MongoDB Cheat Sheet

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

By Maxime Beugnet

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

- Replica Set
- 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 wa
mongo "mongodb://192.168.1.1:27017"
mongo "mongodb+srv://cluster-name.abcde.mongodb.net/<dbname>" --username <username
```

- 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
```

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 bulk
db.coll.insert({date: ISODate()})
db.coll.insert({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}
```

Read

```
db.coll.findOne() // returns a single document
db.coll.find()    // returns a cursor - show 20 results - "it" to display more
db.coll.find().pretty()
db.coll.find({name: "Max", age: 32}) // implicit logical "AND".
db.coll.find({date: ISODate("2020-09-25T13:57:17.180Z")})
db.coll.find({name: "Max", age: 32}).explain("executionStats") // or "queryPlanne
db.coll.distinct("name")
```

```
// Comparison
db.coll.find({"year": {$gt: 1970}})
db.coll.find({"year": {$gte: 1970}})
db.coll.find({"year": {$lt: 1970}})
db.coll.find({"year": {$lte: 1970}})
db.coll.find({"year": {$ne: 1970}})
db.coll.find({"year": {$in: [1958, 1959]}})
db.coll.find({"year": {$nin: [1958, 1959]}})

// 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({sco

// Regex
```

```
db.coll.find({field: {$size: 2}}) // impossible to index - prefer storing the siz
db.coll.find({results: {$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 "summ

// 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

copy code

```
db.coll.update({"_id": 1}, {"year": 2016}) // WARNING! Replaces the entire docume
db.coll.update({"_id": 1}, {$set: {"year": 2016, name: "Max"}})
db.coll.update({"_id": 1}, {$unset: {"year": 1}})
db.coll.update({"_id": 1}, {$rename: {"year": "date"} })
db.coll.update({"_id": 1}, {$inc: {"year": 5}})
db.coll.update({"_id": 1}, {$mul: {price: NumberDecimal("1.25"), qty: 2}})
db.coll.update({"_id": 1}, {$min: {"imdb": 5}})
db.coll.update({"_id": 1}, {$max: {"imdb": 8}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": true}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": {$type: "timestamp"}}}
```

```
db.coll.update({"_id": 1}, {$pop: {"array": -1}}) // first element
db.coll.update({"_id": 1}, {$pullAll: {"array" :[3, 4, 5]}})
db.coll.update({"_id": 1}, {$push: {scores: {$each: [90, 92, 85]}}})
db.coll.updateOne({"_id": 1, "grades": 80}, {$set: {"grades.$": 82}})
db.coll.updateMany({}, {$inc: {"grades.$[]": 10}})
db.coll.update({}, {$set: {"grades.$[element]": 100}}, {multi: true, arrayFilters

// Update many
db.coll.update({"year": 1999}, {$set: {"decade": "90's"}}, {"multi":true})
db.coll.updateMany({"year": 1999}, {$set: {"decade": "90's"}})

// FindOneAndUpdate
db.coll.findOneAndUpdate({"name": "Max"}, {$inc: {"points": 5}}, {returnNewDocume

// Upsert
db.coll.update({"_id": 1}, {$set: {item: "apple"}, $setOnInsert: {defaultQty: 100

// Replace
db.coll.replaceOne({"name": "Max"}, {"firstname": "Maxime", "surname": "Beugnet"}

// Save
db.coll.save({"item": "book", "qty": 40})

// Write concern
db.coll.update({}, {$set: {"x": 1}}, {"writeConcern": {"w": "majority", "wtimeout
```

Delete

copy code

```
db.coll.remove({name: "Max"})
db.coll.remove({name: "Max"}, {justOne: true})
db.coll.remove({}) // WARNING! Deletes all the docs but not the collection itself
db.coll.remove({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}
db.coll.findOneAndDelete({"name": "Max"})
```

Databases and Collections

Drop

```
db.coll.drop()    // removes the collection and its index definitions
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 patte
         },
         status: {
            enum: [ "Unknown", "Incomplete" ],
            description: "can only be one of the enum values"
         }
      }
   }}
})
```

```
db.coll.stats()
db.coll.storageSize()
db.coll.totalIndexSize()
db.coll.totalSize()
db.coll.validate({full: true})
db.coll.renameCollection("new_coll", true) // 2nd parameter to drop the target col
```

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

Table of Contents

```
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()
```

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)
```

copy code

```
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: MaxK
sh.removeTagRange("mydb.coll", {state: "NY", zip: MinKey }, { state: "NY", zip: M
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

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.

Check out our free courses on MongoDB University if you are not too sure what some of the above commands are doing.

If you have questions, please head to our developer community website where the MongoDB engineers and the MongoDB community will help you build your next big idea with MongoDB.

TOP Table of Contents TOP

Security Information

Trust Center

Support

Contact Us

Customer Portal

Atlas Status

Paid Support

Social

Github

Stack Overflow

LinkedIn

Youtube

Twitter

Twitch

Facebook