Mongo DB configuration

MongoDB Engine

storageEngines" : [

"biggie",

"devnull",

"ephemeralForTest",

"wiredTiger" this Defalt engine

## Install MongoDB

**repos**

**vim /etc/yum.repos.d/mongodb-org-4.2.repo**

**yum install mongodb-org**

**systemctl start mongod**

**systemctl enable mongod**

**systemctl restart mongod**

**logon to mongodb**

**mongo**

**User creation and assign roles**

**Root privileges.**

**Full access**

**db.createUser({user: "vijay", pwd: "Nila@2017", roles:[{role: "root", db: "admin"}]})**

**password rest db.changeUserPassword("vijay", "Nila@2017")**

**admin privileges**

**read and write database**

**db.createUser( { user: "mongoadmin", pwd: "Nila@2017",**

**roles: [ {role: "userAdminAnyDatabase", db: "admin" } ]**

**}**

**)**

**logon to user in mongodb**

**mongo –u mongoadmin –p**

**use admin # select database**

**show users # show comment**

> use admin

switched to db admin

> show users

{

"\_id" : "admin.mongoadmin",

"userId" : UUID("ab9021a1-3649-4492-830a-761883e756fa"),

"user" : "mongoadmin",

"db" : "admin",

"roles" : [

{

"role" : "userAdminAnyDatabase",

"db" : "admin"

}

],

"mechanisms" : [

"SCRAM-SHA-1",

"SCRAM-SHA-256"

]

}

**Configuration File Options**

**Configuration file path**

***/etc/mongod.conf # file path***

***file format yaml***

***vim /etc/mongod.conf***

*# mongod.conf*

*# for documentation of all options, see:*

*# http://docs.mongodb.org/manual/reference/configuration-options/*

*# where to write logging data.*

***systemLog:***

*destination: file*

*logAppend: true*

***path: /var/log/mongodb/mongod.log***

*# Where and how to store data.*

***storage:***

***dbPath: /var/lib/mongo***

*journal:*

*enabled: true*

*# engine:*

*# wiredTiger:*

*# how the process runs*

***processManagement:***

*fork: true # fork and run in background*

***pidFilePath: /var/run/mongodb/mongod.pid***  *# location of pidfile*

*timeZoneInfo: /usr/share/zoneinfo*

*#* ***network interfaces***

*net:*

***port: 27017***

***bindIp: 127.0.0.1***  *# Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.bindIpAll setting.*

***#This remoto accest of mongobd server security session***

***security:***

***authorization:enabled***

*#operationProfiling:*

*#replication:*

*#sharding:*

*## Enterprise-Only Options*

*#auditLog:*

*#snmp:*

***storage:***

***dbPath: /var/lib/mongo***

***.wt file format WiredTiger***

***systemLog:***

***https://docs.mongodb.com/manual/reference/configuration-options/***

***path: /var/log/mongodb/mongod.log***

***SHARDING, ACCESS , STORAGE, INDEX, NETWORK , ERROR,FTDC***

# MongoDB CRUD Opaerations

# Create Opaeration, Read Opaerations, Update Opaeration, Delete Opaerations

# Create Opaerations

# Database Creation & open

show dbs

# use dbname # use store

Docment or Collection Creation

db.createCollection(“CollectionName”)

show collections()

Insert Documents

db.**product**.insertOne({item: "canvas", qty: 100, tags: ["cotton"], size: { h: 28, w: 35.5, uom: "cm"}})

product is collections or documents name

Sample output

{

"acknowledged" : true,

"insertedId" : ObjectId("5dd114a5dbf204fbaa7fe267")

}

Read Collections

db.collection.find({})

db.product.find({})

{ "\_id" : **ObjectId("5dd114a5dbf204fbaa7fe267")**, "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ], "size" : { "h" : 28, "w" : 35.5, "uom" : "cm" } }

Objectid \_id filed

MongoDB, each document stored in a collection requires a **unique \_id field** that acts as a **primary key**. If an inserted document omits the \_id field, the MongoDB driver automatically generates an ObjectId for the \_id field.

db.product.insertMany([

{ item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },

{ item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },

{ item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }

])

**Insert Methods**

db.collection.insertOne({})

db.collection.insertMany({})

db.collection.insert({})

Query Document

Comparisons, Logical and array operators

db.product.find({tags: "red"}) search

Out put

{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }

db.product.find({tags: {$in: ["red", "blue"]}}) IN operator

{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }

{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe26a"), "item" : "mousepad", "qty" : 25, "tags" : [ "gel", "blue" ], "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" } }

**db.product.find({tags: "red", qty: {$lt: 100 } }) AND operator**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }**

**> db.product.find({ $or: [{tags: "red"},{qty: {$lt: 100}}]}) OR operator**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe269"), "item" : "mat", "qty" : 85, "tags" : [ "gray" ], "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" } }**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe26a"), "item" : "mousepad", "qty" : 25, "tags" : [ "gel", "blue" ], "size" : { "h" : 19, "w" : 22.85, "uom" : "cm" } }**

**db.product.find({tags: "red", $or:[{qty: {$lt: 100}},{item: /^j/}]}) AND as well as OR Operator**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "cm" } }**

**UPDATE**

**db.product.updateOne({item: "canvas"},{$set: {"size.h": 30}})**

**{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }**

**> db.product.find()**

**{ "\_id" : ObjectId("5dd114a5dbf204fbaa7fe267"), "item" : "canvas", "qty" : 100, "tags" : [ "cotton" ], "size" : { "h" : 30, "w" : 35.5, "uom" : "cm" } }**

**UPDATE MANY**

**db.product.updateMany({"qty": {$lt: 30 }},{$set:{ "size.uom": "in"}})**

**{ "acknowledged" : true, "matchedCount" : 2, "modifiedCount" : 2 }**

**> db.product.find({"qty":{$lt:30}})**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe268"), "item" : "journal", "qty" : 25, "tags" : [ "blank", "red" ], "size" : { "h" : 14, "w" : 21, "uom" : "in" } }**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe26a"), "item" : "mousepad", "qty" : 25, "tags" : [ "gel", "blue" ], "size" : { "h" : 19, "w" : 22.85, "uom" : "in" } }**

**Replace**

**db.product.replaceOne({item: "mat"},{"item":"mat","qty": 100, "tags":["gray", "green"], "size":{"h":27.9, "w": 35.5, "uom": "cm"} })**

**{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }**

**> db.product.find({"item":"mat"})**

**{ "\_id" : ObjectId("5dd11719dbf204fbaa7fe269"), "item" : "mat", "qty" : 100, "tags" : [ "gray", "green" ], "size" : { "h" : 27.9, "w" : 35.5, "uom" : "cm" } }**

**Delete Documents**

**db.collection.deleteMany()**

**db.collection.deleteOne()**

**db.collection.remove()**

# Text Search

db.inside.find()

{ "\_id" : 1, "name" : "Java Hut", "description" : "Coffee and cakes" }

{ "\_id" : 2, "name" : "Burger Buns", "description" : "Gourmet hamburgers" }

{ "\_id" : 3, "name" : "Coffee Shop", "description" : "Just coffee" }

{ "\_id" : 4, "name" : "Clothes Clothes Clothes", "description" : "Discount clothing" }

{ "\_id" : 5, "name" : "Java Shopping", "description" : "Indonesian goods" }

> db.inside.creatIndex({name:"text", description:"text"})

**Text Index**

**db.inside.createIndex({name: "text", description: "text"})**

**{**

**"createdCollectionAutomatically" : false,**

**"numIndexesBefore" : 1,**

**"numIndexesAfter" : 2,**

**"ok" : 1**

**}**

**Search options**

**db.inside.find({$text: {$search: "java coffee shop"}})**

**{ "\_id" : 3, "name" : "Coffee Shop", "description" : "Just coffee" }**

**{ "\_id" : 1, "name" : "Java Hut", "description" : "Coffee and cakes" }**

**{ "\_id" : 5, "name" : "Java Shopping", "description" : "Indonesian goods" }**

**Text search options**

**db.inside.find({$text: {$search: "\"coffee shop\""}})**

**{ "\_id" : 3, "name" : "Coffee Shop", "description" : "Just coffee" }**

**db.inside.find( { $text: { $search: "coffee shop cake" } }, { score: { $meta: "textScore" } }**

**).sort( { score: { $meta: "textScore" } } )**

# Aggregation

**> db.station.find()**

{ "\_id" : ObjectId("5dd15b89df03afdc7d2a6111"), "cut\_id" : "A120", "amount" : 500, "status" : "A" }

{ "\_id" : ObjectId("5dd15b9bdf03afdc7d2a6112"), "cut\_id" : "A121", "amount" : 200, "status" : "A" }

{ "\_id" : ObjectId("5dd15ba4df03afdc7d2a6113"), "cut\_id" : "A122", "amount" : 210, "status" : "A" }

{ "\_id" : ObjectId("5dd15bc9df03afdc7d2a6114"), "cut\_id" : "A123", "amount" : 500, "status" : "B" }

{ "\_id" : ObjectId("5dd15bd8df03afdc7d2a6115"), "cut\_id" : "A124", "amount" : 500, "status" : "C" }

{ "\_id" : ObjectId("5dd15be3df03afdc7d2a6116"), "cut\_id" : "A125", "amount" : 500, "status" : "D" }

{ "\_id" : ObjectId("5dd15bf2df03afdc7d2a6117"), "cut\_id" : "A126", "amount" : 50, "status" : "A" }

{ "\_id" : ObjectId("5dd15bffdf03afdc7d2a6118"), "cut\_id" : "A127", "amount" : 30, "status" : "A" }

{ "\_id" : ObjectId("5dd15c0bdf03afdc7d2a6119"), "cut\_id" : "A128", "amount" : 40, "status" : "A" }

{ "\_id" : ObjectId("5dd15c14df03afdc7d2a611a"), "cut\_id" : "A129", "amount" : 140, "status" : "A" }

{ "\_id" : ObjectId("5dd15c1fdf03afdc7d2a611b"), "cut\_id" : "A130", "amount" : 170, "status" : "A" }

# *Aggregation Pipeline*

MongoDB’s [aggregation framework](https://docs.mongodb.com/manual/core/aggregation-pipeline/) is modeled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transforms the documents into an aggregated result

# *db.station.aggregate([{$match: {"status":"A"}},{$group: { \_id: "$cut\_id", total: { $sum: "$amount"}}}])*

# { "\_id" : "A126", "total" : 50 }

# { "\_id" : "A127", "total" : 30 }

# { "\_id" : "A128", "total" : 40 }

# { "\_id" : "A130", "total" : 170 }

# { "\_id" : "A122", "total" : 420 }

# { "\_id" : "A129", "total" : 140 }

# { "\_id" : "A120", "total" : 1040 }

# { "\_id" : "A121", "total" : 310 }

**First Stage**: The <$match> stage filters the documents by the status field and passes to the next stage those documents that have status equal to "A".

**Second Stage**: The <$group> stage groups the documents by the **cut\_id** field to calculate the sum of the amount for each unique **cut\_id.**

# *Back Up and Restore with MongoDB Tools*

***#mongodumb # mongoexport # mongoimport # mongorestore # mongodumb --out=/backup/***

***Remote server***

***#mongo --host=192.168.43.30 --port=27017 --username=vijay --password=”password” --out=/path/***

***#mongo –host=ip or hostname –port=27017 –username=vijay –password=”password” --collection= collection name --db=dbname --out=/path/***

**Restore**

**#mongorestore --db dbname --drop /path**

**#mongorestore --db admin --drop admin/**

**mongorestore --username="manju" --password="Nila@2017" --db="nila" --drop /vijay/nila/**

[**Replication**](https://docs.mongodb.com/manual/replication/)

**minimum 3 nodes need**

**maximum 50 nodes**

**oplog is called operation log**

***vim /etc*/hosts**

**192.168.5.71 mongodb0**

**192.168.5.68 mongodb1**

**192.168.5.70 mongodb2**

**vim */ect/mongod.conf***

***replication: #uncommand***

***replSetName: rs0 # replication set***

***oplogSizeMB: 100 #opertion log size***

**mongo**

rs.initiate( {

\_id : "rs0",

members: [

{ \_id: 0, host: "mongodb0:27017" },

{ \_id: 1, host: "mongodb1:27017" },

{ \_id: 2, host: "mongodb2:27017" }

]

**})**

rs.status()

mongo mongodb://mongodb0:27017, mongodb1:27017,mongodb2:27017

mongo mongodb://mongodb0,mongodb1,mongodb2/?replicaSet=rs0

rs.printReplicationInfo() # print status of repication

Secondary node to active

slave side active

**rs.slaveOk()**

# *MongoDB Performance*

[***https://docs.mongodb.com/manual/administration/analyzing-mongodb-performance/***](https://docs.mongodb.com/manual/administration/analyzing-mongodb-performance/)

[***https://stackify.com/mongodb-performance-tuning/***](https://stackify.com/mongodb-performance-tuning/)

***[Locking Performance](https://docs.mongodb.com/manual/administration/analyzing-mongodb-performance/" \l "locking-performance)***

***db.serverStatus().locks***

***db.serverStatus().globalLock***

***db.serverStatus(). globalLock.currentQueue.total***

***db.serverStatus().globalLock.totalTime***

***db.serverStatus().uptime***

***db.serverStatus().mem***

***dbserverStatus().connections***

***db.serverStatus().connections.current***

***db.serverStatus().connections.available***