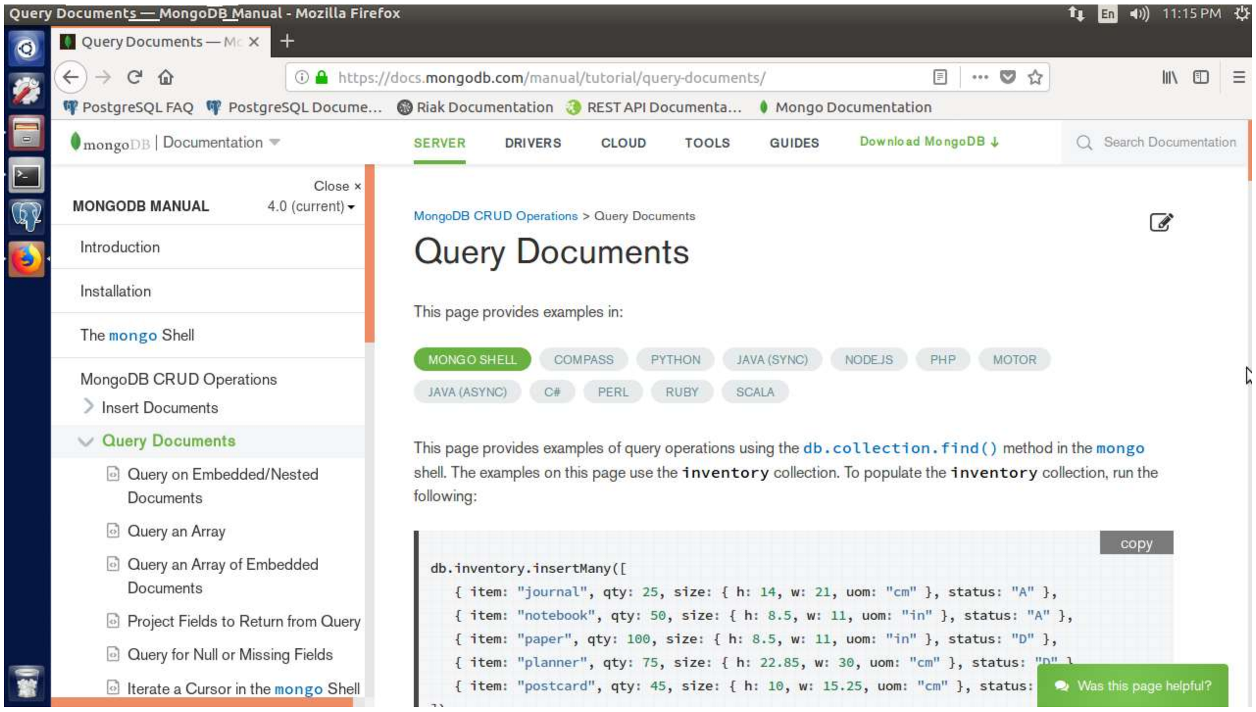
Parameter	Type	Description
command	document or string	A database command, specified either in document form o



```
> db.runCommand("top")
{
        "ok" : 0,
        "errmsg" : "top may only be run against the admin database.",
        "code" : 13,
        "codeName" : "Unauthorized"
}
> db.adminCommand("top")
{
        "totals" : {
                "note" : "all times in microseconds",
                "admin.system.roles" : {
                        "total" : {
                                "time" : 59,
                                "count" : 1
                        },
                        "readLock" : {
                                "time" : 59,
                                "count" : 1
                        },
                        "writeLock" : {
                                "time" : 0,
                                "count" : 0
                        },
                        "queries" : {
                                "time" : 59,
                                "count" : 1
                        },
                        "getmore" : {
                                "time" : 0,
                                "count" : 0
                        },
                        "insert" : {
                                "time" : 0,
                                "count" : 0
                        },
                        "update" : {
                                "time" : 0,
                                "count" : 0
```

The **db.adminCommand()** function allows using **admin** database functions without switching databases. It can be found in the MongoDB manual under Reference -> Database Methods -> db.adminCommand(). In the screenshot above, **db.runCommand("top")** gives an error stating that it can only be run against the **admin** database, but **db.adminCommand("top")** works without switching to **admin**.

2. The online documentation for queries and cursors.


Query Documents — MongoDB Manual - Mozilla Firefox

https://docs.mongodb.com/manual/tutorial/query-documents/

PostgreSQL FAQ · PostgreSQL Docume… · Riak Documentation · REST API Documenta… · Mongo Documentation

mongoDB | Documentation · SERVER · DRIVERS · CLOUD · TOOLS · GUIDES · Download MongoDB · Search Documentation

MONGODB MANUAL 4.0 (current)
- Introduction
- Installation
- The mongo Shell
- MongoDB CRUD Operations
  - Insert Documents
  - Query Documents
    - Query on Embedded/Nested Documents
    - Query an Array
    - Query an Array of Embedded Documents
    - Project Fields to Return from Query
    - Query for Null or Missing Fields
    - Iterate a Cursor in the mongo Shell

MongoDB CRUD Operations > Query Documents

Query Documents

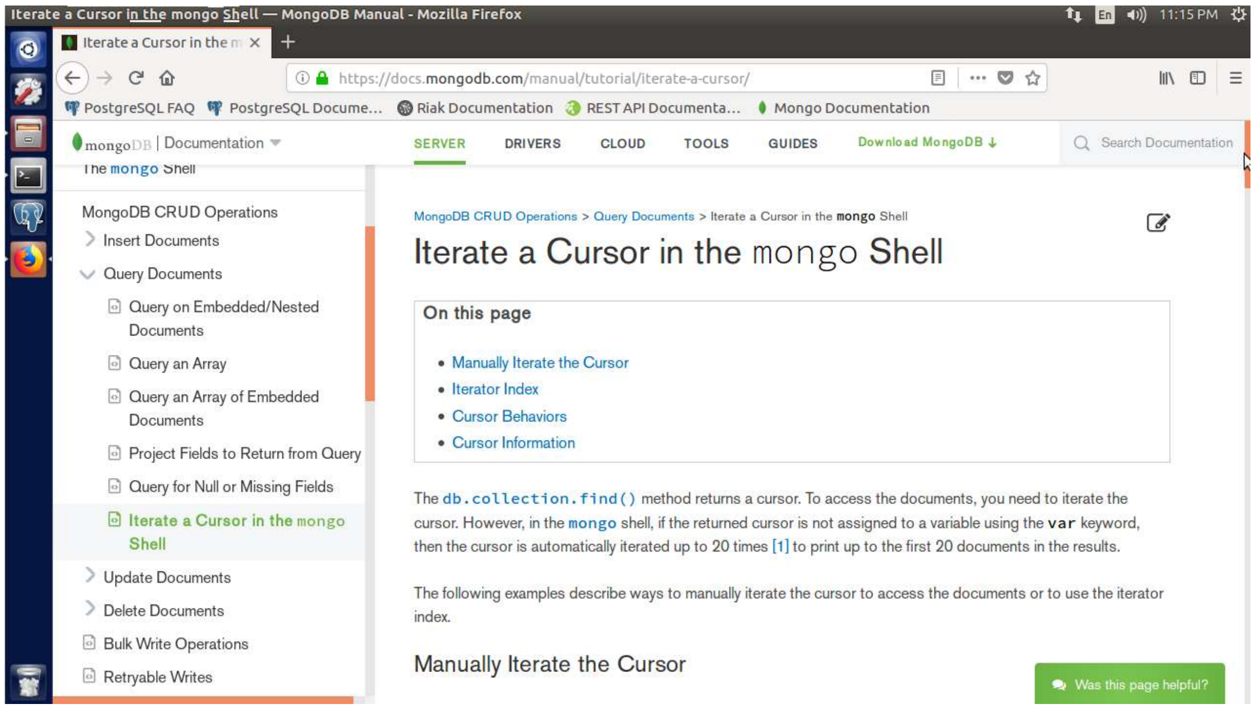
This page provides examples in:

MONGO SHELL · COMPASS · PYTHON · JAVA (SYNC) · NODE.JS · PHP · MOTOR · JAVA (ASYNC) · C# · PERL · RUBY · SCALA

This page provides examples of query operations using the `db.collection.find()` method in the `mongo` shell. The examples on this page use the `inventory` collection. To populate the `inventory` collection, run the following:

```
db.inventory.insertMany([
   { item: "journal", qty: 25, size: { h: 14, w: 21, uom: "cm" }, status: "A" },
   { item: "notebook", qty: 50, size: { h: 8.5, w: 11, uom: "in" }, status: "A" },
   { item: "paper", qty: 100, size: { h: 8.5, w: 11, uom: "in" }, status: "D" },
   { item: "planner", qty: 75, size: { h: 22.85, w: 30, uom: "cm" }, status: "D" },
   { item: "postcard", qty: 45, size: { h: 10, w: 15.25, uom: "cm" }, status:
```

Was this page helpful?


Iterate a Cursor in the mongo Shell — MongoDB Manual - Mozilla Firefox

https://docs.mongodb.com/manual/tutorial/iterate-a-cursor/

- The mongo Shell
- MongoDB CRUD Operations
  - Insert Documents
  - Query Documents
    - Query on Embedded/Nested Documents
    - Query an Array
    - Query an Array of Embedded Documents
    - Project Fields to Return from Query
    - Query for Null or Missing Fields
    - Iterate a Cursor in the mongo Shell
  - Update Documents
  - Delete Documents
  - Bulk Write Operations
  - Retryable Writes

MongoDB CRUD Operations > Query Documents > Iterate a Cursor in the mongo Shell

Iterate a Cursor in the mongo Shell

On this page
- Manually Iterate the Cursor
- Iterator Index
- Cursor Behaviors
- Cursor Information

The `db.collection.find()` method returns a cursor. To access the documents, you need to iterate the cursor. However, in the `mongo` shell, if the returned cursor is not assigned to a variable using the `var` keyword, then the cursor is automatically iterated up to 20 times [1] to print up to the first 20 documents in the results.

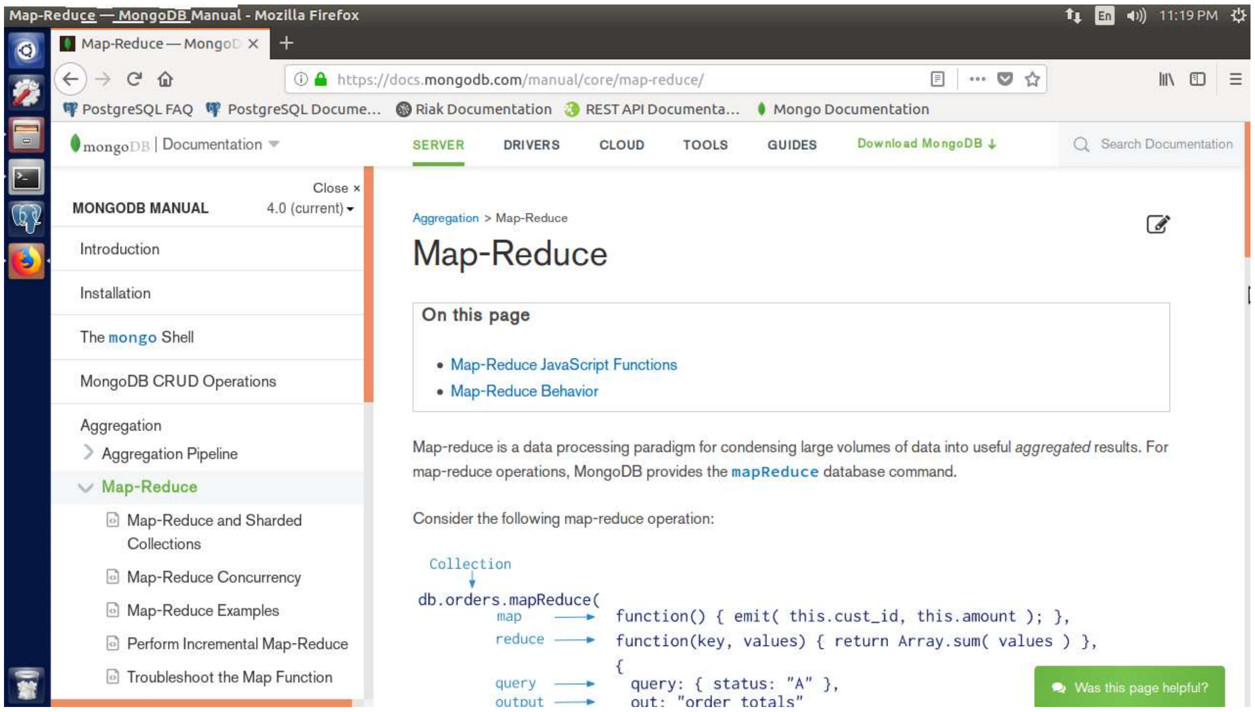
The following examples describe ways to manually iterate the cursor to access the documents or to use the iterator index.

Manually Iterate the Cursor

Was this page helpful?

The docs for queries and cursors can be found in the MongoDB manual, under Query Documents and Query Documents -> Iterate A Cursor In The mongo Shell, respectively.

3. The MongoDB documentation for mapreduce.



Map-Reduce

On this page

- Map-Reduce JavaScript Functions
- Map-Reduce Behavior

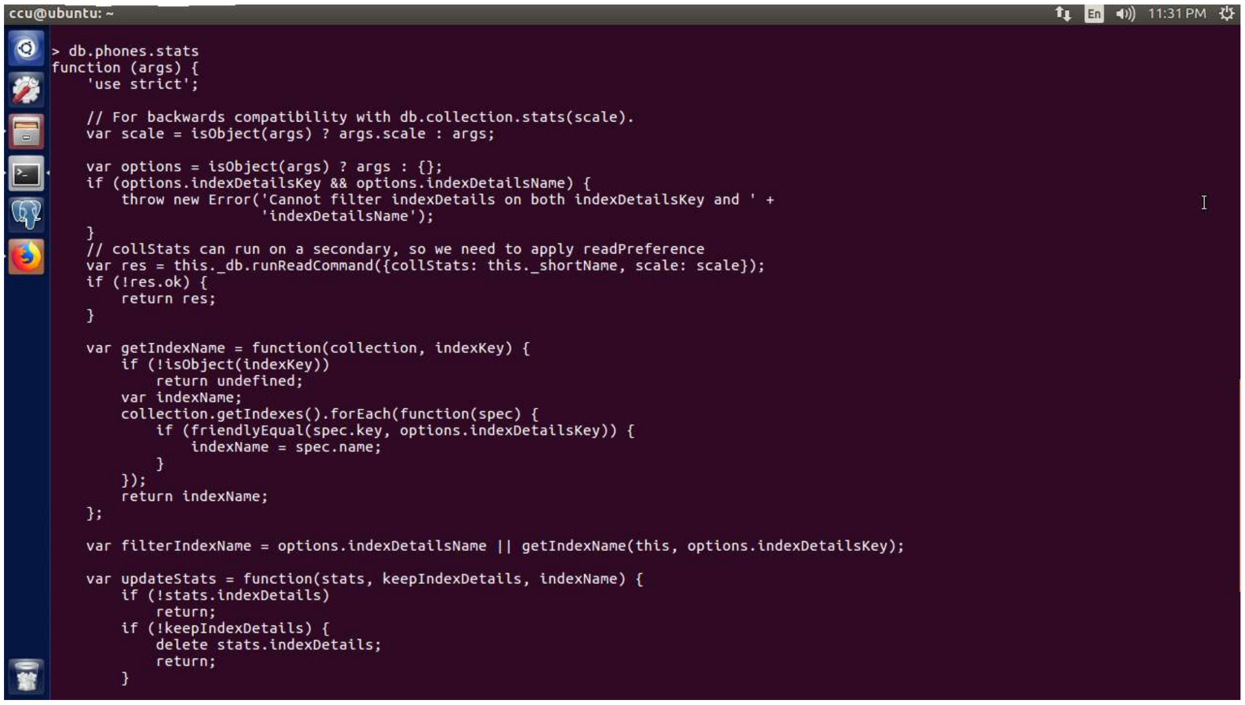
Map-reduce is a data processing paradigm for condensing large volumes of data into useful *aggregated* results. For map-reduce operations, MongoDB provides the `mapReduce` database command.

Consider the following map-reduce operation:

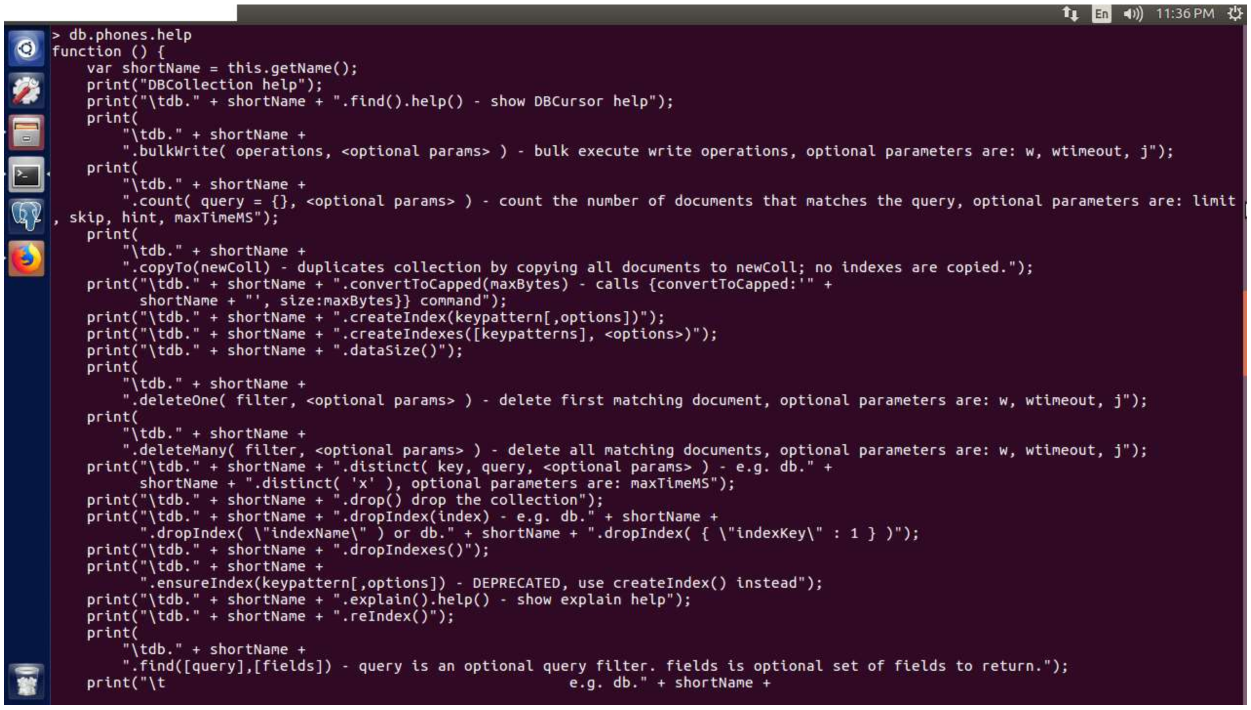
```
 Collection
db.orders.mapReduce(
        map    ——→  function() { emit( this.cust_id, this.amount ); },
        reduce ——→  function(key, values) { return Array.sum( values ) },
                    {
        query  ——→    query: { status: "A" },
        output ——→    out: "order_totals"
```

Documentation for **mapReduce()** is located in the MongoDB manual, under Aggregation -> Map-Reduce.

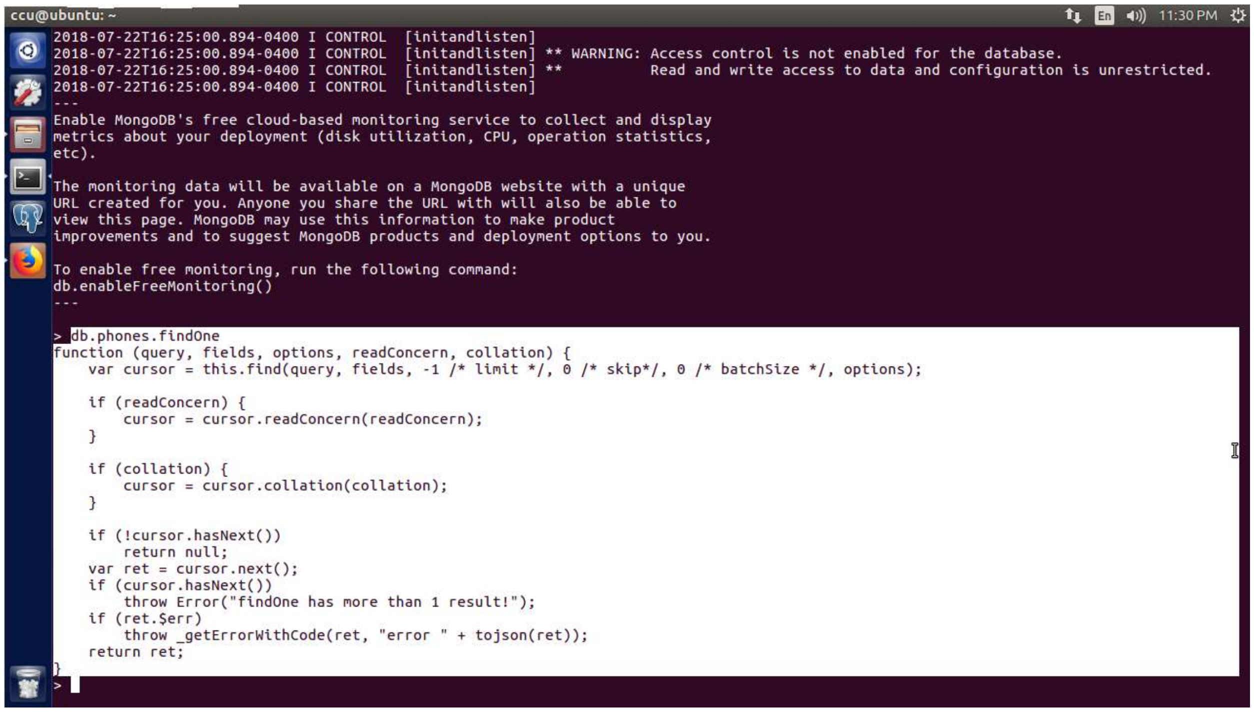
4. Through the JavaScript interface, investigate the code for three collections functions: help(), findOne(), and stats().



```
> db.phones.stats
function (args) {
    'use strict';

    // For backwards compatibility with db.collection.stats(scale).
    var scale = isObject(args) ? args.scale : args;

    var options = isObject(args) ? args : {};
    if (options.indexDetailsKey && options.indexDetailsName) {
        throw new Error('Cannot filter indexDetails on both indexDetailsKey and ' +
                        'indexDetailsName');
    }
    // collStats can run on a secondary, so we need to apply readPreference
    var res = this._db.runReadCommand({collStats: this._shortName, scale: scale});
    if (!res.ok) {
        return res;
    }

    var getIndexName = function(collection, indexKey) {
        if (!isObject(indexKey))
            return undefined;
        var indexName;
        collection.getIndexes().forEach(function(spec) {
            if (friendlyEqual(spec.key, options.indexDetailsKey)) {
                indexName = spec.name;
            }
        });
        return indexName;
    };

    var filterIndexName = options.indexDetailsName || getIndexName(this, options.indexDetailsKey);

    var updateStats = function(stats, keepIndexDetails, indexName) {
        if (!stats.indexDetails)
            return;
        if (!keepIndexDetails) {
            delete stats.indexDetails;
            return;
        }
```



```
> db.phones.help
function () {
    var shortName = this.getName();
    print("DBCollection help");
    print("\tdb." + shortName + ".find().help() - show DBCursor help");
    print(
        "\tdb." + shortName +
        ".bulkWrite( operations, <optional params> ) - bulk execute write operations, optional parameters are: w, wtimeout, j");
    print(
        "\tdb." + shortName +
        ".count( query = {}, <optional params> ) - count the number of documents that matches the query, optional parameters are: limit
, skip, hint, maxTimeMS");
    print(
        "\tdb." + shortName +
        ".copyTo(newColl) - duplicates collection by copying all documents to newColl; no indexes are copied.");
    print("\tdb." + shortName + ".convertToCapped(maxBytes) - calls {convertToCapped:'" +
          shortName + "', size:maxBytes}} command");
    print("\tdb." + shortName + ".createIndex(keypattern[,options])");
    print("\tdb." + shortName + ".createIndexes([keypatterns], <options>)");
    print("\tdb." + shortName + ".dataSize()");
    print(
        "\tdb." + shortName +
        ".deleteOne( filter, <optional params> ) - delete first matching document, optional parameters are: w, wtimeout, j");
    print(
        "\tdb." + shortName +
        ".deleteMany( filter, <optional params> ) - delete all matching documents, optional parameters are: w, wtimeout, j");
    print("\tdb." + shortName + ".distinct( key, query, <optional params> ) - e.g. db." +
          shortName + ".distinct( 'x' ), optional parameters are: maxTimeMS");
    print("\tdb." + shortName + ".drop() drop the collection");
    print("\tdb." + shortName + ".dropIndex(index) - e.g. db." + shortName +
          ".dropIndex( \"indexName\" ) or db." + shortName + ".dropIndex( { \"indexKey\" : 1 } )");
    print("\tdb." + shortName + ".dropIndexes()");
    print("\tdb." + shortName +
          ".ensureIndex(keypattern[,options]) - DEPRECATED, use createIndex() instead");
    print("\tdb." + shortName + ".explain().help() - show explain help");
    print("\tdb." + shortName + ".reIndex()");
    print(
        "\tdb." + shortName +
        ".find([query],[fields]) - query is an optional query filter. fields is optional set of fields to return.");
    print("\t                                              e.g. db." + shortName +
```

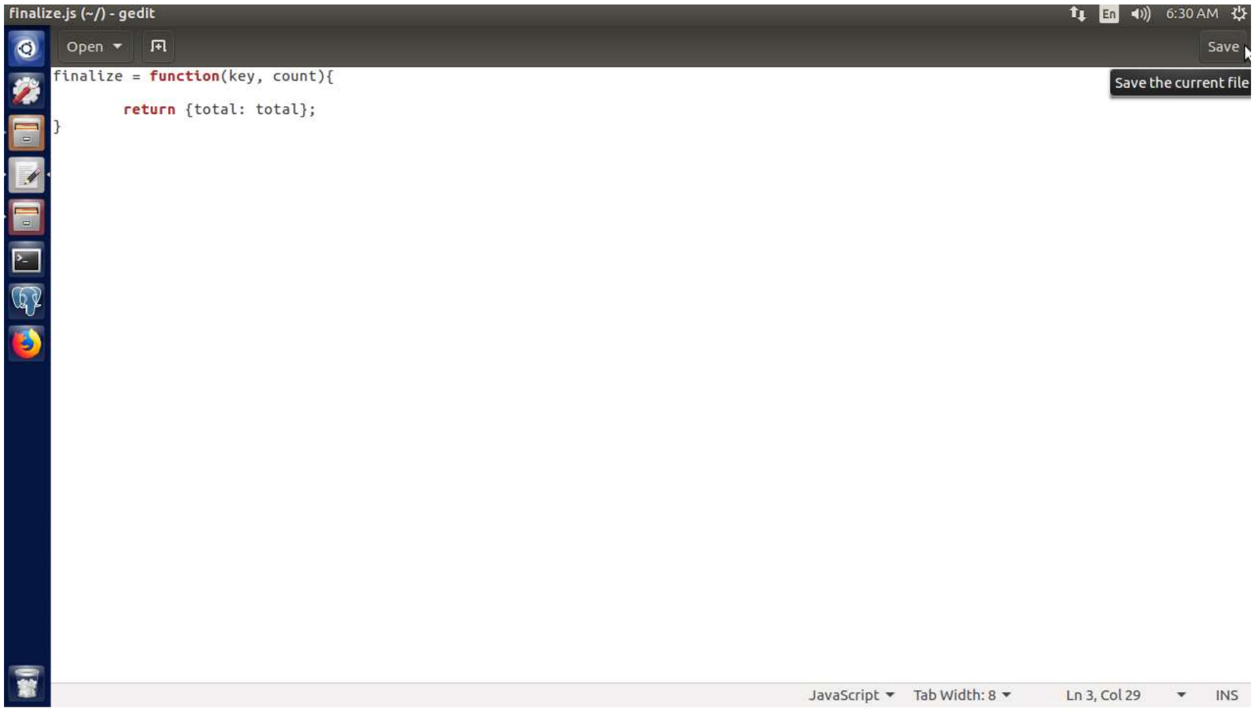


```
ccu@ubuntu: ~
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten] **          Read and write access to data and configuration is unrestricted.
2018-07-22T16:25:00.894-0400 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service to collect and display
metrics about your deployment (disk utilization, CPU, operation statistics,
etc).

The monitoring data will be available on a MongoDB website with a unique
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> db.phones.findOne
function (query, fields, options, readConcern, collation) {
    var cursor = this.find(query, fields, -1 /* limit */, 0 /* skip*/, 0 /* batchSize */, options);

    if (readConcern) {
        cursor = cursor.readConcern(readConcern);
    }

    if (collation) {
        cursor = cursor.collation(collation);
    }

    if (!cursor.hasNext())
        return null;
    var ret = cursor.next();
    if (cursor.hasNext())
        throw Error("findOne has more than 1 result!");
    if (ret.$err)
        throw _getErrorWithCode(ret, "error " + tojson(ret));
    return ret;
}
>
```

To investigate collection functions, all you have to do is use the function without the calling parentheses. So, I chose **phones** as the collection and called **db.phones.help**, **db.phones.findOne**, and **db.phones.stats**, and was shown how the functions are operating.

Do:

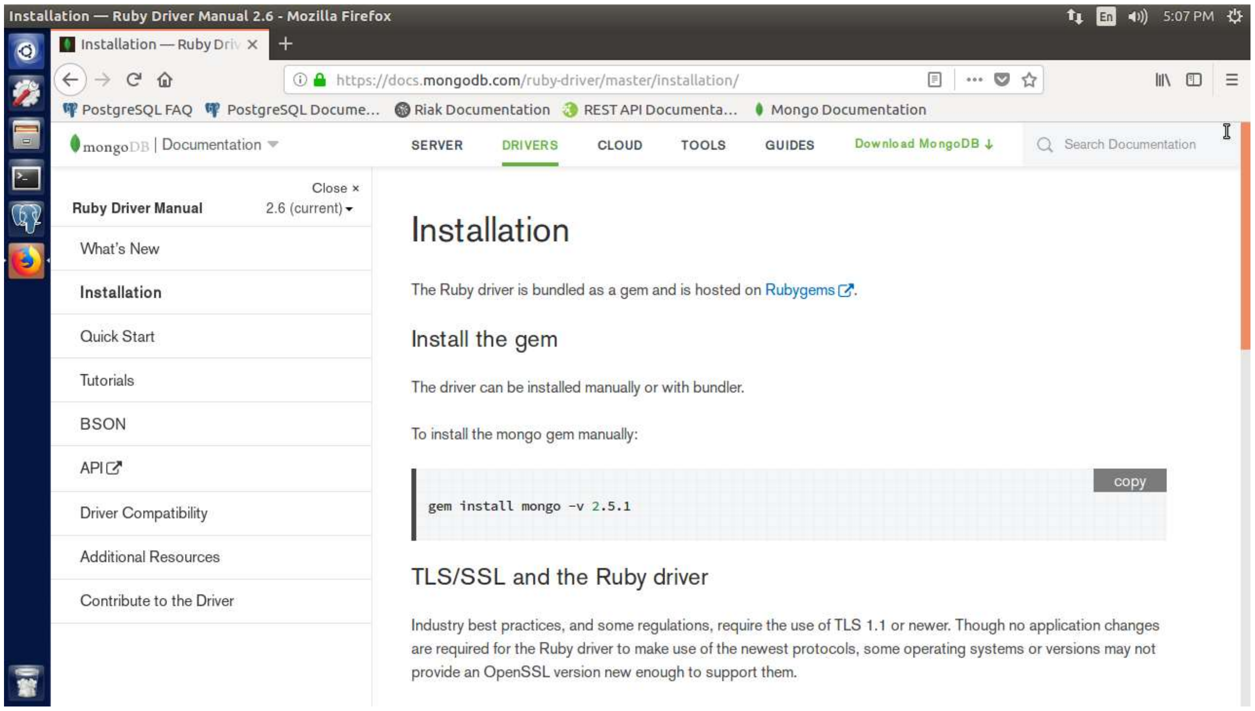
1. Implement a finalize method to output the count as the total.



```
finalize = function(key, count){

        return {total: total};
}
```

Using this function, I was able to get the output to show **total: count:120**, etc., but I couldn't quite figure out how to get the word **count** to go away.

2. Install a Mongo driver for a language of your choice, and connect to the database. Populate a collection through it, and index one of the fields.



Installation

The Ruby driver is bundled as a gem and is hosted on Rubygems.

Install the gem

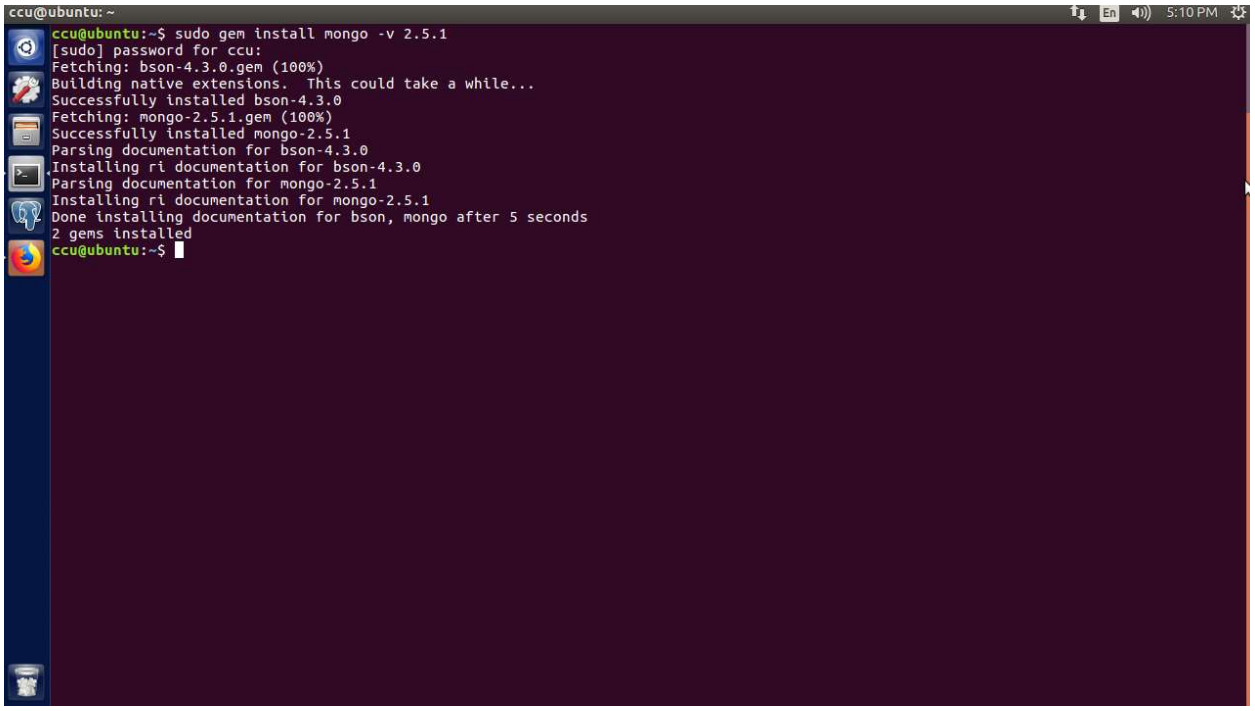
The driver can be installed manually or with bundler.

To install the mongo gem manually:

```
gem install mongo -v 2.5.1
```

TLS/SSL and the Ruby driver

Industry best practices, and some regulations, require the use of TLS 1.1 or newer. Though no application changes are required for the Ruby driver to make use of the newest protocols, some operating systems or versions may not provide an OpenSSL version new enough to support them.

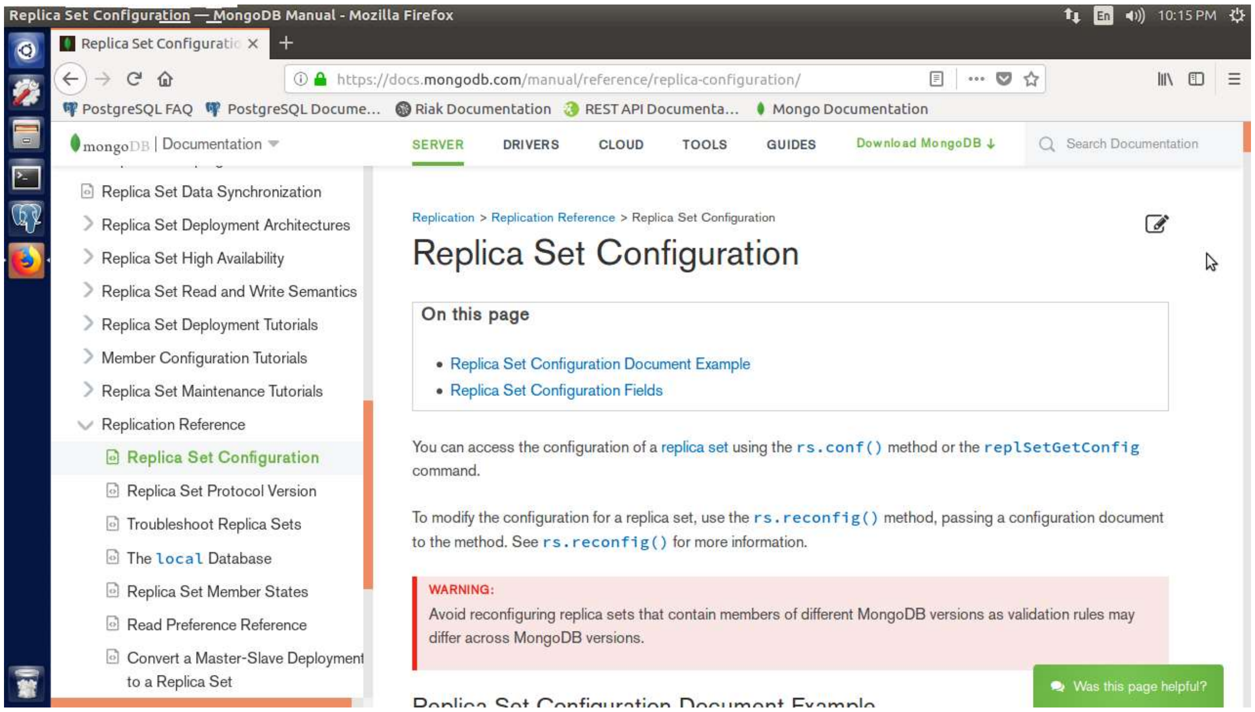


```
ccu@ubuntu:~$ sudo gem install mongo -v 2.5.1
[sudo] password for ccu:
Fetching: bson-4.3.0.gem (100%)
Building native extensions.  This could take a while...
Successfully installed bson-4.3.0
Fetching: mongo-2.5.1.gem (100%)
Successfully installed mongo-2.5.1
Parsing documentation for bson-4.3.0
Installing ri documentation for bson-4.3.0
Parsing documentation for mongo-2.5.1
Installing ri documentation for mongo-2.5.1
Done installing documentation for bson, mongo after 5 seconds
2 gems installed
ccu@ubuntu:~$
```

Day 3

Find:

1. Read the full replica set configuration options in the online docs.



Replica Set Configuration — MongoDB Manual - Mozilla Firefox

https://docs.mongodb.com/manual/reference/replica-configuration/

Replication > Replication Reference > Replica Set Configuration

Replica Set Configuration

On this page

- Replica Set Configuration Document Example
- Replica Set Configuration Fields

You can access the configuration of a replica set using the `rs.conf()` method or the `replSetGetConfig` command.

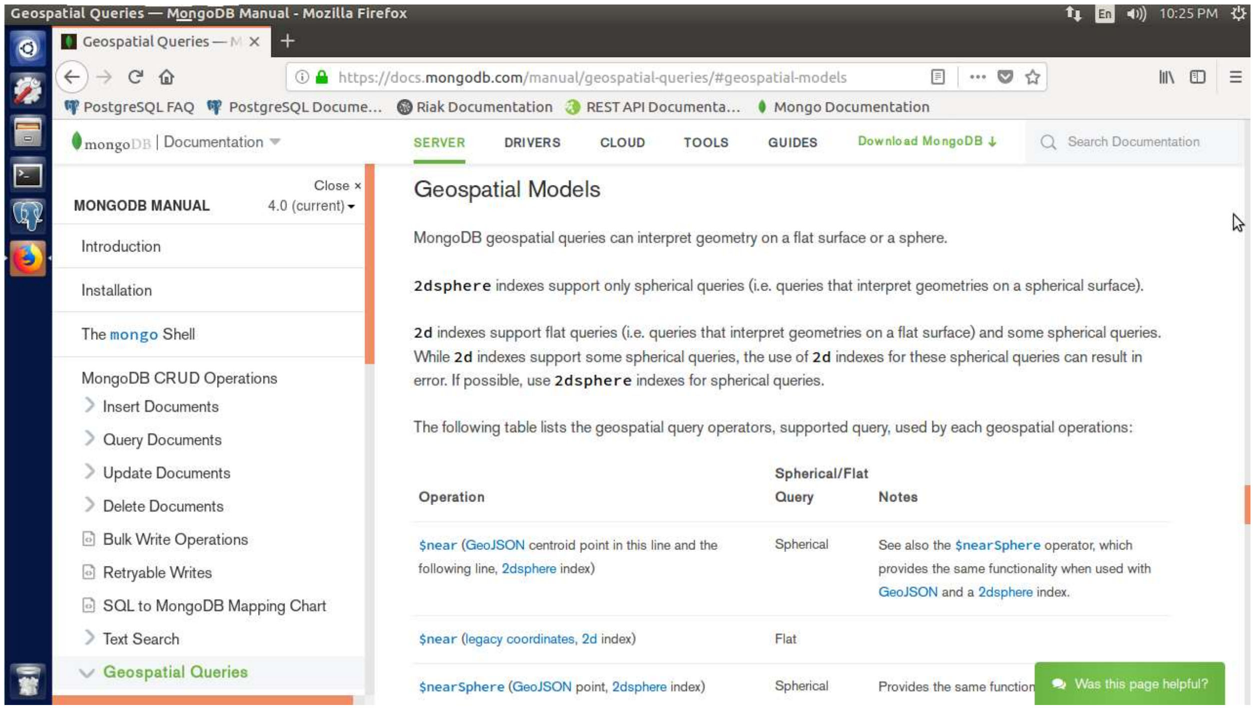
To modify the configuration for a replica set, use the `rs.reconfig()` method, passing a configuration document to the method. See `rs.reconfig()` for more information.

WARNING:
Avoid reconfiguring replica sets that contain members of different MongoDB versions as validation rules may differ across MongoDB versions.

PostgreSQL FAQ
PostgreSQL Docume…
Riak Documentation
REST API Documenta…
Mongo Documentation
mongoDB | Documentation
SERVER
DRIVERS
CLOUD
TOOLS
GUIDES
Download MongoDB
Search Documentation
Replica Set Data Synchronization
Replica Set Deployment Architectures
Replica Set High Availability
Replica Set Read and Write Semantics
Replica Set Deployment Tutorials
Member Configuration Tutorials
Replica Set Maintenance Tutorials
Replication Reference
Replica Set Configuration
Replica Set Protocol Version
Troubleshoot Replica Sets
The local Database
Replica Set Member States
Read Preference Reference
Convert a Master-Slave Deployment to a Replica Set
Was this page helpful?
10:15 PM

Documentation for replica sets can be found in the MongoDB manual under Replication Reference -> Replica Set Configuration.

2. Find out how to create a spherical geo index.



Geospatial Models

MongoDB geospatial queries can interpret geometry on a flat surface or a sphere.

`2dsphere` indexes support only spherical queries (i.e. queries that interpret geometries on a spherical surface).

`2d` indexes support flat queries (i.e. queries that interpret geometries on a flat surface) and some spherical queries. While `2d` indexes support some spherical queries, the use of `2d` indexes for these spherical queries can result in error. If possible, use `2dsphere` indexes for spherical queries.

The following table lists the geospatial query operators, supported query, used by each geospatial operations:

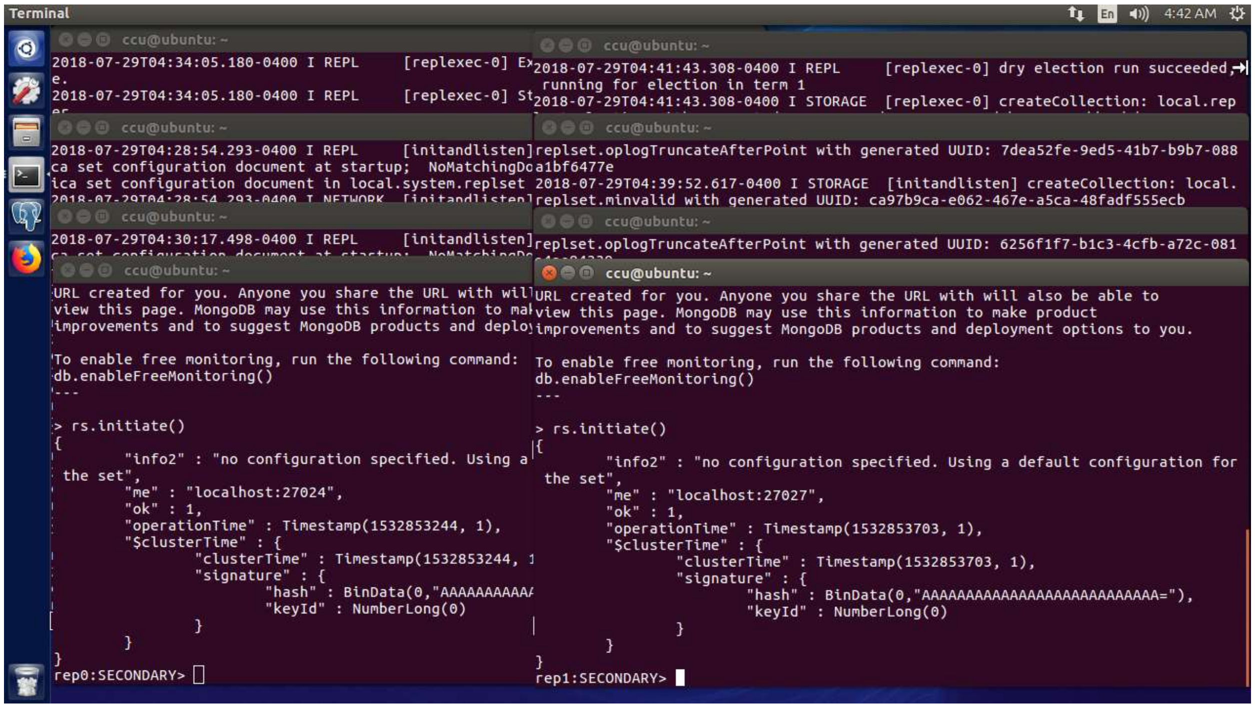
Operation	Spherical/Flat Query	Notes
`$near` (GeoJSON centroid point in this line and the following line, 2dsphere index)	Spherical	See also the `$nearSphere` operator, which provides the same functionality when used with GeoJSON and a 2dsphere index.
`$near` (legacy coordinates, 2d index)	Flat	
`$nearSphere` (GeoJSON point, 2dsphere index)	Spherical	Provides the same function

All that is needed to make a spherical geo index is to use a **2dsphere** index instead of a **2d** index.

Do:

1. Mongo has support for bounding shapes (namely, squares and circles). Find all cities within a 50-mile box around the center of London.

2. Run six servers: three servers in a replica set, and each replica set is one of two shards. Run a config server and mongos. Run GridFS across them (this is the final exam).



```
ccu@ubuntu: ~
2018-07-29T04:34:05.180-0400 I REPL     [replexec-0] Ex
e.
2018-07-29T04:34:05.180-0400 I REPL     [replexec-0] St
```

```
ccu@ubuntu: ~
2018-07-29T04:41:43.308-0400 I REPL     [replexec-0] dry election run succeeded,
 running for election in term 1
2018-07-29T04:41:43.308-0400 I STORAGE  [replexec-0] createCollection: local.rep
```

```
ccu@ubuntu: ~
2018-07-29T04:28:54.293-0400 I REPL     [initandlisten]
ca set configuration document at startup;  NoMatchingDo
ica set configuration document in local.system.replset
```

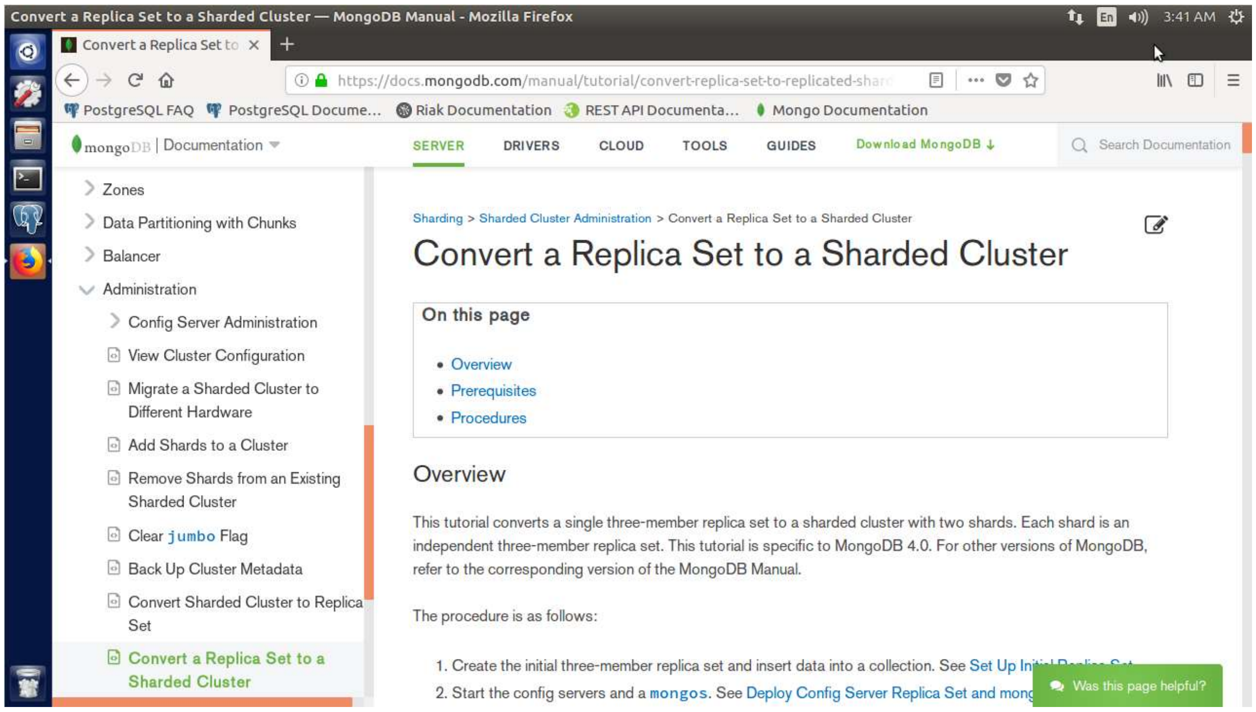
```
ccu@ubuntu: ~
replset.oplogTruncateAfterPoint with generated UUID: 7dea52fe-9ed5-41b7-b9b7-088
a1bf6477e
2018-07-29T04:39:52.617-0400 I STORAGE  [initandlisten] createCollection: local.
replset.minvalid with generated UUID: ca97b9ca-e062-467e-a5ca-48fadf555ecb
```

```
ccu@ubuntu: ~
2018-07-29T04:30:17.498-0400 I REPL     [initandlisten]
```

```
ccu@ubuntu: ~
replset.oplogTruncateAfterPoint with generated UUID: 6256f1f7-b1c3-4cfb-a72c-081
```

```
ccu@ubuntu: ~
URL created for you. Anyone you share the URL with wil
view this page. MongoDB may use this information to ma
improvements and to suggest MongoDB products and deplo

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> rs.initiate()
{
        "info2" : "no configuration specified. Using a
 the set",
        "me" : "localhost:27024",
        "ok" : 1,
        "operationTime" : Timestamp(1532853244, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532853244, 1
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAA
                        "keyId" : NumberLong(0)
                }
        }
}
rep0:SECONDARY>
```

```
ccu@ubuntu: ~
URL created for you. Anyone you share the URL with will also be able to
view this page. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command:
db.enableFreeMonitoring()
---

> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for
 the set",
        "me" : "localhost:27027",
        "ok" : 1,
        "operationTime" : Timestamp(1532853703, 1),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532853703, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
rep1:SECONDARY>
```


Convert a Replica Set to a Sharded Cluster — MongoDB Manual - Mozilla Firefox
En
3:41 AM
Convert a Replica Set to
https://docs.mongodb.com/manual/tutorial/convert-replica-set-to-replicated-shar
PostgreSQL FAQ
PostgreSQL Docume...
Riak Documentation
REST API Documenta...
Mongo Documentation
mongoDB | Documentation
SERVER
DRIVERS
CLOUD
TOOLS
GUIDES
Download MongoDB
Search Documentation
Zones
Data Partitioning with Chunks
Balancer
Administration
Config Server Administration
View Cluster Configuration
Migrate a Sharded Cluster to Different Hardware
Add Shards to a Cluster
Remove Shards from an Existing Sharded Cluster
Clear jumbo Flag
Back Up Cluster Metadata
Convert Sharded Cluster to Replica Set
Convert a Replica Set to a Sharded Cluster
Was this page helpful?

Sharding > Sharded Cluster Administration > Convert a Replica Set to a Sharded Cluster

Convert a Replica Set to a Sharded Cluster

On this page

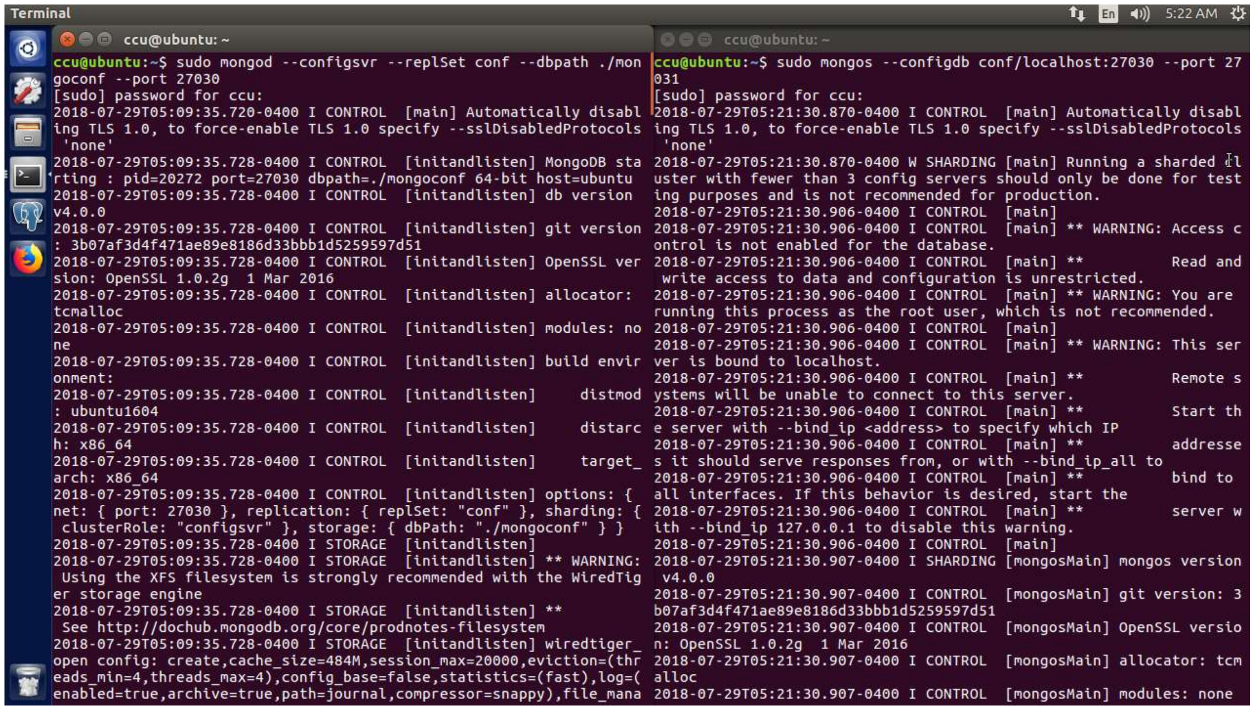
- Overview
- Prerequisites
- Procedures

Overview

This tutorial converts a single three-member replica set to a sharded cluster with two shards. Each shard is an independent three-member replica set. This tutorial is specific to MongoDB 4.0. For other versions of MongoDB, refer to the corresponding version of the MongoDB Manual.

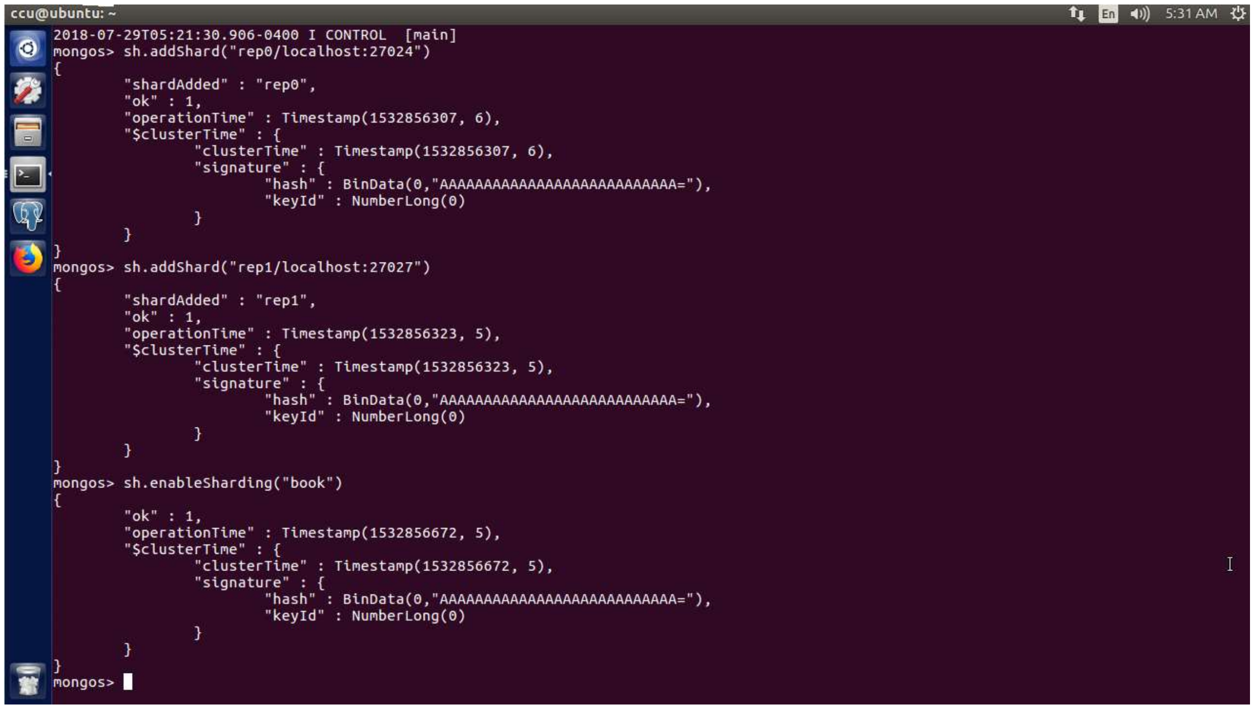
The procedure is as follows:

1. Create the initial three-member replica set and insert data into a collection. See Set Up Initial Replica Set
2. Start the config servers and a `mongos`. See Deploy Config Server Replica Set and mong

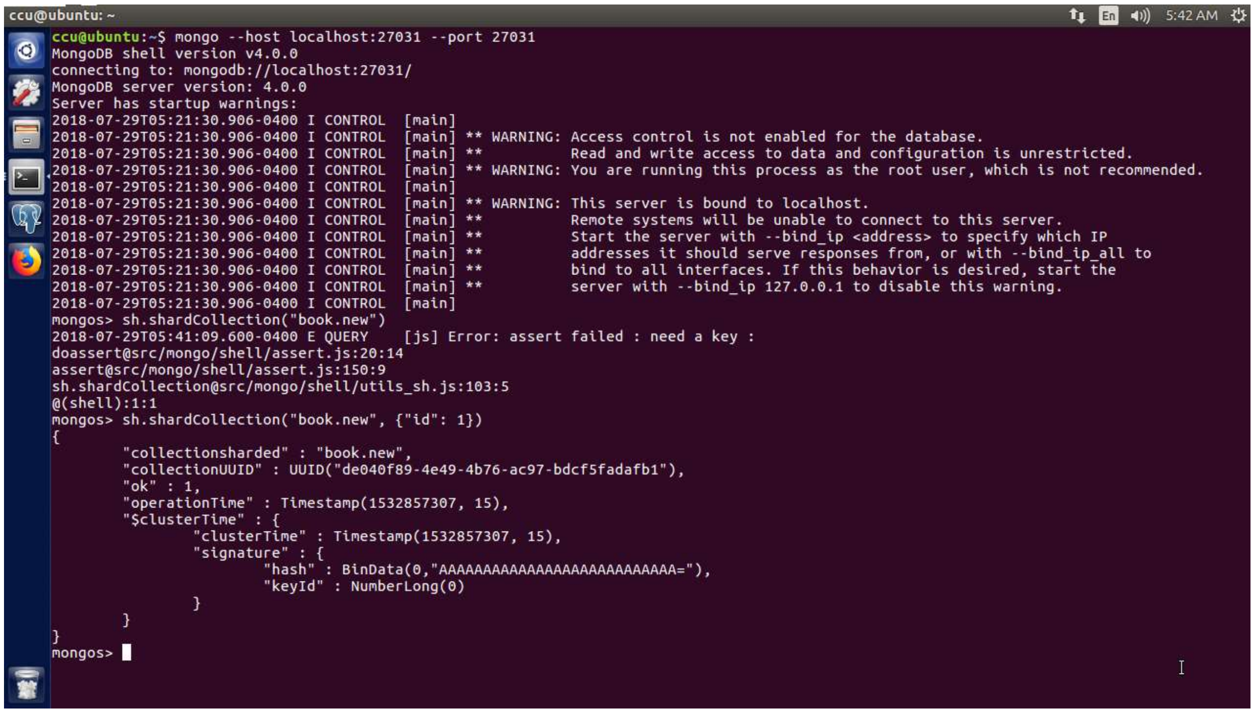


```
ccu@ubuntu:~$ sudo mongod --configsvr --replSet conf --dbpath ./mon
goconf --port 27030
[sudo] password for ccu:
2018-07-29T05:09:35.720-0400 I CONTROL  [main] Automatically disabl
ing TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols
 'none'
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] MongoDB sta
rting : pid=20272 port=27030 dbpath=./mongoconf 64-bit host=ubuntu
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] db version
v4.0.0
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] git version
: 3b07af3d4f471ae89e8186d33bbb1d5259597d51
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] OpenSSL ver
sion: OpenSSL 1.0.2g  1 Mar 2016
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] allocator:
tcmalloc
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] modules: no
ne
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] build envir
onment:
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten]     distmod
: ubuntu1604
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten]     distarc
h: x86_64
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten]     target_
arch: x86_64
2018-07-29T05:09:35.728-0400 I CONTROL  [initandlisten] options: {
net: { port: 27030 }, replication: { replSet: "conf" }, sharding: {
 clusterRole: "configsvr" }, storage: { dbPath: "./mongoconf" } }
2018-07-29T05:09:35.728-0400 I STORAGE  [initandlisten]
2018-07-29T05:09:35.728-0400 I STORAGE  [initandlisten] ** WARNING:
 Using the XFS filesystem is strongly recommended with the WiredTig
er storage engine
2018-07-29T05:09:35.728-0400 I STORAGE  [initandlisten] **
 See http://dochub.mongodb.org/core/prodnotes-filesystem
2018-07-29T05:09:35.728-0400 I STORAGE  [initandlisten] wiredtiger_
open config: create,cache_size=484M,session_max=20000,eviction=(thr
eads_min=4,threads_max=4),config_base=false,statistics=(fast),log=(
enabled=true,archive=true,path=journal,compressor=snappy),file_mana
```

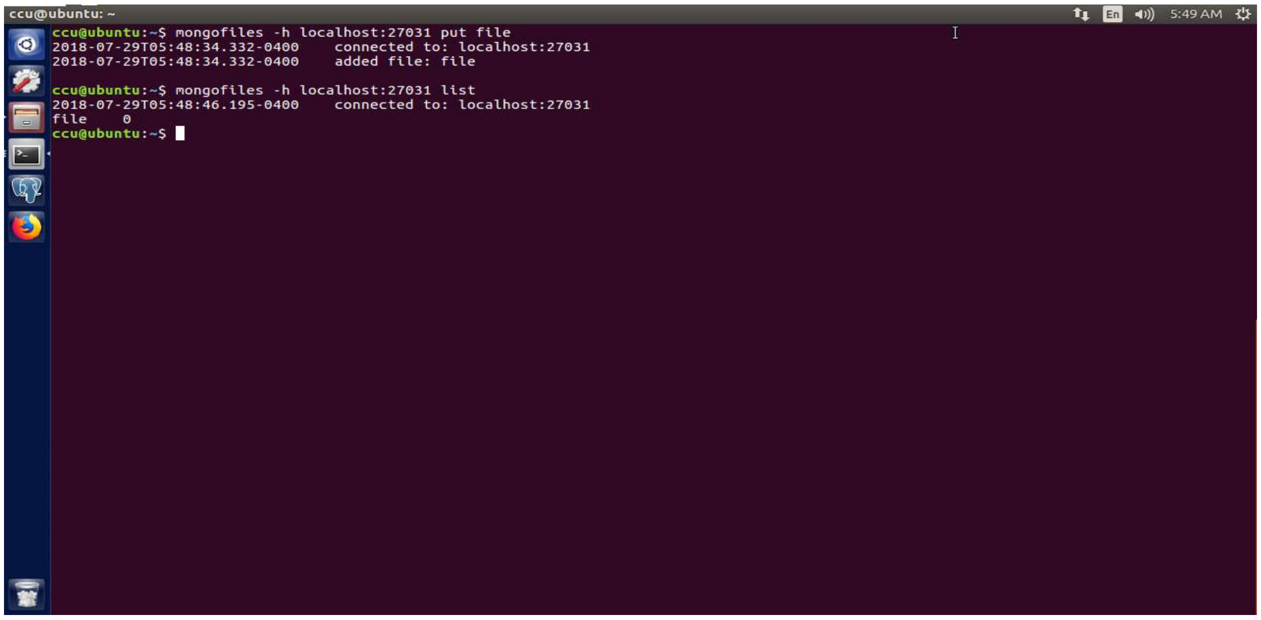
```
ccu@ubuntu:~$ sudo mongos --configdb conf/localhost:27030 --port 27
031
[sudo] password for ccu:
2018-07-29T05:21:30.870-0400 I CONTROL  [main] Automatically disabl
ing TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols
 'none'
2018-07-29T05:21:30.870-0400 W SHARDING [main] Running a sharded cl
uster with fewer than 3 config servers should only be done for test
ing purposes and is not recommended for production.
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: Access c
ontrol is not enabled for the database.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Read and
 write access to data and configuration is unrestricted.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: You are
running this process as the root user, which is not recommended.
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: This ser
ver is bound to localhost.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Remote s
ystems will be unable to connect to this server.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Start th
e server with --bind_ip <address> to specify which IP
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          addresse
s it should serve responses from, or with --bind_ip_all to
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          bind to
all interfaces. If this behavior is desired, start the
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          server w
ith --bind_ip 127.0.0.1 to disable this warning.
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
2018-07-29T05:21:30.907-0400 I SHARDING [mongosMain] mongos version
 v4.0.0
2018-07-29T05:21:30.907-0400 I CONTROL  [mongosMain] git version: 3
b07af3d4f471ae89e8186d33bbb1d5259597d51
2018-07-29T05:21:30.907-0400 I CONTROL  [mongosMain] OpenSSL versio
n: OpenSSL 1.0.2g  1 Mar 2016
2018-07-29T05:21:30.907-0400 I CONTROL  [mongosMain] allocator: tcm
alloc
2018-07-29T05:21:30.907-0400 I CONTROL  [mongosMain] modules: none
```



```
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
mongos> sh.addShard("rep0/localhost:27024")
{
        "shardAdded" : "rep0",
        "ok" : 1,
        "operationTime" : Timestamp(1532856307, 6),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532856307, 6),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.addShard("rep1/localhost:27027")
{
        "shardAdded" : "rep1",
        "ok" : 1,
        "operationTime" : Timestamp(1532856323, 5),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532856323, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> sh.enableSharding("book")
{
        "ok" : 1,
        "operationTime" : Timestamp(1532856672, 5),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532856672, 5),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos>
```



```
ccu@ubuntu: ~
ccu@ubuntu:~$ mongo --host localhost:27031 --port 27031
MongoDB shell version v4.0.0
connecting to: mongodb://localhost:27031/
MongoDB server version: 4.0.0
Server has startup warnings:
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: Access control is not enabled for the database.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Read and write access to data and configuration is unrestricted.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: You are running this process as the root user, which is not recommended.
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
2018-07-29T05:21:30.906-0400 I CONTROL  [main] ** WARNING: This server is bound to localhost.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Remote systems will be unable to connect to this server.
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          Start the server with --bind_ip <address> to specify which IP
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          addresses it should serve responses from, or with --bind_ip_all to
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          bind to all interfaces. If this behavior is desired, start the
2018-07-29T05:21:30.906-0400 I CONTROL  [main] **          server with --bind_ip 127.0.0.1 to disable this warning.
2018-07-29T05:21:30.906-0400 I CONTROL  [main]
mongos> sh.shardCollection("book.new")
2018-07-29T05:41:09.600-0400 E QUERY    [js] Error: assert failed : need a key :
doassert@src/mongo/shell/assert.js:20:14
assert@src/mongo/shell/assert.js:150:9
sh.shardCollection@src/mongo/shell/utils_sh.js:103:5
@(shell):1:1
mongos> sh.shardCollection("book.new", {"id": 1})
{
        "collectionsharded" : "book.new",
        "collectionUUID" : UUID("de040f89-4e49-4b76-ac97-bdcf5fadafb1"),
        "ok" : 1,
        "operationTime" : Timestamp(1532857307, 15),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1532857307, 15),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
mongos> 
```



```
ccu@ubuntu: ~
ccu@ubuntu:~$ mongofiles -h localhost:27031 put file
2018-07-29T05:48:34.332-0400    connected to: localhost:27031
2018-07-29T05:48:34.332-0400    added file: file

ccu@ubuntu:~$ mongofiles -h localhost:27031 list
2018-07-29T05:48:46.195-0400    connected to: localhost:27031
file    0
ccu@ubuntu:~$ 
```

This task, while seemingly daunting turned out to be easy, if not time consuming. First, I created and initialized two sets of three replica set servers. I then found out that you can use the --s**hardsvr** option when creating replica sets, but I had already created both sets of servers. Fortunately, I found a guide in the MongoDB documentation on how to convert an already running server to a sharded server. All you have to do is stop a server and restart it with the –**shardsvr** option included. After that was fixed, I created both the config server replica set and the mongos server. Next, I added the two replica set shards to the mongos and enabled sharding for the **book** database and the **book.new** collection. Finally, to test GridFS, I created a blank text file and used **mongofiles** to upload it to the mongos server.