mongosh MongoDB shell

CONCEPTS

mongosh is a JavaScript shell: db is the current database, db.<name> a collection, and any JS works.

Deployment server or cluster
- **Database** db / use <db>
  - **Collection** db.users
    - **Document** BSON object

A database or collection is created lazily on the first insert; use <db> alone creates nothing.

CONNECT

mongosh
Connect to localhost:27017

mongosh "mongodb://<host>:27017/<db>"
Connect with a URI

mongosh --host <host> --port <port> -u <user> -p
Connect with options, prompt for password

--eval "<js>" <uri>
Run JavaScript and exit

mongosh <uri> <file.js>
Run a script file then exit

exit / quit()
Leave the shell

INSPECT

show dbs
List databases

use <db>
Switch database

db
Print the current database name

show collections
List collections

db.<coll>.countDocuments({})
Count matching documents

db.<coll>.getIndexes()
List indexes

help / db.help() / db.<coll>.help()
List available commands

READ

db.<coll>.find()
All documents (first batch)

db.<coll>.find({ <field>: <value> })
Match on equality

db.<coll>.findOne({ <filter> })
First match as one document

db.<coll>.find({}, { <f>: 1 })
Projection: pick fields

db.<coll>.find().sort({ <f>: -1 })
Sort, 1 ascending, -1 descending

db.<coll>.find().limit(<n>)
Cap results; chain .skip(<n>) to page

db.<coll>.find(<filter>).explain("executionStats")
Show how the query ran

Query operators

Operator	Meaning
$eq $ne	equals, not equals
$gt $gte $lt $lte	comparison
$in $nin	value in / not in list
$and $or $nor	combine conditions
$not	negate an operator
$exists	field is present
$type	BSON type check
$regex	pattern match
$elemMatch	one array element matches all
$size	array length
$all	array holds every value
$expr	compare fields, use aggregation expressions

WRITE

db.<coll>.insertOne({ <doc> })
Insert one document

db.<coll>.insertMany([{ }, { }])
Insert several documents

db.<coll>.updateOne(<filter>, { $set: { <f>: <v> } })
Change first match

WRITE (cont.)

db.<coll>.updateMany(<filter>, { $set: { } })
Change all matches

db.<coll>.updateOne(<filter>, <update>, { upsert: true })
Insert if nothing matches

db.<coll>.deleteOne(<filter>)
Delete first match

db.<coll>.deleteMany(<filter>)
Delete all matches

AGGREGATION

db.<coll>.aggregate([<stage>])
Run a pipeline of stages

{ $match: { <filter> } }
Filter documents early

{ $group: { _id: "$<f>", n: { $sum: 1 } } }
Group and count

{ $sort: { <f>: -1 } }
Order the stream

{ $project: { <f>: 1, <new>: "$<g>" } }
Reshape documents

{ $lookup: { from, localField, foreignField, as } }
Left join another collection

INDEXES

db.<coll>.createIndex({ <f>: 1 })
Create an ascending index

db.<coll>.createIndex({ <a>: 1, <b>: -1 })
Compound index, order matters

db.<coll>.createIndex({ <f>: 1 }, { unique: true })
Enforce uniqueness

USERS & ADMIN

db.createUser({ user: "<u>", pwd: passwordPrompt(), roles: ["readWrite"] })
Create a user, ask for password