安装

进入/etc/yum.repos.d目录，创建文件mongodb-org-5.0.repo,

**输入内容**：  
[mongodb-org]

name=MongoDB Repository

baseurl=http://mirrors.aliyun.com/mongodb/yum/redhat/7Server/mongodb-org/5.0/x86\_64/

gpgcheck=0

enabled=1

**开始安装**：yum -y install mongodb-org

**查看mongo的安装位置：whereis mongod**

修改配置文件 ：

vim /etc/mongod.conf

bindIp: 172.0.0.1 改为 bindIp: 0.0.0.0

启动

systemctl start mongod.service

停止

systemctl stop mongod.service

重启

systemctl restart mongod.service

查看

systemctl status mongod.service

设置开机自启动

systemctl enable mongod.service

mongo shell 的启动

./bin/mongo Docker exec进入容器后直接mongo就可以

指定主机和端口的方式启动

./bin/mongo --host=主机IP --port=端口

**命令**

**查看数据库**

**show dbs;**

**切换数据库 如果没有对应的数据库则创建**

**use 数据库名;**

**创建集合（表）**

**db.createCollection("集合名")**

**查看集合**

**showtables;**

**show collections;**

**删除集合**

**db.集合名.drop();**

**删除当前数据库**

**db.dropDatabase();**

**操作文档（CRUD）**

添加单个文档，如果集合不存在，会创建一个集合

db.文档名(表名).insertOne()

多个文档:

db.文档名(表名).insertMany(

[ 集合1, 集合2, 集合3]

)

查询集合所有文档

db.inventory.find({也可以根据条件查询})

**更新文档**

db.inventory.updateOne(

(根据哪个条件更新)

{ item: "paper" },

更新的数据

{

$set: { "size.uom": "cm", status: "P" },

获取当前时间，并设置新的属性

$currentDate: { lastModified: true }

}

)

多个文档

db.inventory.updateMany(

范围条件

{ "qty": { $lt: 50 } },

更新的数据

{ $set: { "size.uom": "in", status: "P" },

$currentDate: { lastModified: true }

}

)

替换

db.inventory.replaceOne(

以哪个条件查询要替换的数据

{ item: "paper" },

替换的信息

{ item: "paper", instock: [ { warehouse: "A", qty: 60 },{ warehouse: "B", qty: 40 } ] })

**删除文档**

删除集合所有文档

db.inventory.deleteMany({})

删除指定条件的文档

db.inventory.deleteMany({ status : "A" })

最多删除1个指定条件的文档

db.inventory.deleteOne( { status: "D" } )

**聚合操作**

db.orders.aggregate( [

// Stage 1: 匹配size:"medium"的文档

{

$match: { size: "medium" }

},

// Stage 2: 根据name统计过滤后的文档，并把"quantity"值相加

{

$group: { \_id: "$name", totalQuantity: {

$sum: "$quantity" } }

}

] )

**db.orders.aggregate( [**

**// Stage 1: 根据日期范围过滤**

**{**

**$match:**

**{**

**"date": { $gte: new ISODate( "2030-01-01" ), $lt: new ISODate( "2030-01-30" ) }**

**}**

**},**

**// Stage 2: 对过滤后文档以日期为条件进行分组并计算**

**{**

**$group:**

**{**

**\_id: { $dateToString: { format: "%Y-%m-%d", date: "$date" } },**

**totalOrderValue: { $sum: { $multiply: [ "$price", "$quantity" ] } },**

**averageOrderQuantity: { $avg: "$quantity" }**

**}**

**},**

**// Stage 3: 按照订单价值倒序排列文档**

**{**

**$sort: { totalOrderValue: -1 }**

**}**

**] )**

统计集合中文档数量

db.collection.count()

根据指定的字段进行过滤，去掉重复的文档

db.collection.distinct(“指定字段”)

创建索引，1代表升序，-1代表降序

db.records.createIndex( {

字段名(可以是嵌套 XX.xx) : 1/-1

} )

**字符串索引**

MongoDB提供了针对string内容的文本查询，Text Index支持任意属性值为string或string数组元素的索引查询。注意：一个集合仅支持最多一个Text Index，中文分词不理想推荐ES

创建:

db.fullText.createIndex(

{ description : "text" },

{ default\_language: "english" }

)

**查找**

db.fullText.find({"$text":{"$search":"pains"}})

全文索引名称

db.collection.createIndex(

{

content: "text",

"users.comments": "text",

"users.profiles": "text"

}

)

生成的默认索引名是\_text\_和索引字段的拼接，可以自定义索引名称

db.collection.createIndex(

{

content: "text",

"users.comments": "text",

"users.profiles": "text"

},

{

name: "MyTextIndex"

}

)

使用指定名称删除索引

db.collection.dropIndex("MyTextIndex")

备份

进入dockers后无需进入mongo，直接命令

单个库备份

**mongodump -h 127.0.0.1:27017 -d (数据库名) -o /usr/local**

**存放在容器中的地址**

整个库备份

**mongodump**

# **用户管理和安全认证**

以auth方式启动MongoDB:

docker run -itd --name 容器名称 -p 27017:27017 mongo:xxx --auth

**用户相关操作**

use admin; 进入admin数据库

db.createUser(

{

user:"root",

pwd:"123321",

roles:[{role:"root",db:"admin"}]

}

)

验证用户 :进入admin数据库验证

db.auth("账号","密码")，返回 1 说明认证成功

修改密码

db.changeUserPassword( 'root' , 'root' );

添加角色 进入admin数据库验证

db.grantRolesToUser('用户名',[{ role:'角色名' , db:'数据库名'}])

删除用户

db.dropUser("用户名")

恢复数据（在用户认证之后）

mongorestore -h localhost -u root -p 密码

--db mydb 备份路径 --authenticationDatabase

admin

**角色信息**