

# Create, modify and run Node files

## Welcome to this lab activity

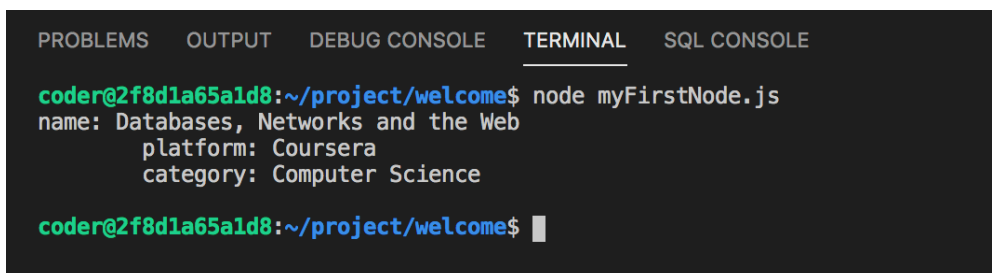
In our previous lab activity we explored how to start a specific lab and the core user interface of the Visual Studio Code IDE. It is now the time for you to write some code through a guided exercise. The main focus here is for you to get familiar with the explorer, terminal and editor sections of Visual Studio Code. By the end of this exercise, you should be able to create your own project folder structure, add and modify JavaScript files and run them using the Node.js runtime environment.

## The exercise overview

The goal of this exercise is to create a very simple program that prints to the terminal all the information regarding a particular object in the form of key/value pairs. More specifically we will initialise a *course* object with the following properties:

```
let course = {  
  name: "Databases, Networks and the Web",  
  platform: "Coursera",  
  category: "Computer Science"  
};
```

The aim is to print all the key/value pairs inside the *course* object to the terminal with the help of a call to a custom *printCourseInformation(course)* function. The function will accept the *course* object as a parameter and will print all the information to the terminal as shown below:

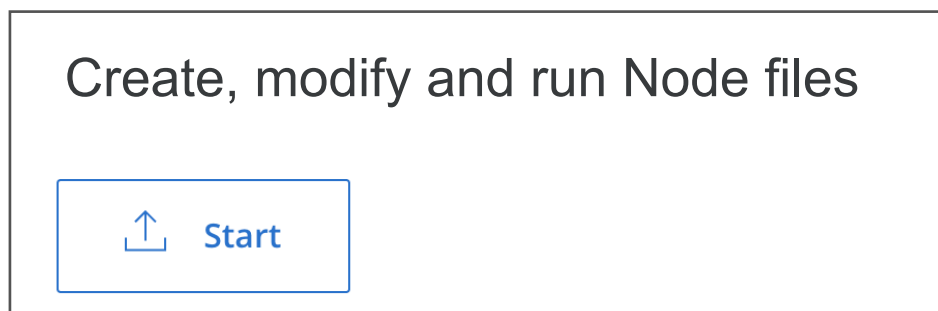


```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE  
coder@2f8d1a65a1d8:~/project/welcome$ node myFirstNode.js  
name: Databases, Networks and the Web  
  platform: Coursera  
  category: Computer Science  
coder@2f8d1a65a1d8:~/project/welcome$
```

## Start the Lab environment application

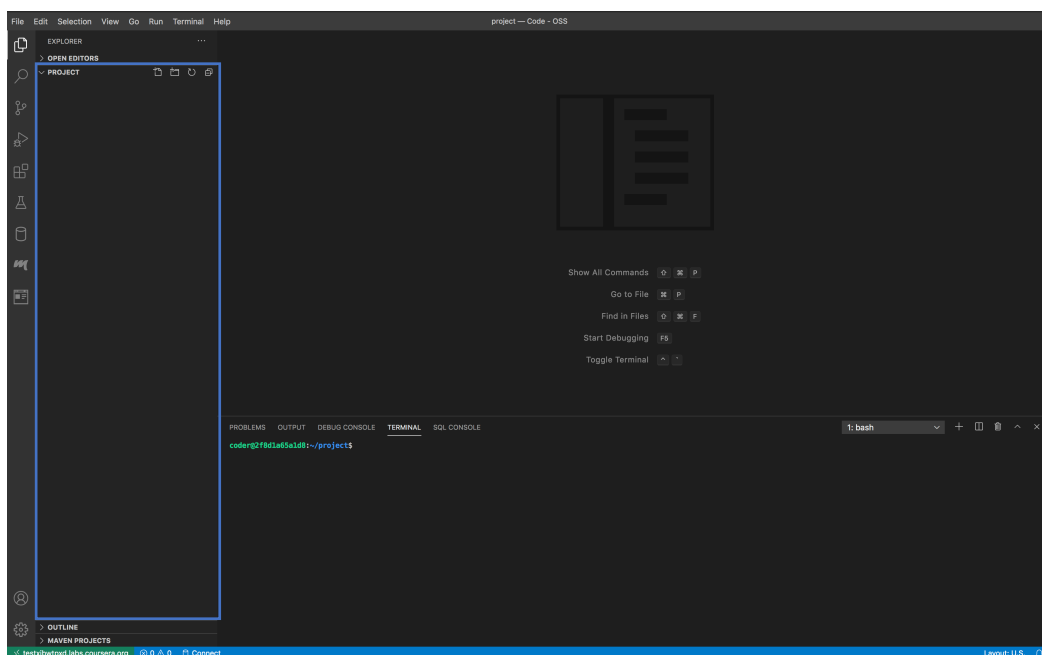
It is simple to launch a lab exercise. You only need to click on the button “Start” below the activity title to enter this lab environment.

Let’s explore this lab activity. Go ahead and click on the “Start” button!



## How to organise the project folder structure

The first step would be for you to create the folder structure for this exercise. Go ahead and close the Visual Studio “welcome” tab at the top of the editor panel. Furthermore, enable the Terminal panel by clicking on the “View” tab, then the “Terminal” tab (check your previous lab session for these two steps). All of your folders and files are listed and can be managed from the explorer project section in Visual Studio Code:



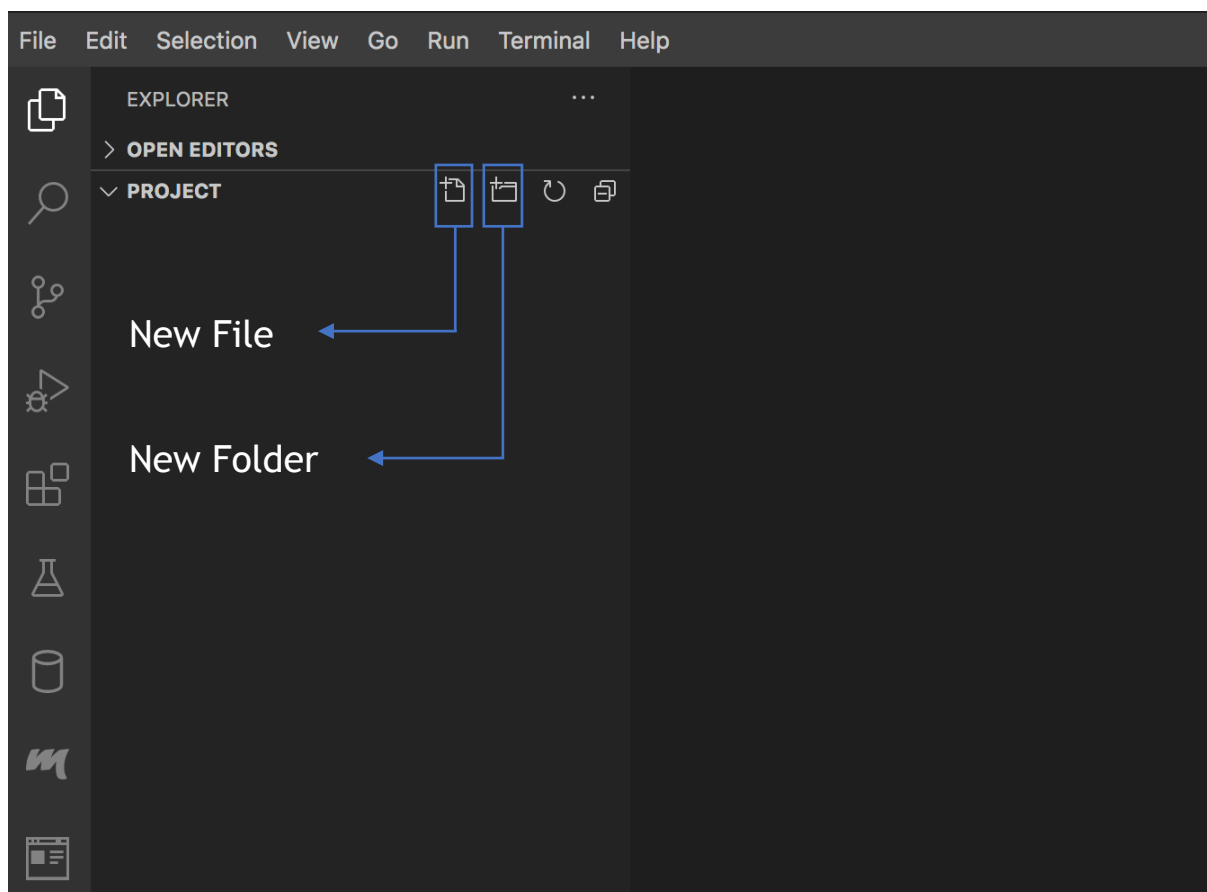
## How to create folders and files

The project structure for this exercise is straight forward. You will create a folder called *welcome* at the root of the explorer project section which will contain a single JavaScript file called *myFirstNode.js*.

There are different ways to create folders and files in Visual Studio Code; you can either take advantage of the explorer project section GUI or use the Terminal to do so. We will explore both ways.

## How to create folders and files: The Explorer project GUI

The Explorer project GUI makes it really easy to create folders and files:



- Create a folder called *welcome* (click on the “New Folder” icon)
- Create a file called *myFirstNode.js* inside the *welcome* folder (click on the “New Folder” icon)

## How to create folders and files: The Terminal panel

You can also use the Terminal panel to create folders and files. Although it might appear less straight forward and involves more steps, knowing how to use the Terminal is a skill that is in demand, especially for web development.

Type the following commands inside the Terminal panel and press *Enter* on your keyboard to run them:

**mkdir welcome:** type this command and press *Enter*. This command will create a folder called *welcome* inside your root directory.

**cd welcome/:** type this command and press *Enter*. This command will change your working directory to be the *welcome* folder just created.

**touch myFirstNode.js:** type this command and press *Enter*. This command will create a JavaScript file called *myFirstNode.js* inside the *welcome* folder.



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

coder@2f8d1a65a1d8:~/project$ mkdir welcome
coder@2f8d1a65a1d8:~/project$ cd welcome/
coder@2f8d1a65a1d8:~/project/welcome$ touch myFirstNode.js
coder@2f8d1a65a1d8:~/project/welcome$
```

# How to add content to files and save them

Adding content to a file using the built in Visual Studio Code GUI is extremely easy. Let's add some code to the *myFirstNode.js* file:

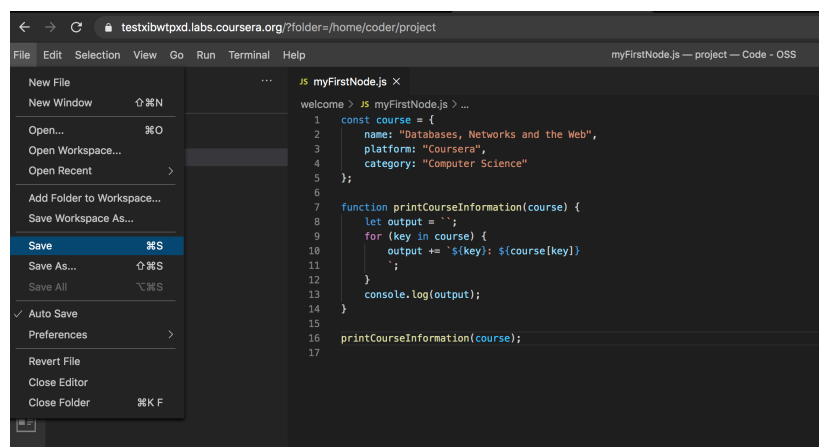
- Select the *myFirstNode.js* file in the “Explorer” project section
- Copy the following code:

```
const course = {
  name: "Databases, Networks and the Web",
  platform: "Coursera",
  category: "Computer Science"
};

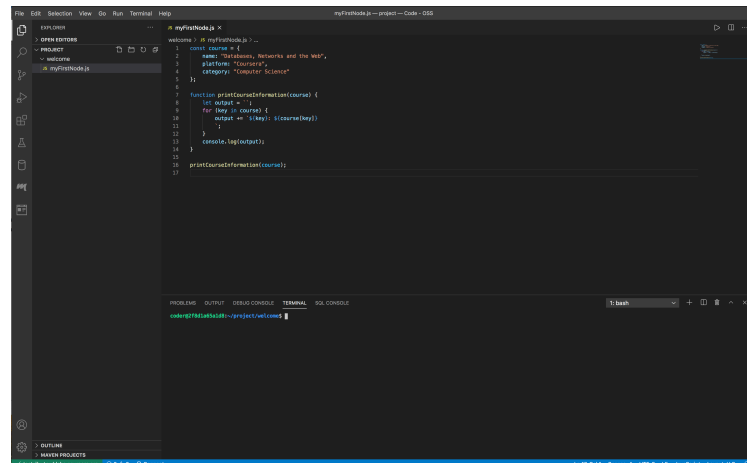
function printCourseInformation(course) {
  let output = ``;
  for (key in course) {
    output += `${key}: ${course[key]}
  `;
  }
  console.log(output);
}

printCourseInformation(course);
```

- Paste it inside the “Editor” section of the *myFirstNode.js* file
- The editor automatically saves your files but you can manually save them: select the *myFirstNode.js* file, click the “File” tab, then the “Save” tab:



Once you have completed all the above steps, your final result should look like the picture below:



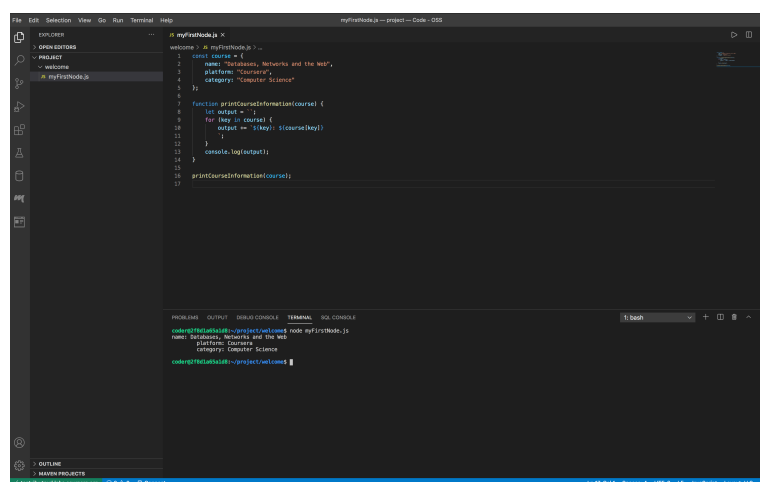
## How to run a JavaScript file with Node.js

In order to execute a JavaScript file using Node.js, you need to type the following command in the Terminal panel:

- **node myFirstNode.js:** type this command and press *Enter*. The `node` command, followed by the file name, tells Node.js to execute the content of the file.

The `node` command only works with JavaScript files located in the same working directory as where the command was executed.

Running the above command will produce the following output:



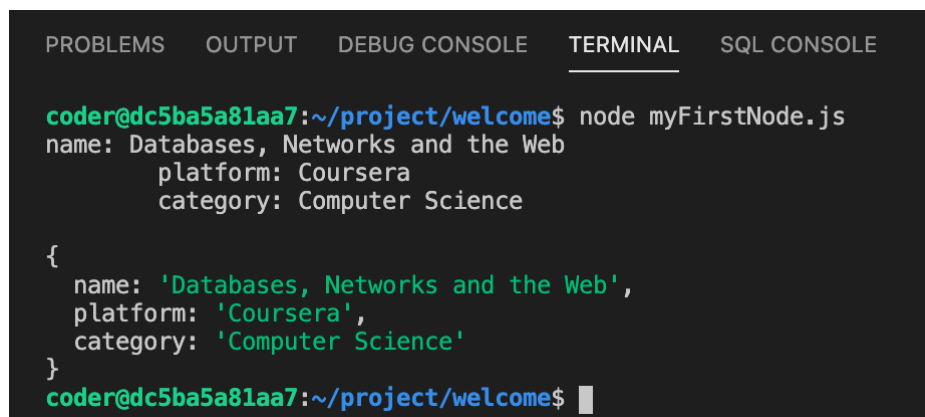
You can see that the Terminal panel logged out the information about the `course` object in the form of key/value pairs.

The “console.log” command allows you to quickly debug and output data about your running application.

Add the following line at the end of the “myFirstNode.js” script:

**console.log(course);**

Now run your application again and you should see the following output:

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'SQL CONSOLE'. The terminal shows a command prompt 'coder@dc5ba5a81aa7:~/project/welcome\$' followed by the command 'node myFirstNode.js'. The output of the command is displayed in three lines: 'name: Databases, Networks and the Web', 'platform: Coursera', and 'category: Computer Science'. Below this, a JSON object is printed: '{ name: 'Databases, Networks and the Web', platform: 'Coursera', category: 'Computer Science' }'. The prompt returns to 'coder@dc5ba5a81aa7:~/project/welcome\$' with a cursor at the end.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

coder@dc5ba5a81aa7:~/project/welcome$ node myFirstNode.js
name: Databases, Networks and the Web
platform: Coursera
category: Computer Science

{
  name: 'Databases, Networks and the Web',
  platform: 'Coursera',
  category: 'Computer Science'
}
coder@dc5ba5a81aa7:~/project/welcome$
```

As you can see, the “console.log(course);” printed out the content of the variable “course”.

Get in the habit of using “console.log” to quickly debug your lab applications and print out useful information.

## Clear the Terminal panel logs

You can clear your Terminal panel logs by typing the following command in the Terminal:

- **clear:** type this command and press *Enter*. This command will clear your Terminal logs.

# How to ask for assistance

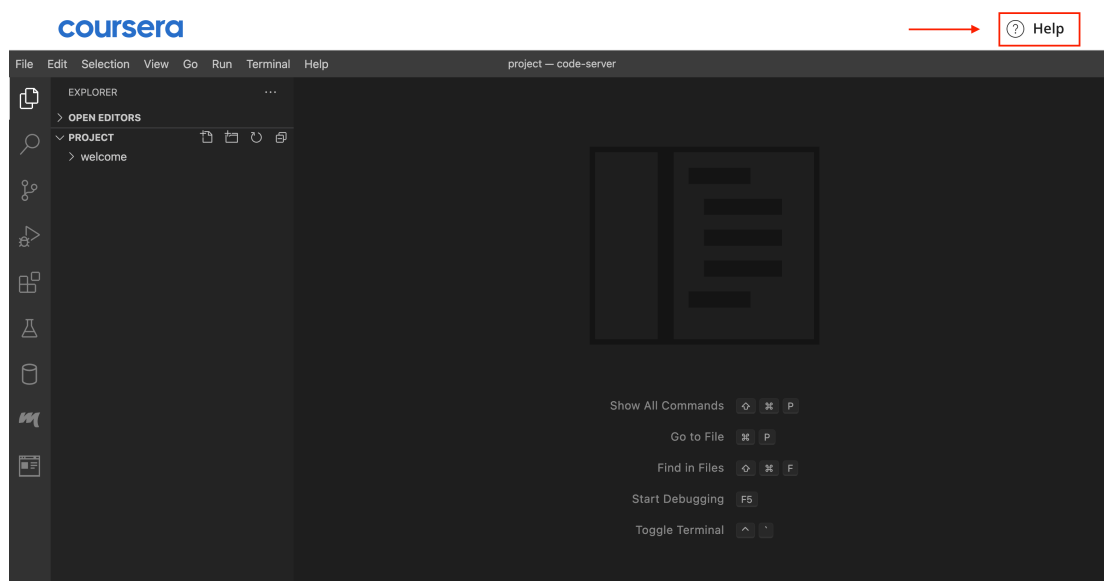
Use the discussion forum if you get stuck on a particular lab assignment.

**Make sure you do not share your lab URL with other students. Revealing your lab URL will allow others to access and potentially modify your lab environment.**

If you do not receive the help you hoped for in a discussion forum thread, you can share your lab ID with the course instructors so that they can access your lab and help you debug your issues. The lab ID can only be used to access your lab environment by the course instructors and it is therefore safe for you to share on the discussion forum.

Here is how you can obtain the lab ID of a particular lab environment:

With your lab environment open, click on the “Help” button located on the top right corner as shown on the picture below:

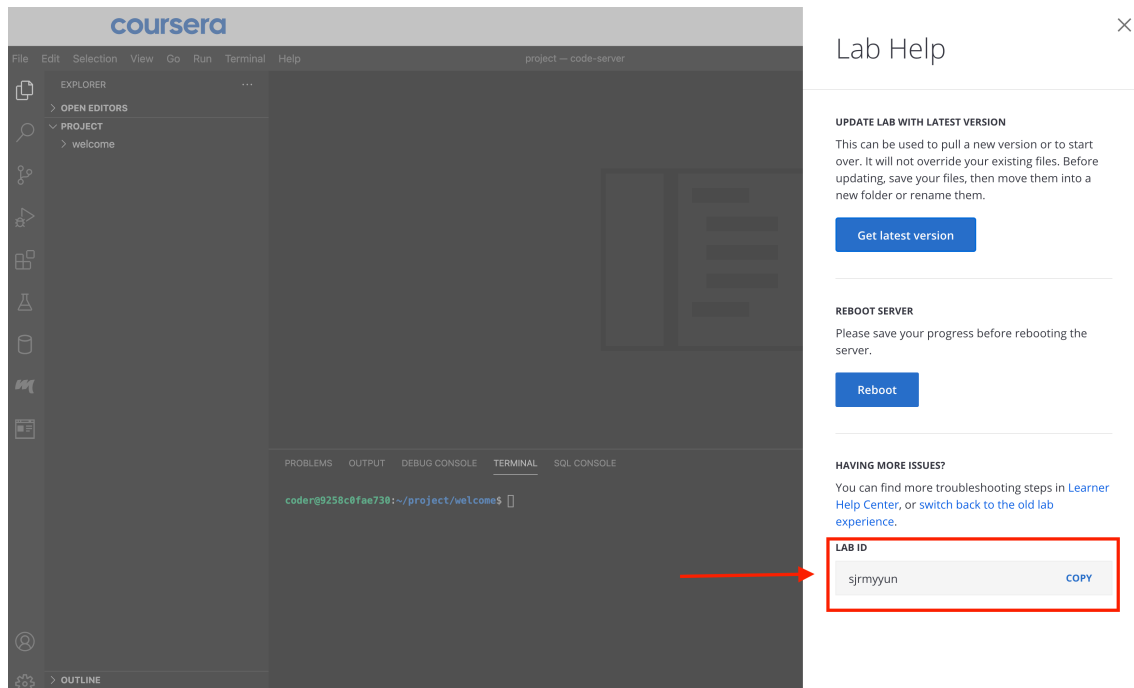


This will open a side menu on the right hand side of the lab environment with the LAB ID as shown on the image below:

Simply copy the LAB ID and share it on a discussion forum to ask for assistance.

Finally the menu also contains additional links to troubleshoot your issues as shown on the image below:



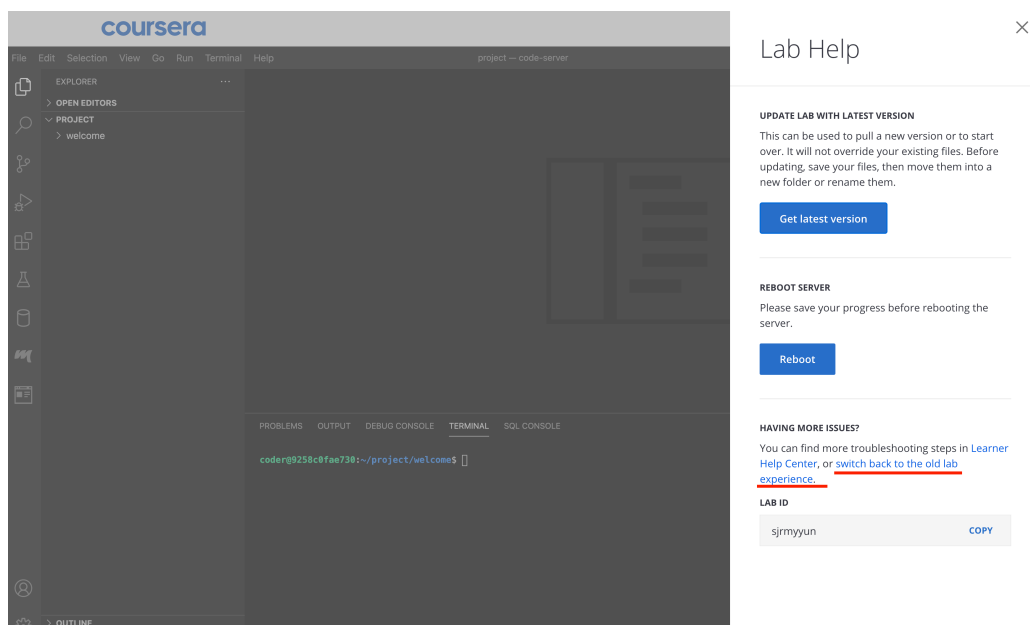


## How to backup your work

It is always a great idea to backup your work in case something goes wrong and we all know that things do go wrong from time to time!

We have provided you with a simple way to download your project folder for every lab activity.

**Note:** This feature only works with the **old lab experience**. If you wish to download a copy of your work you will have to first switch your lab to the old experience. You can do so by clicking on the “switch back to the old lab experience” link inside the “Help” menu as shown on the image below:

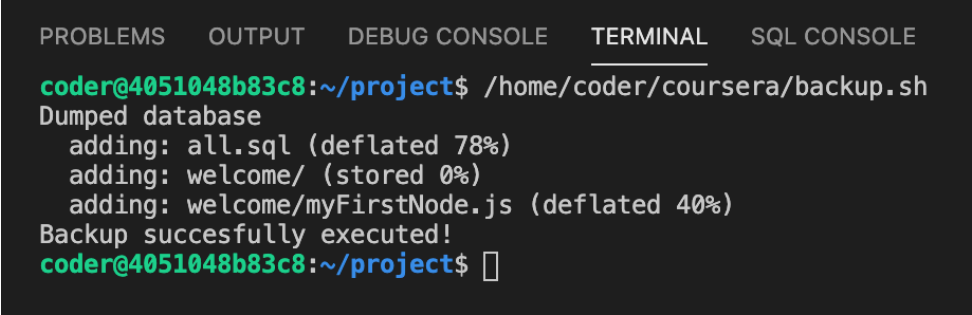


**Do not worry as the latest lab experience will be restored in future accesses.**

At this point, your lab will reboot and you will not notice major differences. What changes is that the “Help” button on the top right corner will not be visible (this is how you can tell if you are in the old lab experience or not).

Type the following commands inside the Terminal panel and press *Enter* on your keyboard to run them:

**/home/coder/coursera/backup.sh**: type this command and press *Enter*. This command will create a zip backup file called *backup.zip* inside your root directory:

A screenshot of a terminal window with a dark background. At the top, there are five tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected and underlined), and SQL CONSOLE. The terminal shows a prompt 'coder@4051048b83c8:~/project\$' followed by the command '/home/coder/coursera/backup.sh'. The output of the command is: 'Dumped database', 'adding: all.sql (deflated 78%)', 'adding: welcome/ (stored 0%)', 'adding: welcome/myFirstNode.js (deflated 40%)', and 'Backup succesfully executed!'. The prompt returns to 'coder@4051048b83c8:~/project\$' with a cursor.

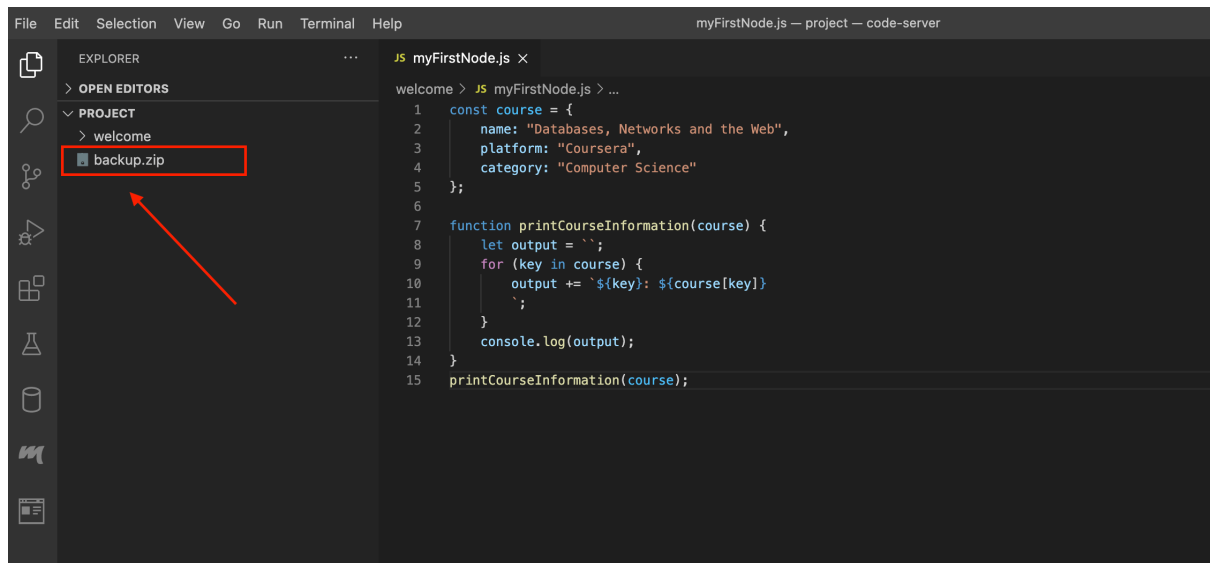
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

coder@4051048b83c8:~/project$ /home/coder/coursera/backup.sh
Dumped database
  adding: all.sql (deflated 78%)
  adding: welcome/ (stored 0%)
  adding: welcome/myFirstNode.js (deflated 40%)
Backup succesfully executed!
coder@4051048b83c8:~/project$ █
```

The “backup.zip” file will contain a backup of all of your folders and files, as well as your MySQL data (more on MySQL later in the course).

Simply drag and drop the generated “backup.zip” file from the lab environment to your desktop to save a local copy of your work:

You can of course repeat the backup process multiple times and it is strongly recommended that you backup your work frequently.



Once you have finished with the backup process, please remember to reboot the lab in order to restore the latest lab experience.

## End of Section

If you are getting the same output, congratulations for completing this lab activity. You can practice further by modifying the *myFirstNode.js* file; perhaps you can add more properties to the *course* object or rewrite your own version of the *printCourseInformation(course)* function. Make sure to save your changes before running the script again.

In the next lab activity you will create your own virtual server, isn't that exciting? There are no other requirements for this section and you can carry on with the course material.