MongoDB

Updated: Apr 06, 2022 | Published: Sep 30, 2020

MONGODB

UNIVERSITY

Rate this article

First steps in the MongoDB World? This cheat sheet is filled with some handy tips, commands, and quick references to get you connected and CRUD'ing in no time!

- Get a free MongoDB cluster in MongoDB Atlas.
- Follow a course in MongoDB University.

Table of Contents

- Sharded Cluster
- Wrap-up

Connect MongoDB Shell

```
mongo # connects to mongodb://127.0.0.1:27017 by default
mongo --host <host> --port <port> -u <user> -p <pwd> # omit the password if you
mongo "mongodb://192.168.1.1:27017"
mongo "mongodb+srv://cluster-name.abcde.mongodb.net/<dbname>" --username <usern
```

- More documentation about the MongoDB Shell.
- To connect with the new mongosh, just replace `mongo` by `mongosh`.

TOP Table of Contents TOP

Helpers

Show Databases

```
show dbs
db // prints the current database
```

Switch Database

```
use <database_name>
```

Show Collections

```
show collections
```

Run JavaScript File

```
load("myScript.js")
```

TOP Table of Contents TOP

CRUD

Create

```
db.coll.insertOne({name: "Max"})
db.coll.insert([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert
db.coll.insert([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered b
db.coll.insert({date: ISODate()})
db.coll.insert({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 50
```

Read

```
                // returns a single document
                // returns a cursor - show 20 results - "it" to display more

            name  "Max"  age  32   // implicit logical "AND".
            date          "2020-09-25T13:57:17.180Z"
            name  "Max"  age  32          "executionStats"  // or "queryPla
               "name"

// Count
             age  32           // estimation based on collection metadata
                               // estimation based on collection metadata
                    age  32   // alias for an aggregation pipeline - accur

// Comparison
            "year"   $gt  1970
            "year"   $gte  1970
            "year"   $lt  1970
            "year"   $lte  1970
            "year"   $ne  1970
            "year"   $in   1958  1959
            "year"   $nin   1958  1959

// Logical
```

```
// Logical
db.coll.find({name:{$not: {$eq: "Max"}}})
db.coll.find({$or: [{"year" : 1958}, {"year" : 1959}]})
db.coll.find({$nor: [{price: 1.99}, {sale: true}]})
db.coll.find({
  $and: [
    {$or: [{qty: {$lt :10}}, {qty :{$gt: 50}}]},
    {$or: [{sale: true}, {price: {$lt: 5 }}]}
  ]
})

// Element
db.coll.find({name: {$exists: true}})
db.coll.find({"zipCode": {$type: 2 }})
db.coll.find({"zipCode": {$type: "string"}})

// Aggregation Pipeline
db.coll.aggregate([
  {$match: {status: "A"}},
  {$group: {_id: "$cust_id", total: {$sum: "$amount"}}},
  {$sort: {total: -1}}
])

// Text search with a "text" index
db.coll.find({$text: {$search: "cake"}}, {score: {$meta: "textScore"}}).sort({

// Regex
db.coll.find({name: /^Max/})   // regex: starts by letter "M"
db.coll.find({name: /^Max$/i}) // regex case insensitive

// Array
db.coll.find({tags: {$all: ["Realm", "Charts"]}})
db.coll.find({field: {$size: 2}}) // impossible to index - prefer storing the
db.coll.find({results: {$elemMatch: {product: "xyz", score: {$gte: 8}}}})

// Projections
db.coll.find({"x": 1}, {"actors": 1})               // actors + _id
db.coll.find({"x": 1}, {"actors": 1, "_id": 0})     // actors
db.coll.find({"x": 1}, {"actors": 0, "summary": 0}) // all but "actors" and "s

// Sort, skip, limit
db.coll.find({}).sort({"year": 1, "rating": -1}).skip(10).limit(3)

// Read Concern
```

```
db.coll.find().readConcern("majority")
```

- db.collection.find()
- Query and Projection Operators
- BSON types
- Read Concern

Update

```
                                                                    
                "_id"  1    "year"  2016   // WARNING! Replaces the entire doc
                "_id"  1    $set   "year"  2016  name  "Max"
                "_id"  1    $unset   "year"  1
                "_id"  1    $rename   "year"  "date"
                "_id"  1    $inc   "year"  5
                "_id"  1    $mul   price              "1.25"   qty  2
                "_id"  1    $min   "imdb"  5
                "_id"  1    $max   "imdb"  8
                "_id"  1    $currentDate   "lastModified"  true
                "_id"  1    $currentDate   "lastModified"   $type  "timestamp"

// Array
                "_id"  1    $push   "array"  1
                "_id"  1    $pull   "array"  1
                "_id"  1    $addToSet   "array"  2
                "_id"  1    $pop   "array"  1     // last element
                "_id"  1    $pop   "array"   1    // first element
                "_id"  1    $pullAll   "array"   3  4  5
                "_id"  1    $push   scores   $each   90  92  85
                  "_id"  1  "grades"  80    $set   "grades.$"  82
                       $inc   "grades.$[]"  10
                   $set   "grades.$[element]"  100     multi  true  arrayFilt

// Update many
               "year"  1999    $set   "decade"  "90's"     "multi" true
                   "year"  1999    $set   "decade"  "90's"

// FindOneAndUpdate
                         "name"  "Max"    $inc   "points"  5     returnNewDoc

// Upsert
                "_id"  1    $set   item  "apple"   $setOnInsert   defaultQty
```

```
33
34  // Replace
35  db.coll.replaceOne({"name": "Max"}, {"firstname": "Maxime", "surname": "Beugne
36
37  // Save
38  db.coll.save({"item": "book", "qty": 40})
39
40  // Write concern
41  db.coll.update({}, {$set: {"x": 1}}, {"writeConcern": {"w": "majority", "wtime
```

Delete

```
1  name  "Max"
2  name  "Max"    justOne  true
3     // WARNING! Deletes all the docs but not the collection itse
4  name  "Max"    "writeConcern"   "w"  "majority"  "wtimeout"  50
5           "name"  "Max"
```

TOP Table of Contents TOP

Databases and Collections

Drop

```
copy code
1  db.coll.drop()    // removes the collection and its index definitions
2  db.dropDatabase() // double check that you are *NOT* on the PROD cluster... :-)
```

Create Collection

```
// Create collection with a $jsonschema
db.createCollection("contacts", {
   validator: {$jsonSchema: {
      bsonType: "object",
      required: ["phone"],
      properties: {
         phone: {
            bsonType: "string",
            description: "must be a string and is required"
         },
         email: {
            bsonType: "string",
            pattern: "@mongodb\.com$",
            description: "must be a string and match the regular expression pa
         },
         status: {
            enum: [ "Unknown", "Incomplete" ],
            description: "can only be one of the enum values"
         }
      }
   }}
})
```

Other Collection Functions

```
                                                                    
                                                                    
                                                                    
                                                                    
                  full  true
                        "new_coll"  true  // 2nd parameter to drop the target
```

TOP Table of Contents TOP

Indexes

List Indexes

```
db.coll.getIndexes()
db.coll.getIndexKeys()
```

Create Indexes

```
// Index Types
db.coll.createIndex({"name": 1})                // single field index
db.coll.createIndex({"name": 1, "date": 1})     // compound index
db.coll.createIndex({foo: "text", bar: "text"}) // text index
db.coll.createIndex({"$**": "text"})            // wildcard text index
db.coll.createIndex({"userMetadata.$**": 1})    // wildcard index
db.coll.createIndex({"loc": "2d"})              // 2d index
db.coll.createIndex({"loc": "2dsphere"})        // 2dsphere index
db.coll.createIndex({"_id": "hashed"})          // hashed index

// Index Options
db.coll.createIndex({"lastModifiedDate": 1}, {expireAfterSeconds: 3600})
db.coll.createIndex({"name": 1}, {unique: true})
db.coll.createIndex({"name": 1}, {partialFilterExpression: {age: {$gt: 18}}})
db.coll.createIndex({"name": 1}, {collation: {locale: 'en', strength: 1}})
db.coll.createIndex({"name": 1 }, {sparse: true})
```

Drop Indexes

```
db.coll.dropIndex("name_1")
```

Hide/Unhide Indexes

```
db.coll.hideIndex("name_1")
db.coll.unhideIndex("name_1")
```

- Indexes documentation

TOP Table of Contents TOP

Handy commands

copy code

```
use admin
db.createUser({"user": "root", "pwd": passwordPrompt(), "roles": ["root"]})
db.dropUser("root")
db.auth( "user", passwordPrompt() )

use test
db.getSiblingDB("dbname")
db.currentOp()
db.killOp(123) // opid

db.fsyncLock()
db.fsyncUnlock()

db.getCollectionNames()
db.getCollectionInfos()
db.printCollectionStats()
db.stats()

db.getReplicationInfo()
db.printReplicationInfo()
db.isMaster()
db.hostInfo()
db.printShardingStatus()
db.shutdownServer()
db.serverStatus()

db.setSlaveOk()
db.getSlaveOk()

db.getProfilingLevel()
db.getProfilingStatus()
db.setProfilingLevel(1, 200) // 0 == OFF, 1 == ON with slowms, 2 == ON

db.enableFreeMonitoring()
db.disableFreeMonitoring()
db.getFreeMonitoringStatus()

db.createView("viewName", "sourceColl", [{$project:{department: 1}}])
```

TOP Table of Contents TOP

Change Streams

```
watchCursor = db.coll.watch( [ { $match : {"operationType" : "insert" } } ] )

while (!watchCursor.isExhausted()){
   if (watchCursor.hasNext()){
      print(tojson(watchCursor.next()));
   }
}
```

TOP Table of Contents TOP

Replica Set

```
rs.status()
rs.initiate({"_id": "replicaTest",
  members: [
    { _id: 0, host: "127.0.0.1:27017" },
    { _id: 1, host: "127.0.0.1:27018" },
    { _id: 2, host: "127.0.0.1:27019", arbiterOnly:true }]
})
rs.add("mongodbd1.example.net:27017")
rs.addArb("mongodbd2.example.net:27017")
rs.remove("mongodbd1.example.net:27017")
rs.conf()
rs.isMaster()
rs.printReplicationInfo()
rs.printSlaveReplicationInfo()
rs.reconfig(<valid_conf>)
rs.slaveOk()
rs.stepDown(20, 5) // (stepDownSecs, secondaryCatchUpPeriodSecs)
```

TOP Table of Contents TOP

Sharded Cluster

```
sh.status()
sh.addShard("rs1/mongodbd1.example.net:27017")
sh.shardCollection("mydb.coll", {zipcode: 1})

sh.moveChunk("mydb.coll", { zipcode: "53187" }, "shard0019")
sh.splitAt("mydb.coll", {x: 70})
sh.splitFind("mydb.coll", {x: 70})
sh.disableAutoSplit()
sh.enableAutoSplit()

sh.startBalancer()
sh.stopBalancer()
sh.disableBalancing("mydb.coll")
sh.enableBalancing("mydb.coll")
sh.getBalancerState()
sh.setBalancerState(true/false)
sh.isBalancerRunning()

sh.addTagRange("mydb.coll", {state: "NY", zip: MinKey }, { state: "NY", zip: M
sh.removeTagRange("mydb.coll", {state: "NY", zip: MinKey }, { state: "NY", zip
sh.addShardTag("shard0000", "NYC")
sh.removeShardTag("shard0000", "NYC")

sh.addShardToZone("shard0000", "JFK")
sh.removeShardFromZone("shard0000", "NYC")
sh.removeRangeFromZone("mydb.coll", {a: 1, b: 1}, {a: 10, b: 10})
```

TOP Table of Contents TOP

Wrap-up

I hope you liked my little but - hopefully - helpful cheat sheet. Of course, this list isn't exhaustive at all. There are a lot more commands but I'm sure you will find them in the MongoDB documentation.

If you feel like I forgot a critical command in this list, please send me a tweet and I will make sure to fix it.