COMP306: DATABASE MANAGEMENT SYSTEMS

Installation Guide for Nodejs on Ubuntu, Windows and Mac

How to install Nodejs on Mac with Homebrew

- If you don’t have homebrew, you can download via terminal
  - Copy and paste this command: */bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"*
  - Check your installation of homebrew in terminal. Write *brew help* to terminal. If brew is there, you get output. If not, you get 'command not found'.
- If you do have homebrew
  - Copy and paste this command: *brew install node*

How to install Nodejs on Ubuntu with Terminal

Please run the following commands in your terminal

- sudo apt update
- sudo apt install nodejs
- sudo apt install npm

How to install Nodejs from website

Please visit official site of Nodejs

- https://nodejs.org/en/download
  - Windows
  - Ubuntu
  - Mac

Download the installer of nodejs for your operating system and follow the installation guides.

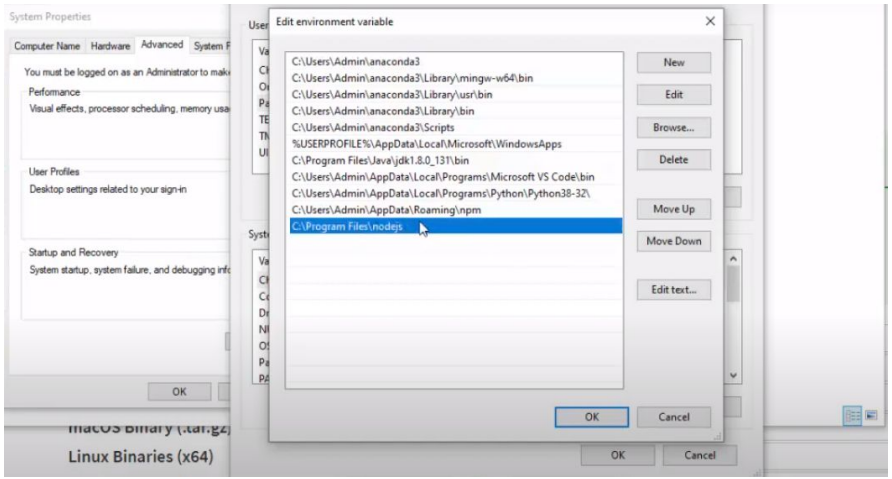
Check If Nodejs is Correctly Installed

Write ‘*node -v’* to terminal. If you see the version in the output, you have successfully installed nodejs.

- For some Windows users: Even if you installed nodejs correctly, because of PATH problems, terminal may not understand the ‘*node*’ keyword.

Environment Setup for Windows

- Write ‘Edit the system environment variables’
- Click the ‘Environment Variables…’ button on the bottom-right corner
- In the first table, find ‘Path’ variable row and click ‘Edit…’ button.
- Add ‘C:\Program Files\nodejs\’ (or the folder you installed nodejs into)



IDE Recommendation

- We recommend Visual Studio Code for IDE, but any other IDE will do the work (e.g., WebStorm). You can download Visual Studio Code from https://code.visualstudio.com/download
- Create a sample project folder
- Open the terminal and go to your project folder
- From the terminal, type '*npm init -y*' (-y tag means you accepted the default setting). You should see the package.json file in your project folder.

Packages We Need

- MySQL
  - MySQL package is needed to make a connection between Nodejs and MySQL
- Express.js
  - Express.js, or simply Express, is a back-end web application framework for building RESTful APIs with Nodejs
- Cors
  - CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.
- Body-Parser
  - Parse incoming request bodies in a middleware before your handlers, available under the req.body property.

From the terminal, type ‘npm install express mysql cors body-parser’. With this command, you will install the above packages, which you will use in the PS and Homework 3.

Demo

Let’s use our tools in a sample project.

In your project folder, create an empty javascript file named index.js (js stands for javascript). In order to import the packages, you need to paste these lines at the top of your index.js file:

```
1  const express = require('express');
2  const mysql = require('mysql');
```

MySQL Connection

Now, we will connect to our localhost server from javascript. But first, we need to tell javascript about our MySQL server credentials. Paste the following lines to your index.js:

```
const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: 'root',
});
```

These are my credentials, **your username and password may be different**. In default for MySQL, username is ‘root’ and there is no password set.

Testing the Connection

err stands for error – it will tell you if you have any error when you connect to your database.

- The if statement below is to see the error in the console (terminal)
- If there's no error, we should see 'Connected to database' line in the console

Let's run our file. Again open the terminal, and from the terminal, navigate to your project folder. Then type *node index.js*. You should see 'Connected to database' line if you don't have any errors.

```
db.connect((err) => {
    if (err) {
        console.log(err);
    }
    console.log('Connected to database');
});
```

Express.js

For web applications, you shouldn’t directly call these functions from your webpage. That’s why we need RESTfull services like Express.js. Let’s create our expressJS app with:

```
const app = express();
```

Send SQL Query to Database

In order to get “GET REQUEST” from our webpage, we need to use app.get() function from expressJS. Paste this code block:

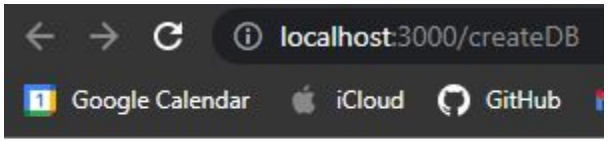
```
app.get('/createDB', (req, res) => {
    let sql = "CREATE DATABASE IF NOT EXISTS nodejs";
    db.query(sql, (err, result) => {
        if(err){
            console.log(err);
        }
        res.send('Database created');
    });
});
```

This code block will create a database in your MySQL server. ‘Res’ stands for a response. When your webpage (e.g., index.js) uses “GET REQUEST” to your server, it will get ‘Database created’ information from the server. But also, your server should listen to requests from a port number, which you can achieve using this code:

```
app.listen('3000', () => {
    console.log('Server started on port 3000');
});
```

Check the Output

Now run your server (with the command: node index.js) and open a browser and type ‘http://localhost:3000/createDB’ into the search bar. You should see something like this.



You can also check via your MySQL terminal or Workbench to verify that a database named “nodejs” now exists in your MySQL.