Install MongoDB Driver

access a MongoDB database with Node.js.

Type:

npm install mongodb

mongodb database driver is downloaded and installed

Node.js can use this module to manipulate MongoDB databases:

## Creating 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.

var MongoClient = require('mongodb').MongoClient;

//Create a database named "mydb":

var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {

if (err) throw err;

console.log("Database created!");

db.close();

});

Save the code above in a file called "demo\_create\_mongo\_db.js" and run the file:

Run "demo\_create\_mongo\_db.js"

Introduction to Node.js and MongoDB:

Node.js is an open-source, server-side runtime environment built on Chrome's V8 JavaScript engine. It allows you to run JavaScript code outside of the web browser, making it a popular choice for building server-side applications. MongoDB, on the other hand, is a NoSQL database that stores data in a flexible, JSON-like format. Node.js and MongoDB work well together, as Node.js can be used to build the server-side logic for applications that interact with MongoDB.

Using Node.js packages in MongoDB:

Node.js has a rich ecosystem of packages (also known as modules or libraries) that can be easily integrated into your applications. You can use the Node Package Manager (NPM) to install these packages. In the context of MongoDB, you can use packages like "mongodb" to interact with the database from your Node.js application. These packages provide methods and APIs for connecting to MongoDB, performing CRUD (Create, Read, Update, Delete) operations, and more.

Accessing and manipulating databases and collections:

To access and manipulate databases and collections in MongoDB from your Node.js application, you need to follow these general steps:l

1. Install the MongoDB Node.js driver: Use NPM to install the "mongodb" package, which provides the necessary tools for connecting to MongoDB from Node.js.

**npm install mongodb**

1. Import the MongoDB driver in your Node.js code:

**const MongoClient = require('mongodb').MongoClient;**

1. Connect to the MongoDB server:

**const url = 'mongodb://localhost:27017'; // Replace with your MongoDB server URL**

**MongoClient.connect(url, (err, client) => {**

**if (err) {**

**console.error('Error connecting to MongoDB:', err);**

**return;**

**}**

**const db = client.db('mydatabase'); // Replace with your database name**

**// You're now connected to the database and can perform operations.**

**});**

CRUD operations using Node.js in MongoDB:

1. **Create (Insert)**:

const collection = db.collection('mycollection');

const dataToInsert = { name: 'John', age: 30 };

collection.insertOne(dataToInsert, (err, result) => {

if (err) {

console.error('Error inserting document:', err);

return;

}

console.log('Document inserted:', result.ops[0]);

});

1. **Read (Query)**:

const collection = db.collection('mycollection');

collection.find({ name: 'John' }).toArray((err, docs) => {

if (err) {

console.error('Error querying documents:', err);

return;

}

console.log('Documents found:', docs);

});

1. **Update**:

const collection = db.collection('mycollection');

collection.updateOne({ name: 'John' }, { $set: { age: 31 } }, (err, result) => {

if (err) {

console.error('Error updating document:', err);

return;

}

console.log('Document updated');

});

**Delete**

const collection = db.collection('mycollection');

collection.deleteOne({ name: 'John' }, (err, result) => {

if (err) {

console.error('Error deleting document:', err);

return;

}

console.log('Document deleted');

});

### Running a Node.js File in Ubuntu Terminal:

Here's how to run a Node.js file in the Ubuntu terminal:

1. Open the Ubuntu terminal.
2. Navigate to the directory where your Node.js file is located using the cd command.
3. Run your Node.js file using the node command followed by the file name. For example:

**node yourfile.js**