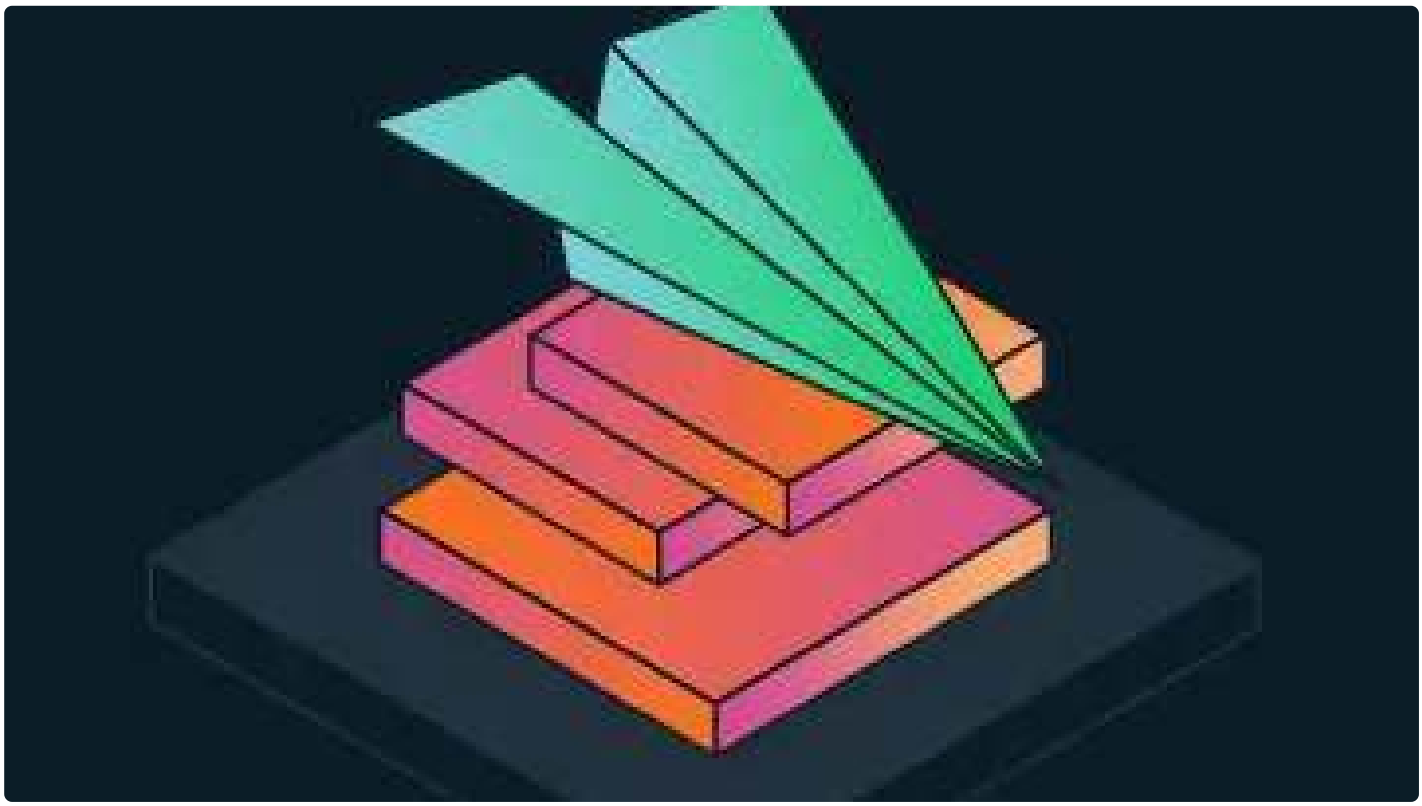
MongoDB Cheat Sheet

Maxime Beugnet

5 min read • Published Jan 31, 2022 • Updated Sep 29, 2023

MongoDB



Rate this quickstart ☆ ☆ ☆ ☆ ☆

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.

Updates

- September 2023: Updated for MongoDB 7.0.

Table of Contents

Connect via `mongosh`

```
mongosh # connects to mongodb://127.0.0.1:27017 by default
mongosh --host <host> --port <port> --authenticationDatabase admin -u <user> -p <pwd> # omit the password if you want a prompt
```

```
mongosh "mongodb://<user>:<password>@192.168.1.1:27017"
mongosh "mongodb://192.168.1.1:27017"
mongosh "mongodb+srv://cluster-name.abcde.mongodb.net/<dbname>" --apiVersion 1 --username <username> # MongoDB Atlas
```

- mongosh documentation.

Table of Contents

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")
```

Table of Contents

CRUD

Create

```
db.coll.insertOne({name: "Max"})
db.coll.insertMany([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert
db.coll.insertMany([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered bulk insert
db.coll.insertOne({date: ISODate()})
db.coll.insertOne({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 "queryPlanner" or "allPlansExecution"
db.coll.distinct("name")

// Count
db.coll.countDocuments({age: 32}) // alias for an aggregation pipeline - accurate count
db.coll.estimatedDocumentCount()  // estimation based on collection metadata

// 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}}
])
```

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

Update

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

// Array
db.coll.updateOne({"_id": 1}, {$push :{"array": 1}})
db.coll.updateOne({"_id": 1}, {$pull :{"array": 1}})
db.coll.updateOne({"_id": 1}, {$addToSet :{"array": 2}})
db.coll.updateOne({"_id": 1}, {$pop: {"array": 1}})  // last element
db.coll.updateOne({"_id": 1}, {$pop: {"array": -1}}) // first element
db.coll.updateOne({"_id": 1}, {$pullAll: {"array" :[3, 4, 5]}})
```

```
db.coll.updateOne({"_id": 1}, {$push: {"scores": {$each: [90, 92]}}})
```

Delete

```
db.coll.deleteOne({name: "Max"})
db.coll.deleteMany({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}})
db.coll.deleteMany({}) // WARNING! Deletes all the docs but not the collection itself and its index definitions
db.coll.findOneAndDelete({"name": "Max"})
```

Table of Contents

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 pattern"
         },
         status: {
            enum: [ "Unknown", "Incomplete" ],
            description: "can only be one of the enum values"
         }
      }
   }}
})
```

Other Collection Functions

```
db.coll.stats()
db.coll.storageSize()
db.coll.totalIndexSize()
db.coll.totalSize()
db.coll.validate({full: true})
```

Table of Contents

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})      // TTL index
db.coll.createIndex({"name": 1}, {unique: true})
db.coll.createIndex({"name": 1}, {partialFilterExpression: {age: {$gt: 18}}}) // partial index
db.coll.createIndex({"name": 1}, {collation: {locale: 'en', strength: 1}})    // case insensitive index with strength = 1 or 2
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

Handy commands

```
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.hello()
```

```

```

Table of Contents

Change Streams

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

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

Table of Contents

Replica Set

```
rs.status()
rs.initiate({"_id": "RS1",
  members: [
    { _id: 0, host: "mongodb1.net:27017" },
    { _id: 1, host: "mongodb2.net:27017" },
    { _id: 2, host: "mongodb3.net:27017" }]
})
rs.add("mongodb4.net:27017")
rs.addArb("mongodb5.net:27017")
rs.remove("mongodb1.net:27017")
rs.conf()
rs.hello()
rs.printReplicationInfo()
rs.printSecondaryReplicationInfo()
rs.reconfig(config)
rs.reconfigForPSASet(memberIndex, config, { options } )
db.getMongo().setReadPref('secondaryPreferred')
rs.stepDown(20, 5) // (stepDownSecs, secondaryCatchUpPeriodSecs)
```

Table of Contents

Sharded Cluster

```
db.printShardingStatus()

sh.status()
sh.addShard("rs1/mongodb1.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.startBalancer()
sh.stopBalancer()
sh.disableBalancing("mydb.coll")
```

```
14 sh.enableBalancing("mydb.coll")
15 sh.getBalancerState()
```

Table of Contents

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.

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.

Table of Contents

Rate this quickstart ☆ ☆ ☆ ☆ ☆

Related

TUTORIAL

Building with Patterns: The Bucket Pattern

May 17, 2022 | 3 min read

TUTORIAL

MongoDB Advanced Aggregations With Spring Boot and Amazon Corretto

Jun 26, 2024 | 5 min read

TUTORIAL

Revolutionizing AI Interaction: Integrating Mistral AI and MongoDB for a Custom LLM GenAI Application

Feb 13, 2024 | 11 min read

QUICKSTART

Getting Started With MongoDB and Sanic

Jul 12, 2024 | 5 min read

Request a Quickstart

MongoDB.

English

About

Careers

Investor Relations

Legal Notices

Privacy Notices

Security Information

Trust Center

Support

Contact Us

Customer Portal

Atlas Status

Customer Support

Manage Cookies

Social

GitHub

Stack Overflow

LinkedIn

YouTube

X

Twitch

Facebook