

Today, we will be talking about databases and do operations on one ourselves

kingabesh ★

@here what is your definition of a database?

Your first task was designed to show you how to write data to somewhere, albeit, a file  
You will make use of the fs module as you proceed and when you start writing CLIs and  
certain packages (edited)

**The database** is a systematic collection of data databases support storage and  
manipulation of data. **Databases** make data management easy

The main operations you carry out on a database are:

Insert, Read, Update and Delete

We have two major types of databases

Relational databases and non-relational databases

Relational databases are well suited for creating relations and make use of tables, rows,  
and columns

Non-relational databases make use of documents, collections, and fields

If we are to create a comparison

Relational	Non-relational
Tables	=== Documents
Rows	=== Collections
Columns	=== Fields

Examples of relational databases are but not limited to MySQL, Microsoft SQL, Oracle  
Database, PostgreSQL

Example of non-relational databases are: MongoDB, DocumentDB, Cassandra, Couchbase,  
HBase, Redis, and Neo4j

I highlighted the words creating relations

In the world of databases, you are better off associating items with their owners.

Let me use you guys as examples

Every intern here must have a slack username:

You are related to your slack username in the world of databases

This is helpful so we can do stuff like `getting all slack details for a particular intern`

That relation enables us to do this

This is just what you need to know

Should in case anyone decides to ask you what relations are

Non-relational databases do not prioritize creating relations where you can relate items with their owners of course

The key difference to note is, relational databases make use of tables but non-relational databases do not

That is easily the difference.

Our main focus today is on a database that goes well with node

`Mongo DB`

You can use any database with node but this is the most popular and for reasons, you might find interesting

- It is fast
- Collections are just like plain javascript objects so you have an easier time wrapping your head around them

A document is made of many collections

Matter of factly speaking, data is stored like objects in non-relational databases and javascript engineers love objects

a collection has fields just like an object would have a field

```
{
  name: "Node JS",
  email: "nodejs@gmail.com",
  phone_no: 07012345678
  address: "start ng road, node, HNG"
}
```

the above is what a collection looks like

the `name`, `email`, `phone_no`, and `address` are fields on the collection

the above could be a user collection. That could be how you are stored in our database



Why don't we create our own database?

**@here** you will all be creating databases now

```
1). Install Mongo DB
- Open up your terminal
- Run npm install mongodb
```

React here once you have done so:

**5 replies**

```
create a folder or use the same folder you used for the previous class:
- Create a file called mongo_db.js
- Write var mongo = require('mongodb') at the top of this file;
```

React here once you do so:

**@here**

```
2) Now we create a database.
```

- To create a database in MongoDB, start by creating a MongoClient object, then specify a connection URL with the correct IP address and the name of the database you want to create.

MongoDB will create the database if it does not exist, and make a connection to it.

The above may seem like too much talk:

Let me show you in code:

Paste this in the file you just created.

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";
MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```

What does this do?

The MongoClient comes with MongoDB

It is an object.

I guess at this point, we all understand that that connect is a method on the MongoClient object **@here**

right?

connect does what it says, connect your app to MongoDB

connect takes in an error object and a callback

we talked about callbacks in the last class, it is a function that is passed into a function.

Run `node mongo_db.js`

What do you see on your terminal?

**6 replies**

[View thread](#)

@class run `mongo --version` and send a screenshot

### 9 replies

[View thread](#)

@class:

navigate here:

`C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe`

on your terminal

go to program files

click on Mongo DB

click on Server

click on 4.2

Click on bin

click on mongo

Do this and send a screenshot **@here**

### 8 replies

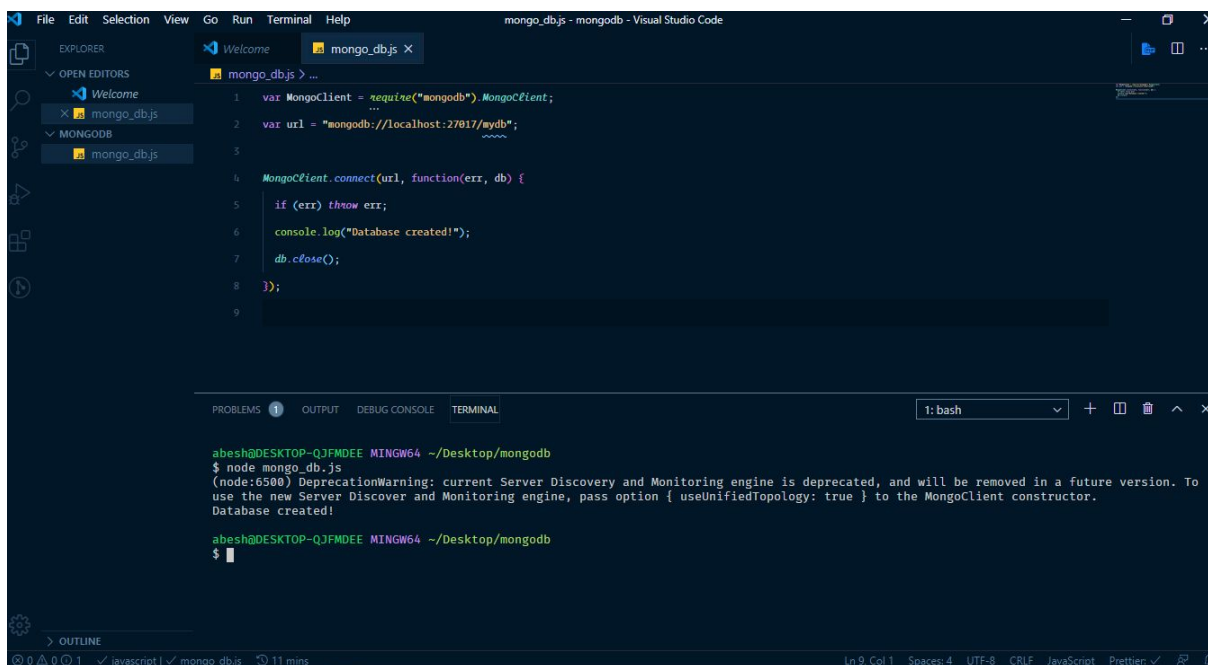
[View thread](#)

**@here** did you run `npm install MongoDB` as instructed?

### 3 replies

[View thread](#)

## Capture.PNG



The screenshot shows the Visual Studio Code interface with a file named `mongo_db.js` open. The code in the editor is as follows:

```
1 var MongoClient = require("mongodb").MongoClient;
2 var url = "mongodb://localhost:27017/mydb";
3
4 MongoClient.connect(url, function(err, db) {
5   if (err) throw err;
6   console.log("Database created!");
7   db.close();
8 });
```

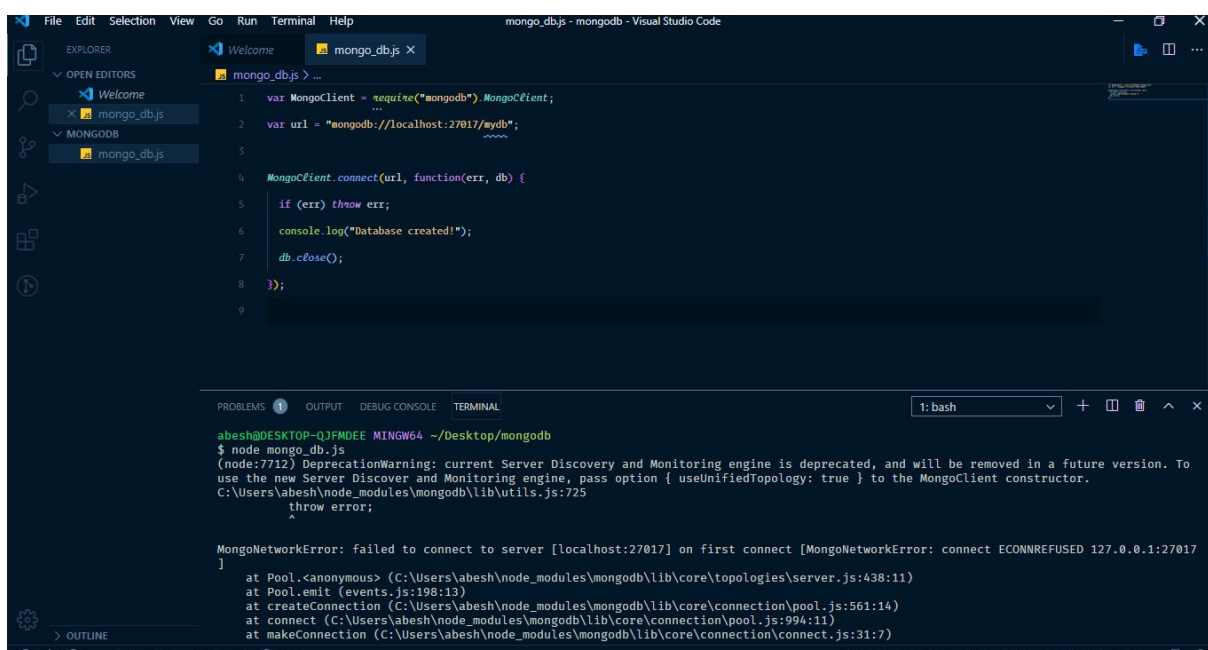
The terminal at the bottom shows the command `node mongo_db.js` being executed, which results in the output: `(node:6580) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor. Database created!`. The status bar at the bottom indicates the file is `mongo_db.js` and the editor is using `JavaScript` and `Prettier` formatting.

@here to get it running like that you need to have your local mongo db server running

Those who navigated to that path, can run `node mongo_db.js` and send a screenshot again

When you do not have your server running

## Capture.PNG

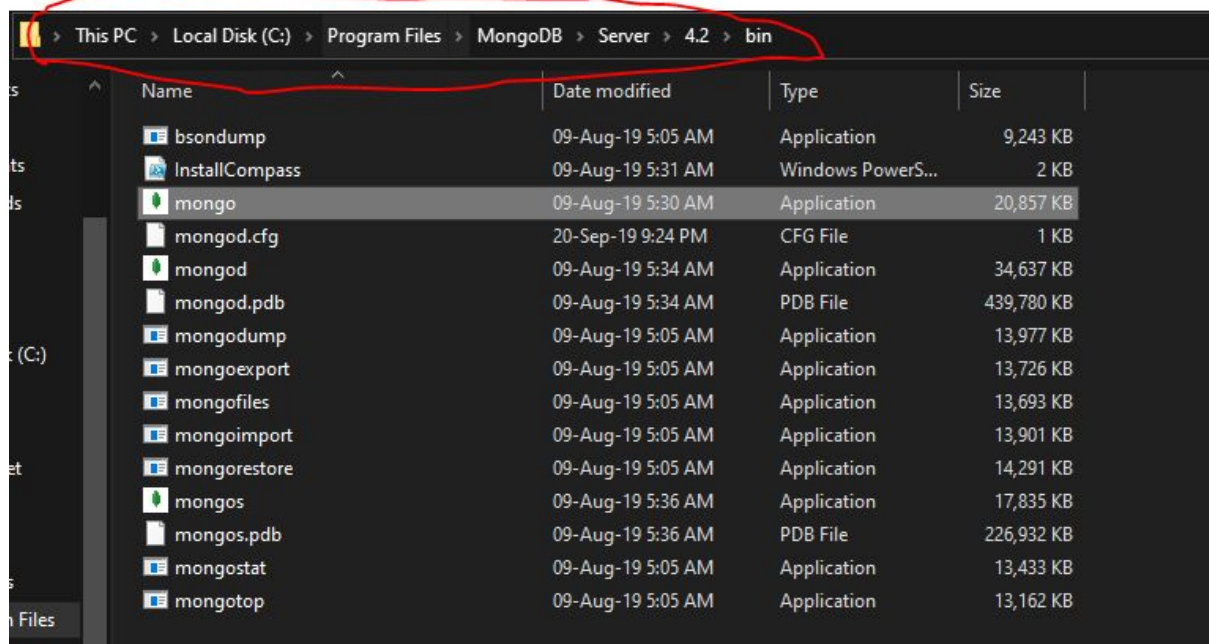


The screenshot shows the Visual Studio Code interface with the same file `mongo_db.js` open. The code is identical to the previous screenshot. The terminal at the bottom shows the command `node mongo_db.js` being executed, which results in the output: `(node:7712) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in a future version. To use the new Server Discovery and Monitoring engine, pass option { useUnifiedTopology: true } to the MongoClient constructor. C:\Users\abesh\node_modules\mongodb\lib\utils.js:725 throw error;`. Below this, a detailed error message is displayed: `MongoNetworkError: failed to connect to server [localhost:27017] on first connect [MongoNetworkError: connect ECONNREFUSED 127.0.0.1:27017]`. The status bar at the bottom indicates the file is `mongo_db.js` and the editor is using `JavaScript` and `Prettier` formatting.

@here send a screenshot if you have successfully run `node mongo_db.js`

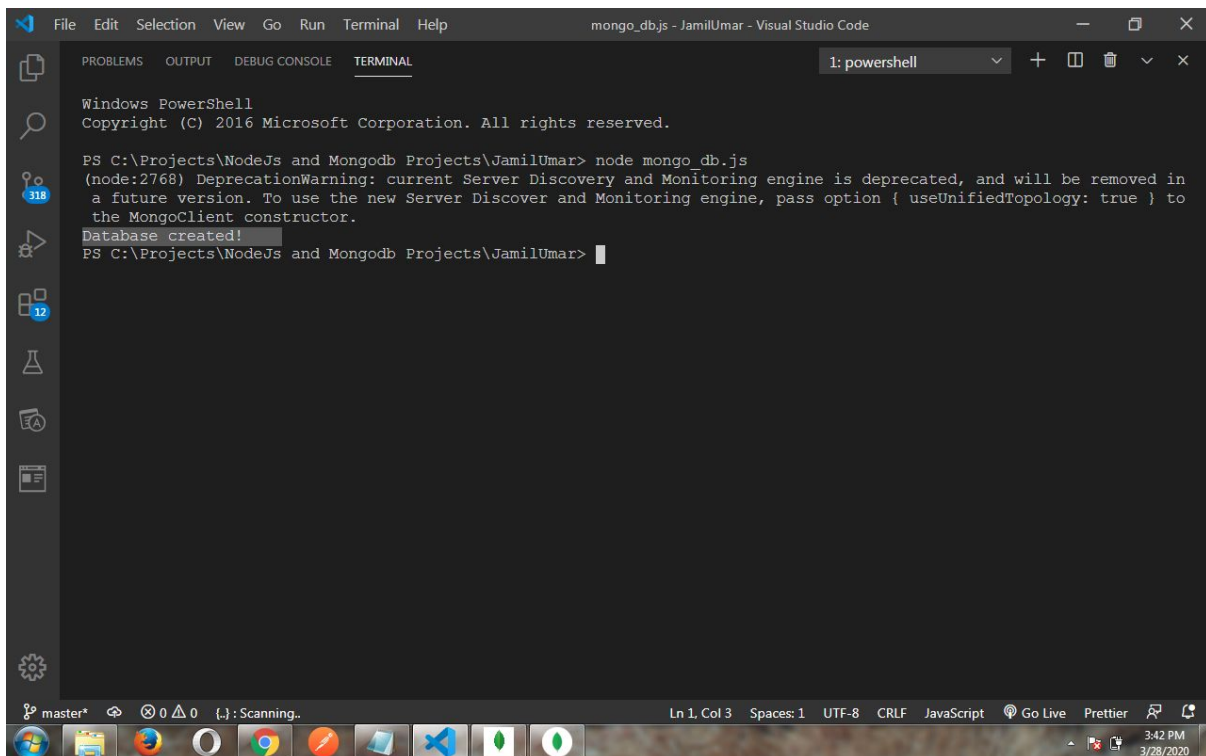
See where to find your server and start it for windows users

Capture.PNG



@here you should get this

mongodb\_1.png



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal is running a PowerShell session. The output shows the execution of a Node.js script named 'mongo\_db.js'. The script connects to a MongoDB instance at 'localhost:27017' and creates a database named 'mydb'. The output includes a deprecation warning about the Server Discovery and Monitoring engine and a confirmation message 'Database created!'.

```
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Projects\NodeJs and MongoDB Projects\JamilUmar> node mongo_db.js
(node:2768) DeprecationWarning: current Server Discovery and Monitoring engine is deprecated, and will be removed in
a future version. To use the new Server Discover and Monitoring engine, pass option { useUnifiedTopology: true } to
the MongoClient constructor.
Database created!
PS C:\Projects\NodeJs and MongoDB Projects\JamilUmar>
```

@JamilUmar Please explain how you got that for the class

The floor is open for you

## JamilUmar

First, install MongoDB:

```
npm install MongoDB
```

3:51

once install is complete put down the following code

```
var MongoClient = require('mongodb').MongoClient;
var url = 'mongodb://localhost:27017/mydb';
MongoClient.connect(url, function (err, db) {
  if (err) throw err;
  console.log('Database created!');
  db.close();
});
```

just run node mongo\_db.js which is the name of your file



if everything is successful you should get the message Database Created

else an error is thrown

done sir

**kingabesh**

@here you can install MongoDB server here

<https://www.mongodb.com/download-center/community>

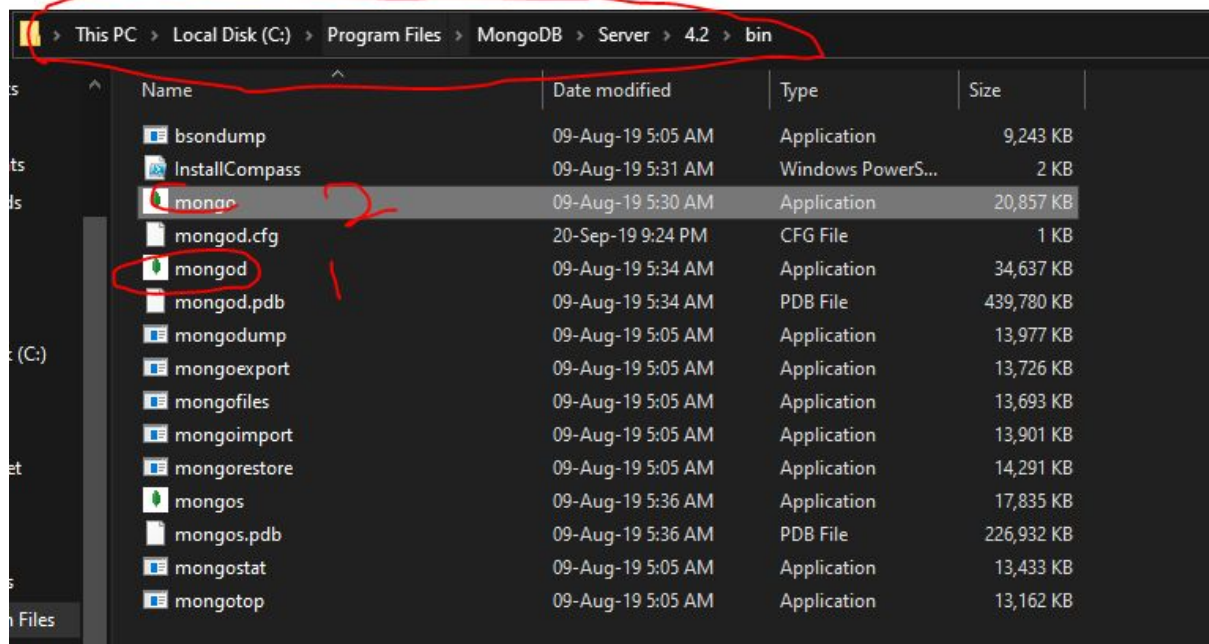
hit install the MongoDB server above first

then you can run npm install MongoDB in your project's directory

navigate to the mongod.exe and start it

then, you can optionally start mongo.exe

Capture.PNG



those are your local servers...

MongoDB is the server

mongo is just a terminal you can use

we will play there next class

The class is adjourned for now.

Please reach out to your fellow classmates for help on how to install the MongoDB server

Next class, we will be running database commands