MongoDB command cheatsheet

8 MIN READ • 11 AUG 17 • DEVOPS

#CHEATSHEET #DB

MongoDB is open-source, NoSQL database that stores data in a JSON-like document. It has highly flexible and dynamic data model which is faster, agile and scalable. It evolves to meet the need to manage the growing size and complexity of data. This post is all about MongoDB cheat sheet for quick reference.

MySQL & MongoDB

MySQL	MongoDB
database	database
table	collection

MySQL	MongoDB
column	field
primary key column	_id field(automatically created for each document)
joining tables	embedding or/and referencing documents
group by	aggregation
index	index

Install, Start & Stop

Installation steps will vary from system to system, follow this link to know more. To know how to start and stop MongoDB follow this link

MongoDB Component

Description	Command
server	mongod
client	mongo
export & import related	mongodump , mongorestore , mongoexport , mongoimport
monitoring related	mongostat , mongotop

Connecting to MongoDB

Description	Command
connect to local mongodb, default port: 27017	mongo
connect to remote mongodb	mongo host:port/db -u usr -p pwd
mongodb connect URI format	mongodb://usr:pwd@host:port/db

Description	Command

user must have proper access before connecting to DB. To create dbuser with RBAC(Role based access control)	use admin;
	
	`db.createUser({`
	`  user: 'dbadmin',`
	`  pwd: '12345678',`
	`  roles: [`
	`    {role: "dbOwner", db: "admin"}`
	`  ]`
	`})`

mongodb must be running in 'authorization mode' inorder to authentication work, put these into mongod yml config file	`security:`
	`  authorization: enabled`

Database

Description	Command
show all database	show dbs
show current database	db.getName()
switch(and create if doesn't exit) to database 'userdb'	use usersdb
drop current database	db.dropDatabase()

Collection

Description	Command
show all the collection under current database	db.getCollectionNames()
create collection 'users'	db.createCollection("users")
drop collection 'users'	db.users.drop()

Document: CRUD

Description	Command

MONGODB COMMAND CHEATSHEET search

Description	Command
insert document into ‘users’ collection	`db.users.insert({ userid: "123", age: 18, name: "vikash" });`
find all documents in ‘users’ collection	db.users.find()
find all documents from collection ‘users’ but select field ‘_id’, ‘name’ & ‘age’ only	`db.users.find( {}, { name: 1, age: 1 } )`
find all documents from collection ‘users’ but select field ‘name’ & ‘age’ only. By default ‘_id’ field is always selected, so to remove it.	`db.users.find( {}, { _id: 0, name: 1, age: 1 } )`
find all documents from collection ‘users’ where ‘name’=‘vikash’	`db.users.find({ name: "vikash" })`
find all documents from collection ‘users’ where ‘name’=‘vikash’ and select field ‘userid’ only	`db.users.find( { name: "vikash" }, { _id: 0 userid: 1 } )`
find all documents from collection ‘users’ where ‘name’!=‘vikash’	`db.users.find({ name: { $ne: "vikash" } })`
find all documents from collection ‘users’ where	`db.users.find({ name: "vikash",`

Description	Command

find all documents from collection ‘users’ where ‘name’=‘vikash’ or age=18	`db.users.find({`
	`  $or: [`
	`    {`
	`      name: "vikash"`
	`    },`
	`    {`
	`      age: 18`
	`    }`
	`  ]`
	`})`

find all documents from collection ‘users’ where ‘age > 18’	`db.users.find({`
	`  age: {`
	`    $gt: 18`
	`  }`
	`})`

find all documents from collection ‘users’ where ‘age < 18’	`db.users.find({`
	`  age: {`
	`    $lt: 18`
	`  }`
	`})`

find all documents from collection ‘users’ where ‘age >= 18’	`db.users.find({`
	`  age: {`
	`    $gte: 18`
	`  }`
	`})`

find all documents from collection ‘users’ where ‘name is like %ind%’	`db.users.find({`
	`  name: /ind/`
	`})`

find all documents from collection ‘users’ where ‘name is like ind%’	`db.users.find({`
	`  name: /^ind/`
	`})`

find all documents from collection ‘users’ where ‘name is like %ind%’ and order(ASC) by field ‘age’	`db.users.find({`
	`  name: /ind/`
	`})`
	`.sort({`
	`  age: 1`
	`})`

MONGODB COMMAND CHEATSHEET

search

Description	Command
find all documents from collection 'users' where 'name is like %ind%' and order(DESC) by field 'age'	`db.users.find({ name: /ind/ }) .sort({ age: -1 })`
find the number of documents in collection 'users'	db.users.find().count()
find the number of documents in collection 'users' where field 'name' exist	`db.users .find({ name: { $exists: true } }) .count()`
show distinct value for field 'name' of collection 'users'	`db.users.distinct('name')`
fetch 2 document skipping first 5 documents from collection 'users'	`db.users.find().limit(2).skip(5)`
updated field 'age' to 19 of collection 'users' where name = 'vikash'	`db.users.update( { name: "vikash" }, { $set: {age: 19} }, { multi: true } )`
increase current value of field 'age' by 5 of collection 'users' where name = 'vikash'	`db.users.update( { name: "vikash" }, { $inc: {age: 5} }, { multi: true } )`
delete all documents from 'users' collection	db.users.remove({})

Description	Command

delete all the documents of collection 'users' where name = 'vikash'	`db.users.remove(`
	`  {`
	`    name: "vikash"`
	`  }`
	`)`

Aggregate

Description	Command
SQL: SELECT, GROUP BY, HAVING	MongoDB: $project , $group , $match

count number of users in each 'age' group	`db.users.aggregate([{`
	`  $group: {`
	`    _id: "$age",`
	`    num_usr: {$sum: 1}`
	`  }`
	`}])`

count number of users in each 'age' group where name="vikash" & sort by 'age'	`db.users.aggregate([`
	`  { $match: {name: "vikash"} },`
	`  { $group: {_id: "$age", num_usr: {$sum: 1`
	`  { $sort: {age: 1} }`
	`])`

sum of field 'age' in each 'name' group	`db.users.aggregate([{`
	`  $group: {`
	`    _id: "$name",`
	`    sum_age: {$sum: "$age"}`
	`  }`
	`}])`

average age in each 'name' group	`db.users.aggregate([{`
	`  $group: {`
	`    _id: "$name",`
	`    avg_age: {$avg: "$age"}`
	`  }`
	`}])`

minmum age in each 'name' group	`db.users.aggregate([{`
	`  $group: {`
	`    _id: "$name",`
	`    avg_age: {$min: "$age"}`

Description	Command
maximum age in each 'name' group	`db.users.aggregate([{ $group: { _id: "$name", avg_age: {$max: "$age"} } }])`

Index

Description	Command
Create an index on field 'name' of collection 'users'	`db.users.ensureIndex({ name: 1 })`
Drop an index from field 'name' of collection 'users'	`db.users.dropIndex({ name: 1 })`
Create an compound index on field 'name' & 'age' of collection 'users'	`db.users.ensureIndex({ name: 1, age: 1 })`
Drop an compound index on field 'name' & 'age' of collection 'users'	`db.users.dropIndex({ name: 1, age: 1 })`

MongoImport & MongoExport

`mongoimport` & `mongoexport` is not recommended for production backup due to difference in json & bson, instead use `mongodump` & `mongorestore`. Don't run mongoimport & mongoexport from `mongo` shell becuase these are independent command.

Description	Command
Import data from accounts.json file to users db & accounts collection by default connect to localhost & port 27017	`mongoimport --db users --collection accounts --file accounts.json`
Import data from accounts.json file to users db, collection(on omit) name will be file name	`mongoimport --db users --file accounts.json`
Import data from accounts.json file to users db, replace matching(_id field) document	`mongoimport --db users --mode upsert --file accounts.json`
Import data from accounts.json file to users db, merge matching(_id field) document	`mongoimport --db users --mode merge --file accounts.json`
Import data from accounts.csv file to users db, collection accounts, headerline is not mandatory but mongodb uses it to identify fieldname	`mongoimport --db users --type csv --headerline --file accounts.json`
Import data from account.json file to remote host:port with auth enabled	`mongoimport -h mongodb1.example.com:3000 -u demouser -p demopwd -d users -c accounts --file accounts.json`
Import data from account.json file to remote host and port with auth enabled	`mongoimport --host mongodb1.example.com --port 3000 --username demouser --password demopwd --db users --collection accounts --file accounts.json`
Export accounts collection to accounts.json file	`mongoexport --db users --collection accounts --out accounts.json`
Export accounts collection to accounts.csv file, –fields are mandatory	`mongoexport --db users --collection accounts --type=csv --fields name,address --out accounts.csv`
Export accounts collection from remote host to accounts.json file	`mongoexport --db users --collection accounts -h mongodb1.example.com:3000 -u user -p 'password' --out accounts.json`

MONGODB COMMAND CHEATSHEET

search

Mongodump & Mongorestore

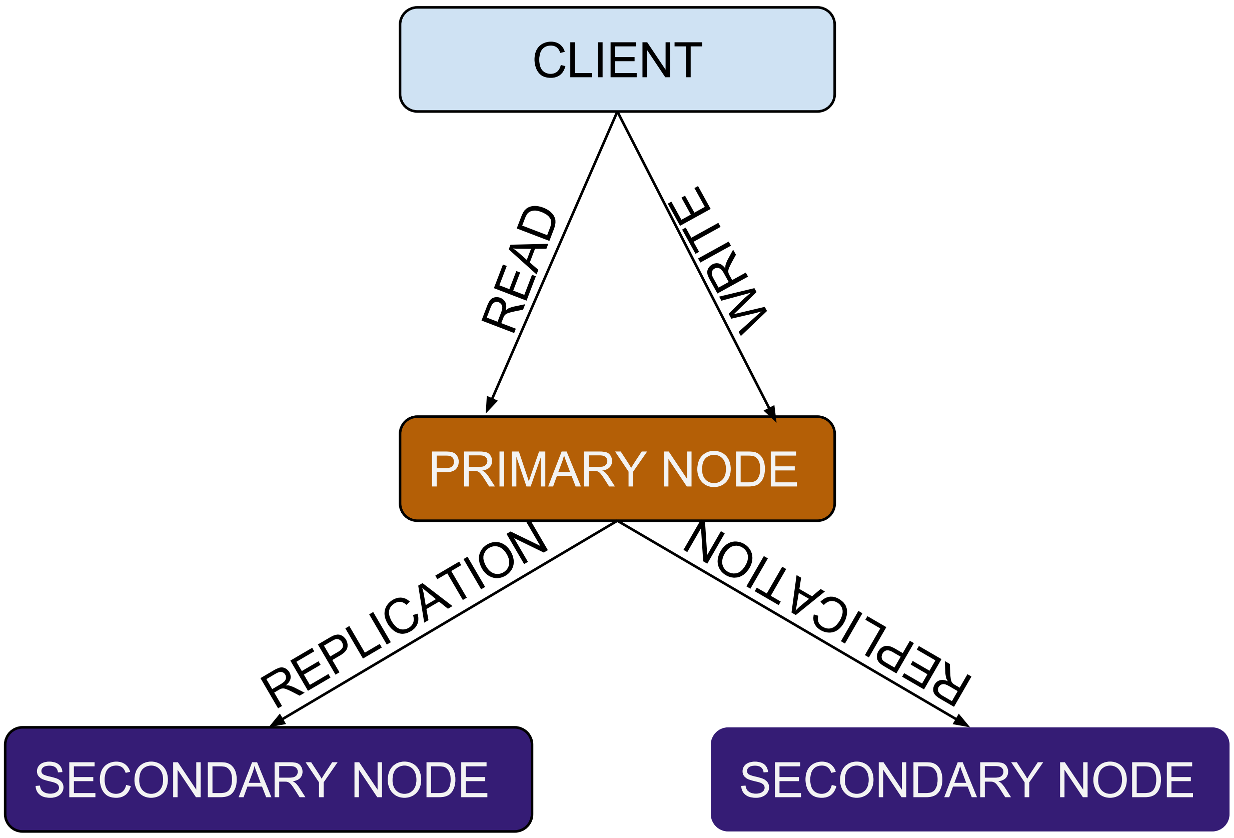
User must have proper access control before running `mongodump` & `mongorestore`. Two built-in mongodb role `backup` & `restore` can be used to give proper priviledge to user to perform backup & restore operation.

Description	Command
Dump all data from defaults host-localhost:27017 & dumpDirectroy-dump/	mongodump
Dump all data from specific host	mongodump --host mongodb1.example.com --port 3000
Dump all data from remote host with auth enabled	`mongodump --host mongodb.example.net --port 27017 --username user --password "pass"`
Dump all data to specific dump directory	mongodump --out mongodb1/date23Sept/
Dump only specific data	mongodump --db users --collection accounts
Restore from backuped location to default host localhost:27017	mongorestore mongodb1/date23Sept/
Restore from backuped location to specific host	`mongorestore --host mongodb1.example.com --port 3000 mongodb1/date23Sept/`
Restore from backuped location to specific host with auth enabled	`mongorestore --host mongodb1.example.com --port 3000 --username user --password 'pass' mongodb1/date23Sept/`

Replication & Sharding

Replication: is the process of syncing data across multiple nodes to provide availability in case of failure of another node. It increases the redundancy but increases the quality of service. Minimum 3 node is required, one is primary

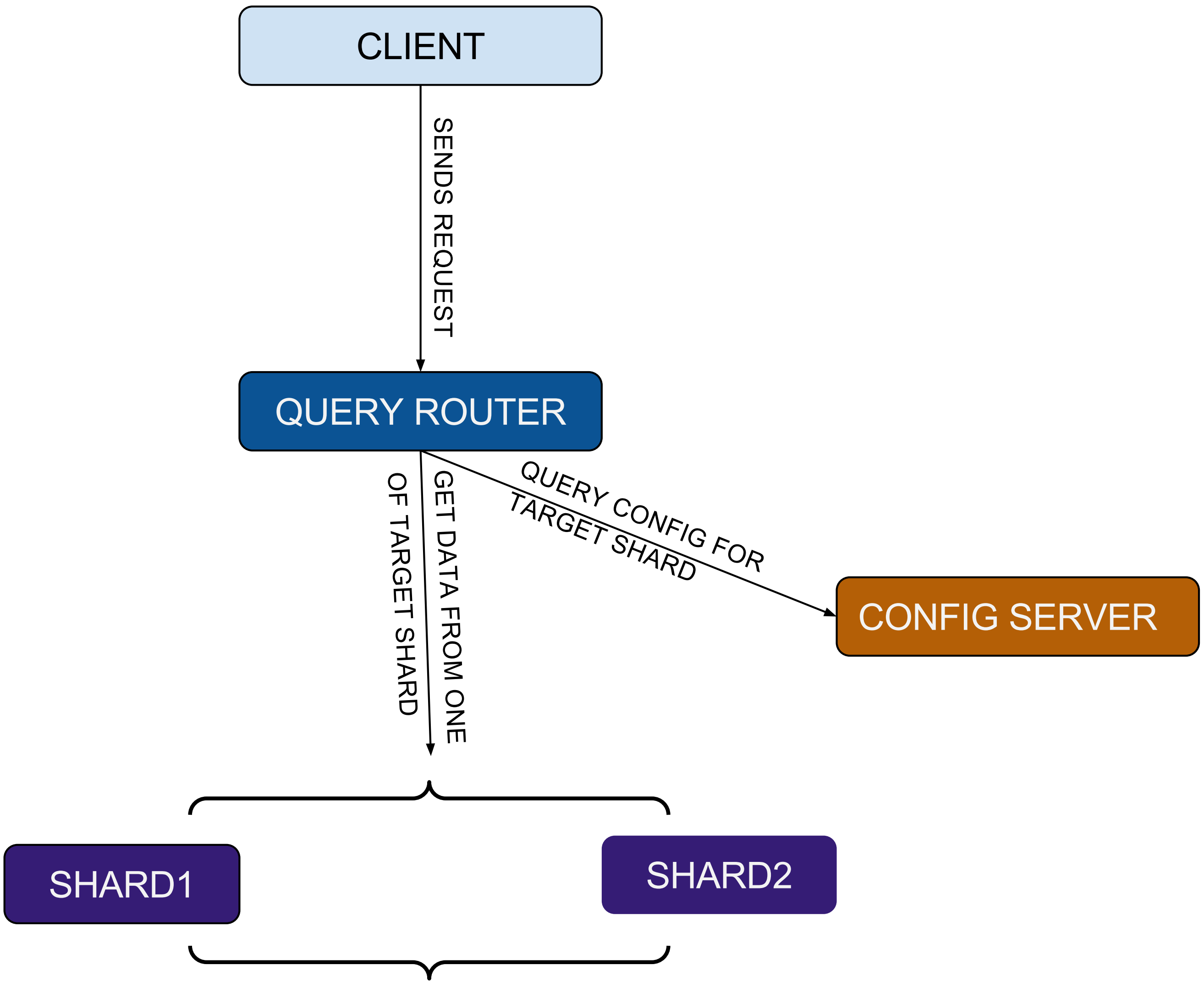
MONGODB COMMAND CHEATSHEET search



In case of failure of the primary node, secondary node can become primary node and once original node got recovered he again gets the role of the primary node.In order to add more host to replicaset(rs), first connect to primary node through mongo client then use below command, HOST_NAME is the address of the new node to be added. Each rs has a limit of 12 nodes.

```
rs.add(HOST_NAME:PORT)
```

Sharding: is the process of distributing data (when data grows considerably) across multiple machines to reduce load and increase the quality of service through horizontal scaling. Each shard is different replica set. Sharding in mongo is implemented through 3 component.


CLIENT
SENDS REQUEST
QUERY ROUTER
QUERY CONFIG FOR TARGET SHARD
CONFIG SERVER
GET DATA FROM ONE OF TARGET SHARD
SHARD1
SHARD2

1. **Shard:** used to store actual data. Each shard contain a subset of data, not the whole data. each shard can be deployed as a replica set.
2. **Config Server:** this store the metadata related to the mapping of cluster's data set to shard. Query server uses these metadata to target shard. Exactly 3 servers is used for this.
3. **Query Router:** client sends the query to this server and on behalf of the client, it communicates to config server, get the target shard and return data from target shard to client. 2 or more than 2 server is used to distribute the client load.

to discuss.

1 Comment **Sysleaf** 1 **Login**

Recommend Tweet Share **Sort by Best**

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Hardik Vasa • 5 months ago

nice article. some basic mongodb shell commands can be found on this link as well.

• Reply • Share ›

ALSO ON **SYSLEAF**

What is promise in javascript, when and how to use it

1 comment • 2 years ago

Avatar BOT MAN — hi sir, the blog on was promise as very helpful to me.thanks a lot. but a correction is needed in

How to create js binding to c/c++ library using nodejs FFI

7 comments • 2 years ago

Avatar Sneha — hi, I am getting this error after following this. Error: Dynamic Symbol Retrieval Error:

Auto hide fixed header on scroll using pure javascript a long with

3 comments • 2 years ago

Avatar Philipp — HI to fix the overflow issue you can ignore negative and to big scrollY values like so: currentScrollY =

Using1 service worker for cache management and offline browsing

1 comment • 2 years ago

Avatar Amos Waweru Welo — Thanks for the Article..I will work on project