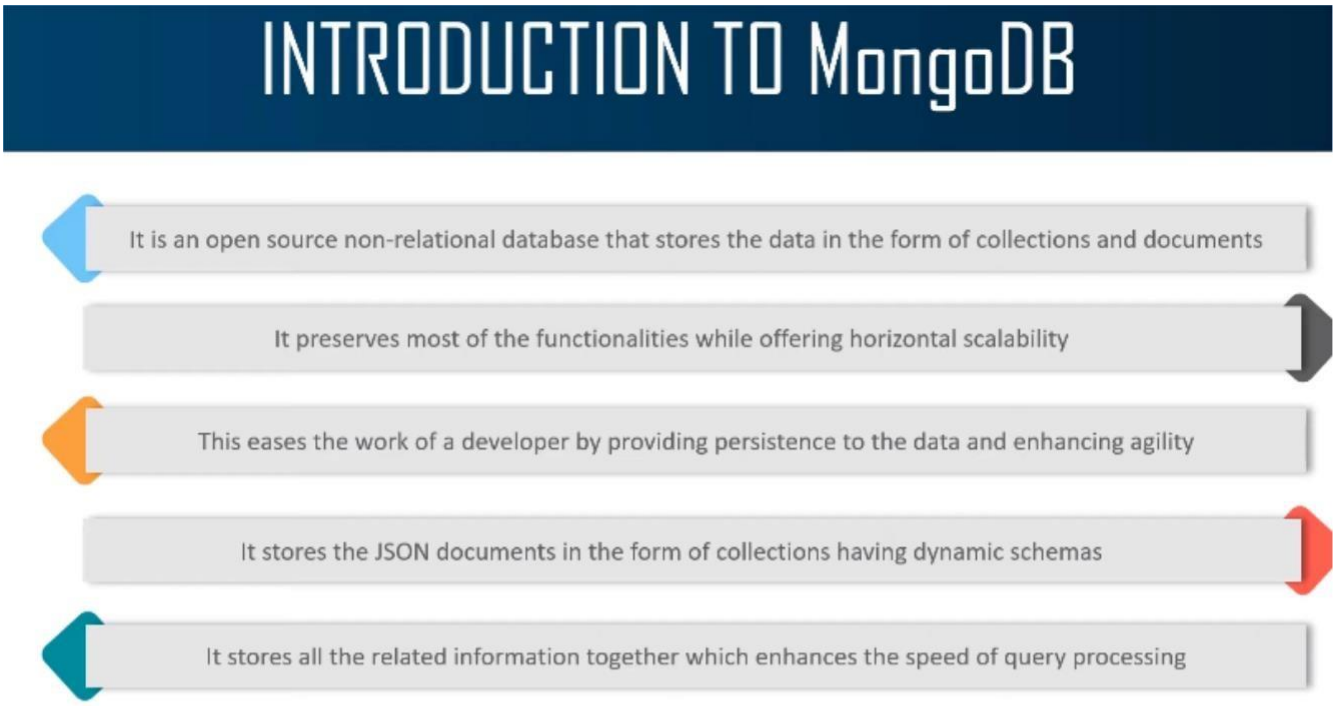
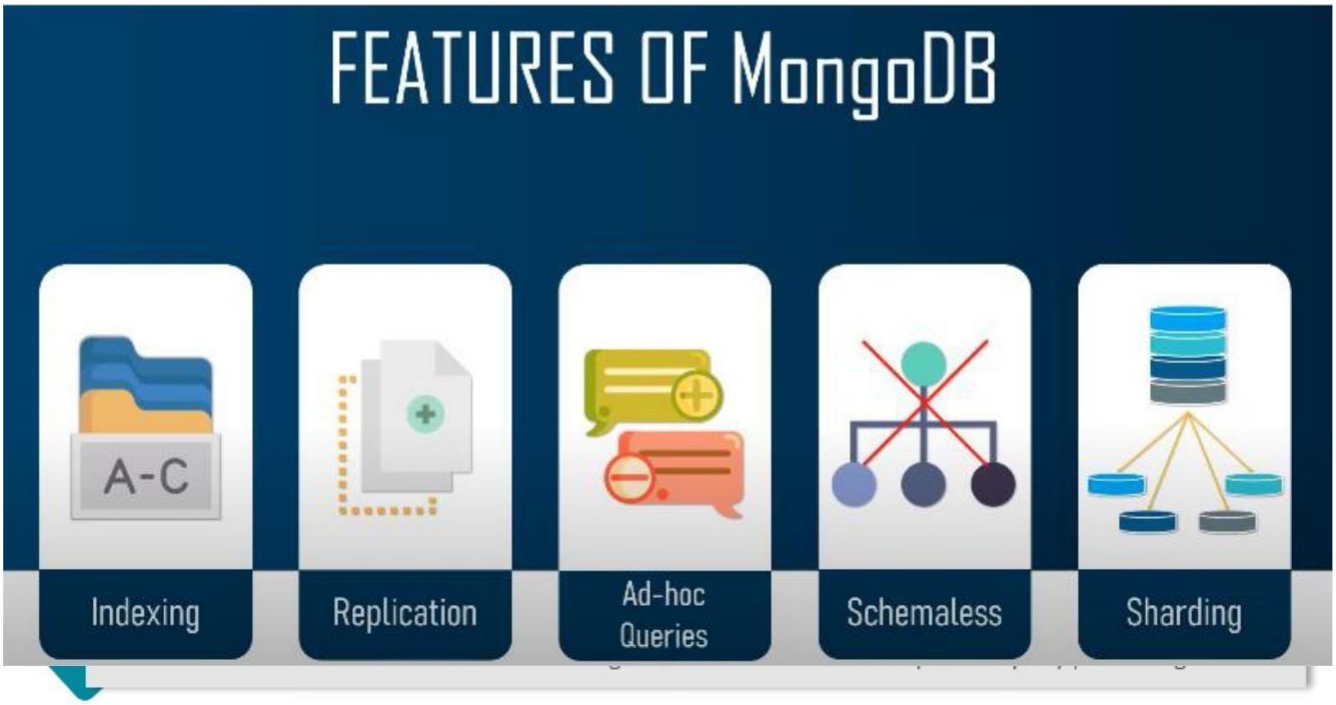
MONGO DB

MongoDB Atlas

Initial release:
February 11, 2009
Stable release :
5.0[2] Edit this on Wikidata / 13 July 2021
Repository:
github.com/mongodb/mongo
Written in:
C++, JavaScript, Python
Operating system:
Windows Vista and later, Linux,
OS X 10.7 and later, Solaris,[3] FreeBSD[4]
DB Type:
Type Document-oriented database
License:
Server Side Public License
Website:
www.mongodb.com





Official Site Doc Link: https://docs.mongodb.com/manual/crud/
https://docs.mongodb.com/manual/tutorial/query-documents/

MongoDB

It is a cross-platform, document oriented database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

Database

Database is a physical container for collections. Each database gets its own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB

Any relational database has a typical schema design that shows number of tables and the relationship between these tables. While in MongoDB, there is no concept of relationship.

Advantages of MongoDB over RDBMS

- Schema less: MongoDB is a document database in which one collection holds different documents. Number of fields, content and size of the document can differ from one

document to another.

- Structure of a single object is clear.

No complex joins.

- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
- Tuning.
- Ease of scale-out: MongoDB is easy to scale.
- Conversion/mapping of application objects to database objects not needed.
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Why Use MongoDB?

- Document Oriented Storage: Data is stored in the form of JSON style documents.
- Index on any attribute
- Replication and high availability

Auto-sharding

- Rich queries
- Fast in-place updates
- Professional support by MongoDB

Where to Use MongoDB?

- Big Data
- Content Management and Delivery

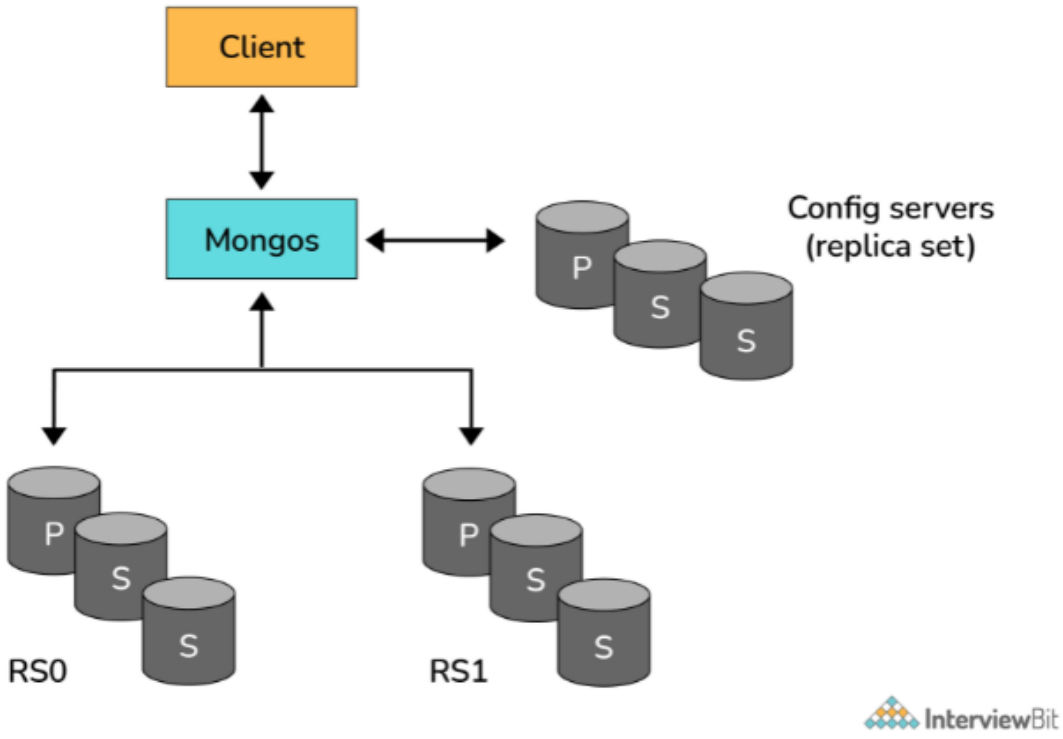
Mobile and Social Infrastructure

- User Data Management
- Data Hub

Sharding.

Sharding is the process of splitting data up across machines. We also use the term “partitioning” sometimes to describe this concept. We can store more data and handle more load without requiring larger or more powerful machines, by putting a subset of data on each machine.

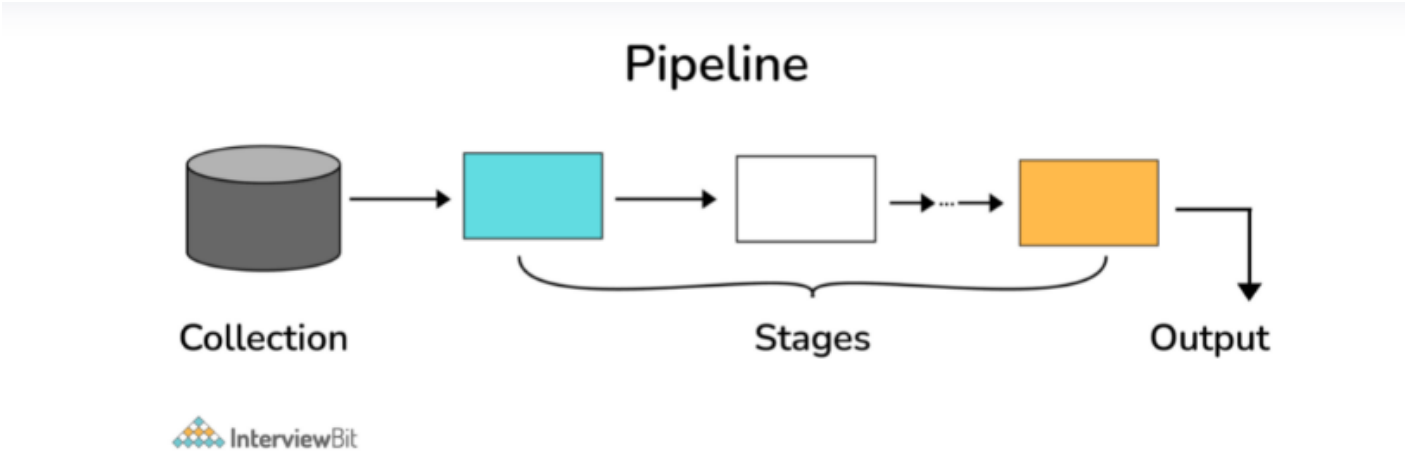
In the figure below, RS0 and RS1 are shards. MongoDB’s sharding allows you to create a cluster of many machines (shards) and break up a collection across them, putting a subset of data on each shard. This allows your application to grow beyond the resource limits of a standalone server or replica set.



Sharded Client Connection

Aggregation Framework in MongoDB?

- The aggregation framework is a set of analytics tools within MongoDB that allow you to do analytics on documents in one or more collections.
- The aggregation framework is based on the concept of a pipeline. With an aggregation pipeline, we take input from a MongoDB collection and pass the documents from that collection through one or more stages, each of which performs a different operation on its inputs (See figure below). Each stage takes as input whatever the stage before it produced as output. The inputs and outputs for all stages are documents—a stream of documents.

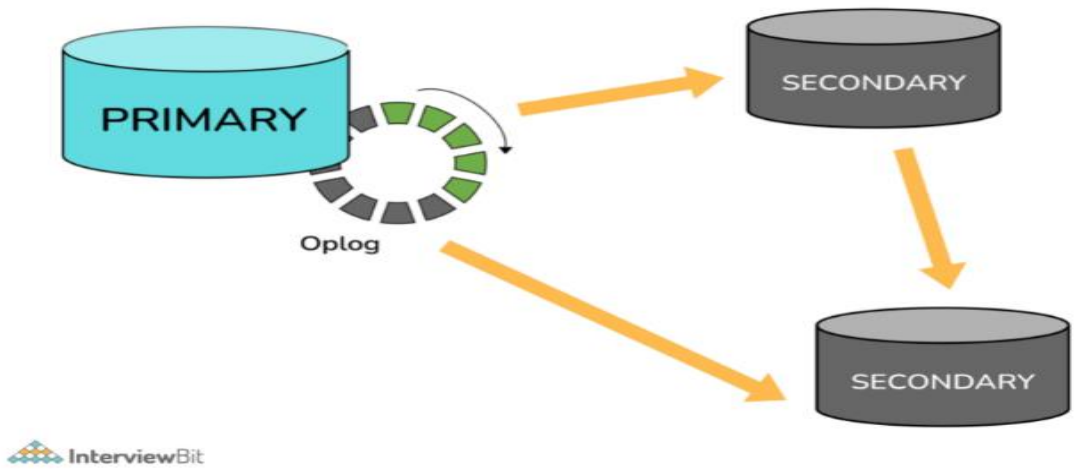


Replica Set in MongoDB?

To keep identical copies of your data on multiple servers, we use replication. It is recommended for all production deployments. Use replication to keep your application running and your data safe, even if something happens to one or more of your servers.

Such replication can be created by a replica set with MongoDB. A replica set is a group of servers with one primary, the server taking writes, and multiple secondaries, servers that keep copies of the primary's data. If the primary crashes, the secondaries can elect a new primary from amongst themselves

Architecture in MongoDB.



some utilities for backup and restore in MongoDB?

The mongo shell does not include functions for exporting, importing, backup, or restore. However, MongoDB has created methods for accomplishing this, so that no scripting work or complex GUIs are needed. For this, several utility scripts are provided that can be used to get data in or out of the database in bulk. These utility scripts are:

- mongoimport

- mongoexport
- mongodump
- mongorestore

Installation:-

Mongo DB From Below URL (32 Bit)

https://www.mongodb.org/dl/win32/i386

Download File :- (mongodb-win32-i386-v3.2-latest-signed.msi) and install. Now

create a Project Like (D:\MERN\NodeExpressMongo_Project)

open CMD
> npm init
replace index.js to app.js

you will get your package.json file. Now install some dependencies.

> npm install express
>npm install mongodb
>npm install mongoose
>npm install -g nodemon –save-dev

Package.json -

```
{
  "name": "nodeexpressmongo_project",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
"start": "nodemon app.js"// replace test with start.
  },

  "dependencies": {
  },
  "devDependencies": {
  }
}
```

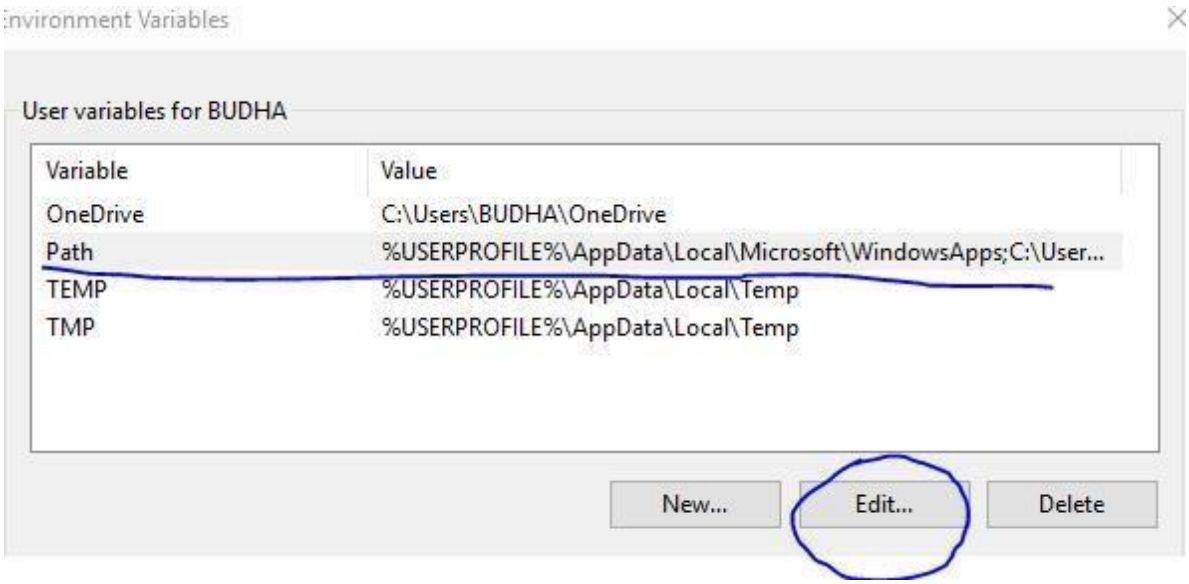
Configuring:-

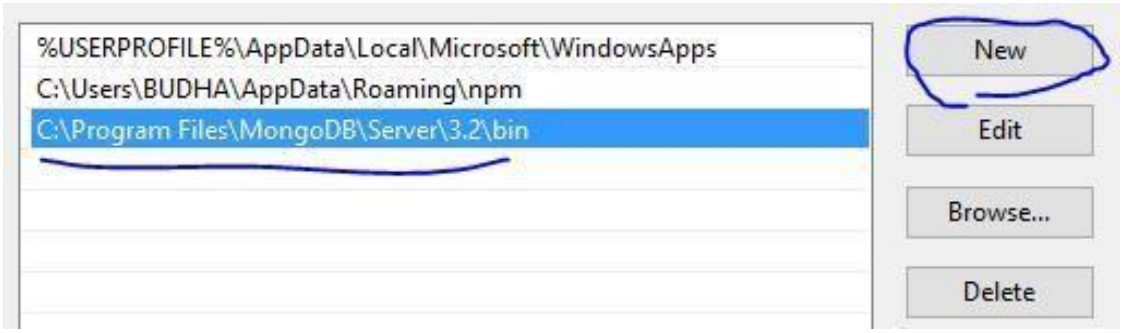
environment Variable

C:\Program

Files\MongoDB\Server\3.2

\bin





Create a folder as below (To save data in local Mongo DB): -

C:\data\db

Go to your Project path and open cmd and run below command to start server (only for 32 bit PC.)

D:\MERN\NodeExpressMongo_Project>**mongod --storageEngine=mmapv1 -- dbpath "C:\data\db"**

you will get below screen


Windows Security Alert
Windows Firewall has blocked some features of this app
Windows Firewall has blocked some features of mongod on all public and private networks.
Name: mongod
Publisher: Unknown
Path: C:\program files\mongodb\server\3.2\bin\mongod.exe
Allow mongod to communicate on these networks:
Private networks, such as my home or work network
Public networks, such as those in airports and coffee shops (not recommended because these networks often have little or no security)
What are the risks of allowing an app through a firewall?
Allow access
Cancel

for 64 bit PC

D:\MERN\NodeExpressMongo_Project>**mongod**

IF your Server started successfully you will see below screen.

```
2021-07-18T19:22:51.126+0530 I CONTROL  [main]
2021-07-18T19:22:51.154+0530 I CONTROL  [initandlisten] MongoDB starting : pid=620 port=27017 dbpath=C:\data\db 32-bit h
ost=ExploreMe-PC
2021-07-18T19:22:51.154+0530 I CONTROL  [initandlisten] targetMinOS: Windows Vista/Windows Server 2008
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] db version v3.2.21-2-g105acca0d4
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] git version: 105acca0d443f9a47c1a5bd608fd7133840a58dd
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] allocator: tcmalloc
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] modules: none
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] build environment:
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten]     distarch: i386
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten]     target_arch: i386
2021-07-18T19:22:51.155+0530 I CONTROL  [initandlisten] options: { storage: { dbPath: "C:\data\db", engine: "mmapv1" } }

2021-07-18T19:22:52.470+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.470+0530 I CONTROL  [initandlisten] ** WARNING: This 32-bit MongoDB binary is deprecated
2021-07-18T19:22:52.471+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.471+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:52.472+0530 I CONTROL  [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2021-07-18T19:22:52.472+0530 I CONTROL  [initandlisten] **       32 bit builds are limited to less than 2GB of data (or
less with --journal).
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten] **       Note that journaling defaults to off for 32 bit and is
currently off.
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten] **       See http://dochub.mongodb.org/core/32bit
2021-07-18T19:22:52.473+0530 I CONTROL  [initandlisten]
2021-07-18T19:22:53.062+0530 I NETWORK  [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2021-07-18T19:22:56.736+0530 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory 'C
:/data/db/diagnostic.data'
2021-07-18T19:22:56.744+0530 I NETWORK  [initandlisten] waiting for connections on port 27017
```

You need to open two terminal

C:\Windows\System32\cmd.exe - mongod --storageEngine=mmapv1 --dbpath "C:\data\db"

```
D:\MERN\NodeExpressMongo_Project>mongod --storageEngine=mmapv1 --dbpath "C:\data\db"
```

C:\Windows\System32\cmd.exe - nodemon run start

```
D:\MERN\NodeExpressMongo_Project>nodemon run start
[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node run start app.js`
(node:4328) [MONGODB DRIVER] Warning: Current Server Discovery and Monitoring engine i
in a future version. To use the new Server Discover and Monitoring engine, pass option
he MongoClient constructor.
(Use `node --trace-warnings ...` to show where the warning was created)
connected..
```

Open new command prompt and run below command

1. D:\MERN\NodeExpressMongo_Project>mongod --storageEngine=mmapv1 --dbpath "C:\data\db"

(to keep server running)

Again Open new command prompt and run below command

2. D:\MERN\NodeExpressMongo_Project>nodemon run start

to run your application.

Adding Code to get response using PostMan

Now create a app.js file and add below code.

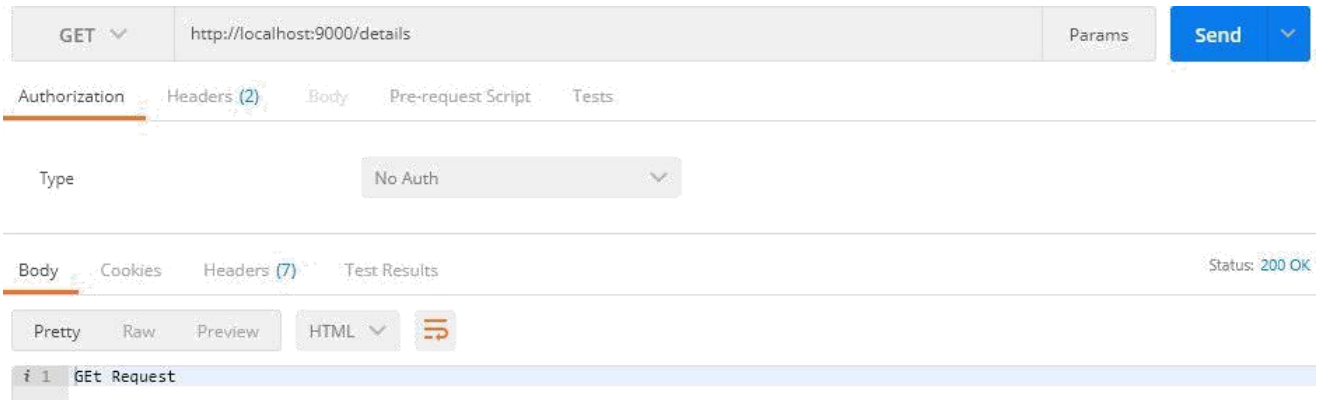
```
const express = require('express'); const
mongoose = require('mongoose'); const url =
'mongodb://localhost/MENApp'; const app =
express();

mongoose.connect(url, {useNewUrlParser: true});
const con = mongoose.connection; con.on('open',
() => {
[redacted]
});

const customRoute = require('./Router/menappRoute');
app.use('/details', customRoute); // this is a middleware.
app.listen(9000, () => {
[redacted]
});
```

IF your server work correctly you will get connected… message in CMD. Now we need to create Router to use (GET/POST/PATCH/DELETE Operations) so create a folder let say Router → menAppRoute.js and add below code

```
const express = require('express');
const router = express.Router();
[redacted]
});
```

```
module.exports = router;
```

Now press send button of your Postman. (URL : http://localhost:9000/details)



MONGO DB COMMANDS

To run mongo commands first you have to open a new command prompt and run below command (for 32 bit) to keep database active and in running state.

Open new command prompt and run below command

> mongod --storageEngine=mmapv1 --dbpath "C:\data\db" (mongod for 64 bit)

Again Open new command prompt and run below command

> nodemon run start

Again Open new command prompt and run below command

> mongo

you will see a shell like below with a cursor blinking.

STEP BY STEP

TO CONNECT WITH MONGODB
>mongod

TO RUN MONGO COMMAND
>mongo

TO SEE ALL EXISTING DATABASES
>show dbs

TO SELECT ANY DATABASE
>use xyz_db

TO SEE COLLECTIONS/TABLES
>show collections

TO CREATE COLLECTIONS/TABLES
>db.createCollection("xyz_employees")

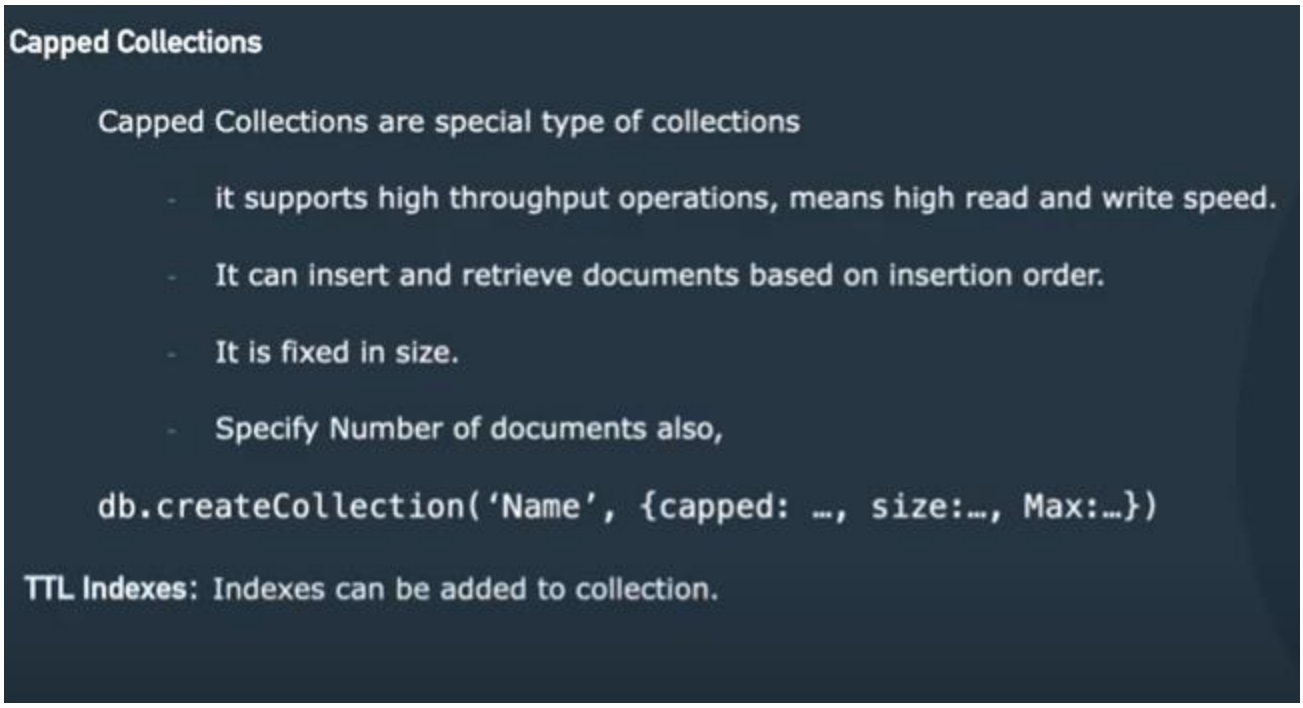
TO DELETE COLLECTION
>db.collection_name.drop()

TO DELETE DATABASE
>db.dropDatabase()

TO SEE IST OF DB COMMANDS
>db.help()

TO CREATE A COLLECTIONS/TABLES AND INSERT DOCUMENT/RECORD
>db.xyz_employees2.insertOne({"first_name": "James"}) // it will create a collection xyz_employees2 and then insert the object.

TO CREATE COLLECTIONS/TABLES WITH FIXED DOCUMENT/RECORD OF MAX SIZE.
>**db.createCollection('logs', {capped: true, size: 2048, max: 2})**
>**db.logs.insertOne({'count': 1})**
>**db.logs.insertOne({'count': 2})**
>**db.logs.insertOne({'count': 3})**



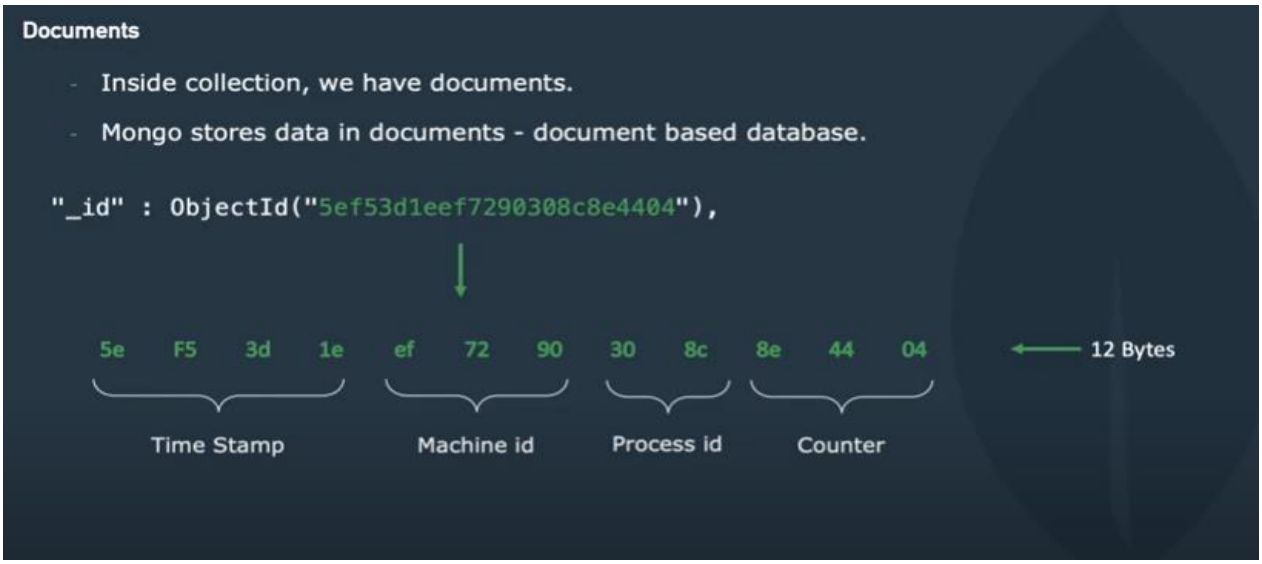
INSERT MULTIPLE RECORDS

>db.employee.insertMany([{'name':'Virat', 'department': 'computers', 'salary':30000},{'name': 'Rahul', 'department': 'commerce','salary': 20000},{'name':'Priya','department': 'science', 'salary': 40000}])

TO SEE DOCUMENT/RECORD
>**db.logs.find().pretty()**
// output
{"_id": ObjectId("5ef53d1eef7290308c8e4404), "count": 2}

{“_id”: ObjectId(“7ef53d1eef0976308c8e4212), “count”: 3}

count: 1 will be removed and newest entry will be inserted as max legth is 2 only.
This is useful for loggers where we use some authentcation for logging.



TO SEE LIST OF DB COMMANDS
>db.db_name.help()

TO DROP A COLLECTIONS
>db.collection_name.drop()

TO SEE LIST OF COLLECTION COMMANDS
>db.collection_name.help()

To exit from mongodb.
>**quit()**

To see server related details
>**db.serverCmdLineOpts()**

To see information related to server , host and some other info.

db.employee.find().explain('executionStats')

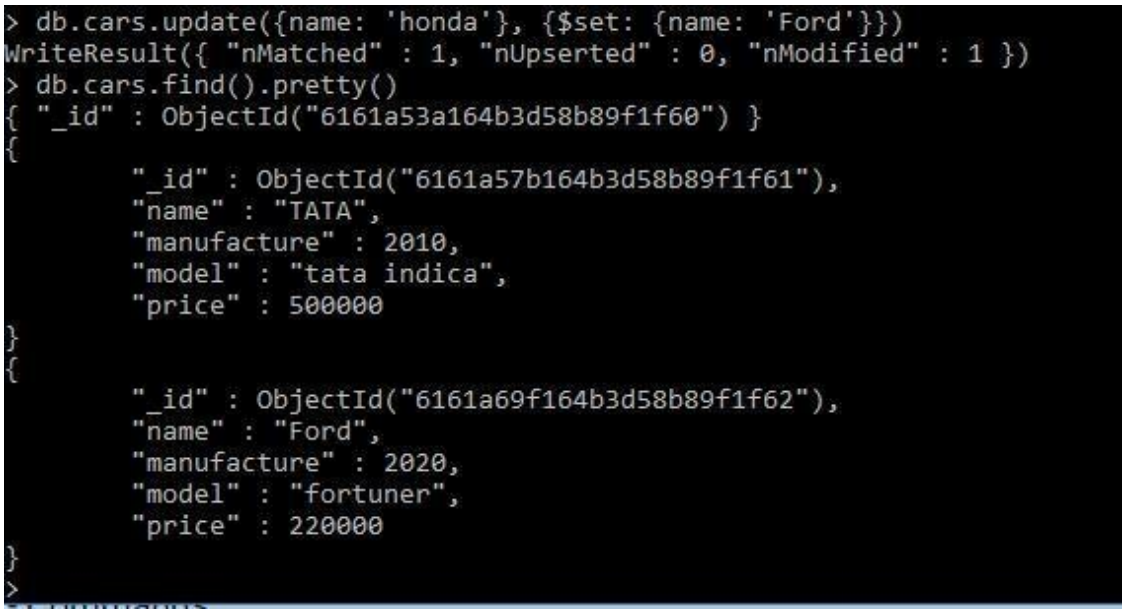
To check db is a Master db
db.isMaster()

Data in mogodb is stored in bson (Binary JSON) formt.

Note : if you insert records more than maxWriteBtachSize, Mongo db will create two seperate batches to execute the operation.

Updating specific field inside any record.

> db.cars.update({name: 'honda'}, {$set: {name: 'Ford'}})



ADDING NEW FIELD TO EXISTING RECORD

```
> db.cars.update({name: "Ford"}, {$set: {quantity: 400}}, {$upsert: true})
```

INSERT NESTED RECORDS

```
> db.employee.update({'name': 'Virat'}, {$set: {'address': {'street': 'abc street'}}}, {$upsert: true})
```

DELETE ANY SPECIFIC RECORD FROM COLLECTIONS

```
> db.employee.remove({name: 'Virat' })
```

SEE SPECIFIC RECORD ONLY

```
>db.employee.find({name: 'Priya'}).pretty()
```

FIND SPECIFIC RECORD GIVING SOME PARAMETER

```
>db.employee.find( { "monthlySalary": { $elemMatch: { "name" : "Priya"}}}).pretty();
```

FIND ALL

```
>db.employee.find({}).pretty()
```

SEARCH QUERIES

SEARCH BY TEXT

To search by text, you need to add index as text.

```
>db.employee.createIndex({name: 'text'})
>db.employee.find({$text: {$search: "\"Priya\""}}).pretty()
```

```
> db.cars.update({name: 'Ford'}, {$set: {quantity: 400}}, {$upsert: true})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.cars.find().pretty()
{ "_id" : ObjectId("6161a53a164b3d58b89f1f60") }
{
        "_id" : ObjectId("6161a57b164b3d58b89f1f61"),
        "name" : "TATA",
        "manufacture" : 2010,
        "model" : "tata indica",
        "price" : 500000
}
{
        "_id" : ObjectId("6161a69f164b3d58b89f1f62"),
        "name" : "Ford",
        "manufacture" : 2020,
        "model" : "fortuner",
        "price" : 220000,
        "quantity" : 400
}
>
```

RENAME A SPECIF FIELD

```
> db.employee.update({'name': 'Virat'}, {$rename: {'salary': 'monthlySalary'}})
```

RENAME ALL FIELDS

> db.employee.update({},{$rename: {'salary': 'monthlySalary'}})

The SET Modifier in MongoDB?

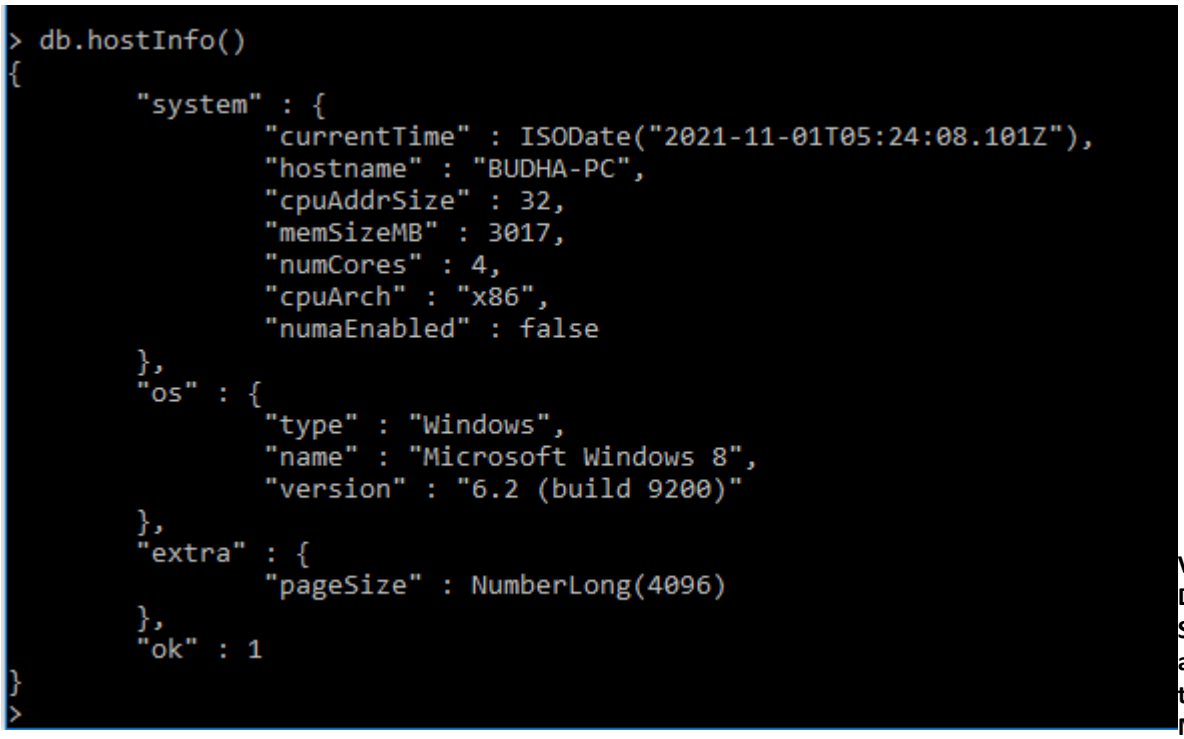
If the value of a field does not yet exist, the "$set" sets the value. This can be useful for updating schemas or adding user-defined keys.

Example:

```
> db.users.findOne()
{
  "_id" : ObjectId("4b253b067525f35f94b60a31"),
  "name" : "alice",
  "age" : 23,
  "sex" : "female",
  "location" : "India"
}
```

To add a field to this, we use “$set”:

```
> db.users.updateOne({"_id" :
ObjectId("4b253b067525f35f94b60a31")},
... {"$set" : {"favorite book" : "Start with Why"}})
```



Various Data Structure and Tools to Manage Data Model in MongoDB

Data Model -

The data models allow applications to retrieve and manipulate related data in a single database operation.

Tools -

Aqua Data Studio for MongoDB
Aqua data studio is an overall development and management tool for the MongoDB database. It is not limited to designing the schema but also manages live databases.

DBeaver
It is also a full-fledged designing and management tool for MongoDB as well as SQL

ER/Studio
It is a powerful data architect software which is capable of designing MongoDB schema.

Nosql and sql database

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents.
Documents consist of key-value pairs which are the basic unit of data in MongoDB.