Table of Contents

[Day03 Installing Node, MongoDB 2](#_Toc19085600)

[**PART 1 – INSTALLING NODEJS** 2](#_Toc19085601)

[**PART 2 – INSTALLING MONGODB** 4](#_Toc19085602)

[**Part 3 – Creating MongoDB Databases and Collections** 5](#_Toc19085603)

[**Part 4 – Integrating NodeJS and MongoDB** 6](#_Toc19085604)

[**Appendix A** 15](#_Toc19085605)

# Day03 Installing Node, MongoDB

**(on Ubuntu 18.04 Stable)**

## PART 1 – INSTALLING NODEJS

This part of the day is all about installation and configuration **MongoDB** and **NodeJS**. In all cases, we will be working with **Ubuntu 18** and the latest software of these three technologies.

*Before attempting to install NodeJS on Ubuntu 18, please make sure that your version does not already have node installed, as some versions of Ubuntu come with node pre-installed.*

|  |  |
| --- | --- |
| Installing nodejs on Ubuntu 18  **These are the commands we will be entering (in order) to get node.js installed**  Install curl  **sudo apt install curl**  Admins use cURL to request or send data to a URL. Its good for reporting if another service is up and how healthy the service is.  Use curl to make sure we are downloading to a stable node PPA (personal package archive)  **curl -sL https://deb.nodesource.com/setup\_10.x | sudo bash -**  [*s is silent, L will redirect the request in the event that the server reported a change of location, the sudo bash at the end is instructing the computer to run the commands it just got from the server* ]   |  | | --- | | If you want to see what this file looks like, you can run this command:  **curl -sL** [**https://deb.nodesource.com/setup\_10.x -o Downloads/temp.txt**](https://deb.nodesource.com/setup_10.x%20-o%20Downloads/temp.txt)  Then open the file with any editor for example: **gedit temp.txt** |   With the PPA downloaded, we can then run the command to install nodejs itself:  **sudo apt-get install -y nodejs**   Check the version of node  **node –v (**should return v10.10.0**)**   Check the version of npm, which is installed automatically  **npm –v (**should return 6.4.1 or newer**)**   Check the installation by running the following command:  **sudo node http\_server**  This should return something like*Debugger listening on ws://127.0.0.1:9229/a...*  In order to see a more elaborate example of a server type this code into a text file and then save it as http\_server.js  **const http = require('http'); //  http.createServer(function (req, res) {  console.log('Hello from Skillsoft');  res.end('Hello from Skillsoft');  }).listen(8000); //  console.log('Server listening on port 8000');**  Now start the server by typing in node **http\_server** from a terminal window |

## PART 2 – INSTALLING MONGODB

|  |
| --- |
| **Installation of MongoDB on Ubuntu**   1. Install a stable MongoDB package   **sudo apt install -y mongodb**   1. Check the service:   **sudo systemctl status mongodb**  you should see something like this:   1. As a triple check run this command to se if you get a response:   **mongo –eval 'db.runCommand({connectionStatus: 1})'** |

## Part 3 – Creating MongoDB Databases and Collections

Before proceeding, delete any Weights database that is currently in the system. Use a terminal window. Do a show dbs and all databases will be shown. To delete a db, just use it then issue the command **db.dropDatabase()**.

In order to get into the MongoDB shell, use the command **sudo mongo**

1. Change the database to Weights and create a new table using the following code:

|  |
| --- |
| **use Weights;** |

1. Add a collection

|  |
| --- |
| **db.createCollection(“EmployeeWeights”)** |

1. Perform a find(), it should not return anytihng but at least we know we now have a database and a collection

|  |
| --- |
| **db.EmployeeWeights.find()** |

1. Enter a record

|  |
| --- |
| **db.EmployeeWeights.insertOne( {empName : “Joe”,  empWeight : 95.6 })** |

1. Verify the record.

|  |
| --- |
| **db.EmployeeWeights.find()** |

1. Add another record by using the up arrow key and just changing the name and weight

|  |
| --- |
| **db.EmployeeWeights.insertOne( {empName : “mary”,  empWeight : 65.9 })** |

1. Verify the new record

|  |
| --- |
| **db.EmployeeWeights.find()** |

1. Lets change (update) Joe’s record:

|  |
| --- |
| **db.EmployeeWeights.update(**  **{empName : "Joe"},**  **{$set: {empWeight : 96.5 } }**  **)** |

1. Verify the change

|  |
| --- |
| **db.EmployeeWeights.find()** |

1. Enter a new document but this one will have a date in addition to the name and weight

|  |
| --- |
| **db.EmployeeWeights.insertOne(**  **{**  **empName : "Sally",**  **empWeight : 65.9,**  **Date : new Date()**  **}**  **)** |

1. Verify the change but this tiime chain the pretty() method

|  |
| --- |
| **db.EmployeeWeights.find().pretty()** |

1. Finally update Joes’s record to include a date and then do a find pretty

|  |
| --- |
| **db.EmployeeWeights.update (   {empName : "Joe"},    {$set: {Date : new Date() } },  false, false )** |

## Part 4 – Integrating NodeJS and MongoDB

1. Make sure that you are in an appropriate folder, for example Day03/MongoTest.

Run **npm init** within that folder to create a package.json file. Accept the defaults except for name, use http\_server.js.

|  |
| --- |
| **npm init** |

1. Run **npm install mongodb** to install the mongodb package.

|  |
| --- |
| **npm install mongodb** |

1. We create a server file, using touch, inside the folder eg http\_server.js.

Enter the connection code below:

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **mdb.close();**  **});** |

1. It In the next block of code, attempt to connect to the collection

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **//**  **let wdb=mdb.db("Weights");**  **wdb.collection("EmployeeWeights").findOne(**  **{"empName":"Joe"},**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

Note: in the code above, we can change this line to show the weight for joe eg **console.log(result.empWeight);**

1. Try to find all the documents in the collection. You may think that just executing the **find()** method without any name value pairs may work, but it doesent

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(url, {useNewUrlParser:true}, function(err, db){**  **if(err) throw err;**  **let wdb=db.db("Weights");  wdb.collection("EmployeeWeights").find(  {},   function(err, result){**  **if(err) throw err;**  **console.log(result);**  **});**  **db.close();**  **} );** |

Well it returns then entire server structure, but in the middle of all that data is our weights informtion, we just need to extract it, see below.

1. Just chain the **toArray()** method onto the **find()** method, then the same function we had before will be the parameter to the **toArray()** method.

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **//**  **let wdb=mdb.db("Weights");**  **wdb.collection("EmployeeWeights").find(**  **{}).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

1. Lets now try to insert a new document using our server js file. We will use the **insertOne()** method, but first create a variable to represent the json object that will represent our new document.

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(url, {useNewUrlParser:true}, function(err, mdb){**  **if(err) throw err;**  **let wdb=mdb.db("Weights");**  **let newEmployee = {empName:"Johan", empWeight: 86.7};**  **wdb.collection("EmployeeWeights").find({}).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **} );** |

1. We will leave the **find()** method in place as we would want to verify that Johan was inserted, but we can copy and past the **find()** method and replace find with **insertOne()**

|  |
| --- |
| **function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **let wdb=mdb.db("Weights");**  **//**  **let newEmployee = {empName:"Johan", empWeight: 86.7};**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(error, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **wdb.collection("EmployeeWeights").find(**  **{}).toArray(** |

Here is the entire code so far:

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **let wdb=mdb.db("Weights");**  **//**  **let newEmployee = {empName:"Johan", empWeight: 86.7};**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(error, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **wdb.collection("EmployeeWeights").find(**  **{}).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

1. Insert a record with a **date** field

|  |
| --- |
| **if(err) throw err;**  **let wdb=db.db("Weights");**  **let newEmployee = {**  **empName:"** **Harry",**  **empWeight: 51.5,**  **date: new Date(Date.now()).toISOString()**  **};**  **wdb.collection("EmployeeWeights").insertOne(newEmployee, function(err, result){** |

When you execute the http\_server file, you see Harry is inserted with the current date.

Here is the entire http\_server.js file so far:

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(url, {useNewUrlParser:true}, function(err, mdb){**  **if(err) throw err;**  **let wdb=mdb.db("Weights");**  **let newEmployee = {**  **empName:"Harry",**  **empWeight: 96.4,**  **date: new Date(Date.now()).toISOString()**  **};**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **wdb.collection("EmployeeWeights").find({}).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

**Optional**

1. Perform a search with a criteria.  
   we can modify the current code to create a search term

|  |
| --- |
| **let wdb=mdb.db("Weights");**  **//**  **let searchTerm = {**  **empName:"Harry"**  **};**  **//**  **wdb.collection("EmployeeWeights").findOne(**  **searchTerm,**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//** |

Here is the entire code

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **let wdb=mdb.db("Weights");**  **//**  **let searchTerm = {**  **empName:"Harry"**  **};**  **//**  **wdb.collection("EmployeeWeights").findOne(**  **searchTerm,**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

**find()** will return an array so we have to chain the **toArray()** method onto find and deal with any error or result:

|  |
| --- |
| **let searchTerm = {**  **empName:"Harry"**  **};**  **//**  **wdb.collection("EmployeeWeights").find(searchTerm).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);** |

Entire file using **find(),** without any criteria to search for

|  |
| --- |
| **const mClient = require('mongodb').MongoClient;**  **const url = "mongodb://localhost:27017/";**  **mClient.connect(**  **url,**  **{useNewUrlParser:true, useUnifiedTopology: true}, function(err, mdb){**  **if(err) throw err;**  **console.log("Connected to mongodb");**  **let wdb=mdb.db("Weights");**  **//**  **let searchTerm = {**  **empName:"Harry"**  **};**  **//**  **wdb.collection("EmployeeWeights").find(searchTerm).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);**  **//**  **mdb.close();**  **});** |

1. This code will attempt to read the result of having made an insert, notice that it is a lot of information

|  |
| --- |
| **let newEmployee = {**  **empName:"Steve",**  **empWeight: 106.7,**  **dateAdded:new Date(Date.now()).toISOString()**  **};**  **//**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **}**  **);** |

But if you want to get just the count, change result to **result.insertedCount**

1. This code will return the entire document that was inserted

|  |
| --- |
| **let newEmployee = {**  **empName:"Jane",**  **empWeight: 69.7,**  **dateAdded:new Date(Date.now()).toISOString()**  **};**  **//**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(err, result){**  **if(err) throw err;**  **console.log(result.ops[0]);**  **}**  **);**  **//** |

1. This will return just the name:

|  |
| --- |
| **let newEmployee = {**  **empName:"Claude",**  **empWeight: 93.8,**  **dateAdded:new Date(Date.now()).toISOString()**  **};**  **//**  **wdb.collection("EmployeeWeights").insertOne(**  **newEmployee,**  **function(err, result){**  **if(err) throw err;**  **console.log(result.ops[0].empName);**  **}**  **);** |

1. This code will update a record and display the full list again

|  |
| --- |
| **mClient.connect(url, {useNewUrlParser:true}, function(err, mdb){**  **if(err) throw err;**  **let wdb=mdb.db("Weights");**  **let updateQuery = { empName: "Jane" };**  **let newWeight = { $set: {empWeight: 68.4} };**  **//**  **wdb.collection("EmployeeWeights").updateOne(**  **updateQuery,**  **newWeight,**  **function(err, result){**  **if(err) throw err;**  **console.log("Document updated");**  **}**  **);**  **//**  **wdb.collection("EmployeeWeights").find({}).toArray(**  **function(err, result){**  **if(err) throw err;**  **console.log(result);**  **});**  **//**  **mdb.close();** |

Or to be sure use c**onsole.log(result.matchedCount);**

## Appendix A

MongoDB delete commands

**db.collection.remove()**

Delete a single document or all documents that match a specified filter.

**db.collection.deleteOne()**

Delete at most a single document that match a specified filter even though multiple documents may match the specified filter.

**db.collection.deleteMany()**

Delete all documents that match a specified filter.