+ -

Helvetica N...
▼

▢▢▢
Step 2 of 5

Hands-on Lab - First Server with ServerSide Java Script (20 min)

Objective for Exercise:

- Use the terminal to git clone and get Node.JS server code
- Create a web server using Server side Java script
- Run the server
- Access the server from the client and get a response from server

Step 1: Verify Environment and Command-line tools

1. Open a terminal window by using the menu in the editor: Terminal > New Terminal.
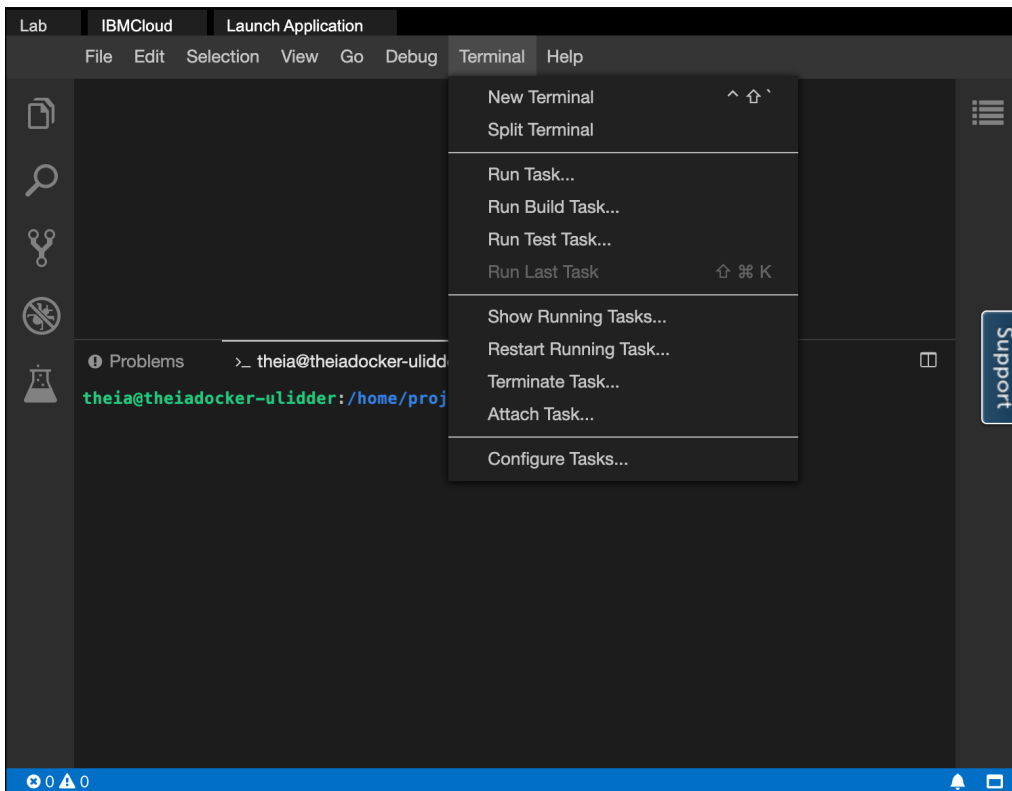
2. Verify that `node` CLI is installed.

```
node --version
```

You should see output similar to this, though the versions may be different:

```
v12.18.3
```

3. Change to your project folder.

```
cd /home/project
```

4. Clone the git repository that contains the artifacts needed for this lab, if it doesn't already exist.

```
git clone https://github.com/ibm-developer-skills-network/lkpho-Cloud-applications-with-Node.js-and-React.git
```

5. Change to the directory for this lab.

```
cd lkpho-Cloud-applications-with-Node.js-and-React/CD220Labs/http_server
```

6. List the contents of this directory to see the artifacts for this lab.

```
ls
```

7. Check the content of index.js. This is the server side script we will run in the next section.

```
cat index.js
```

You should see output similar to this.

```
const http = require('http');

const requestListener = function (req, res) {
  res.writeHead(200);
  res.end('Hello, World!');
}

const port = 8080;
const server = http.createServer(requestListener);
console.log('server listening on port: ' + port);
```

Alternatively, you can also view the content of index.js through the file explorer menu. It would appear like this.
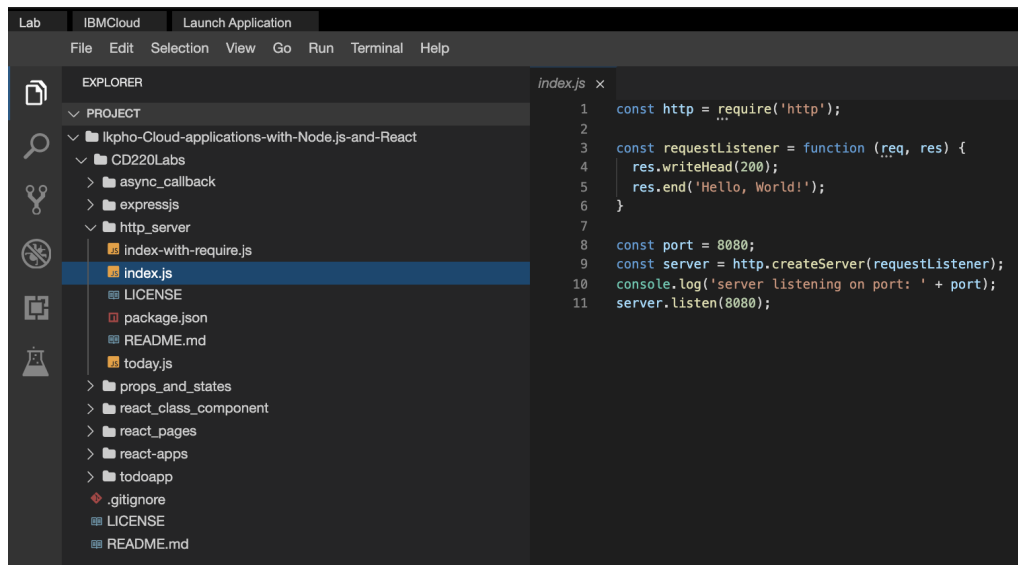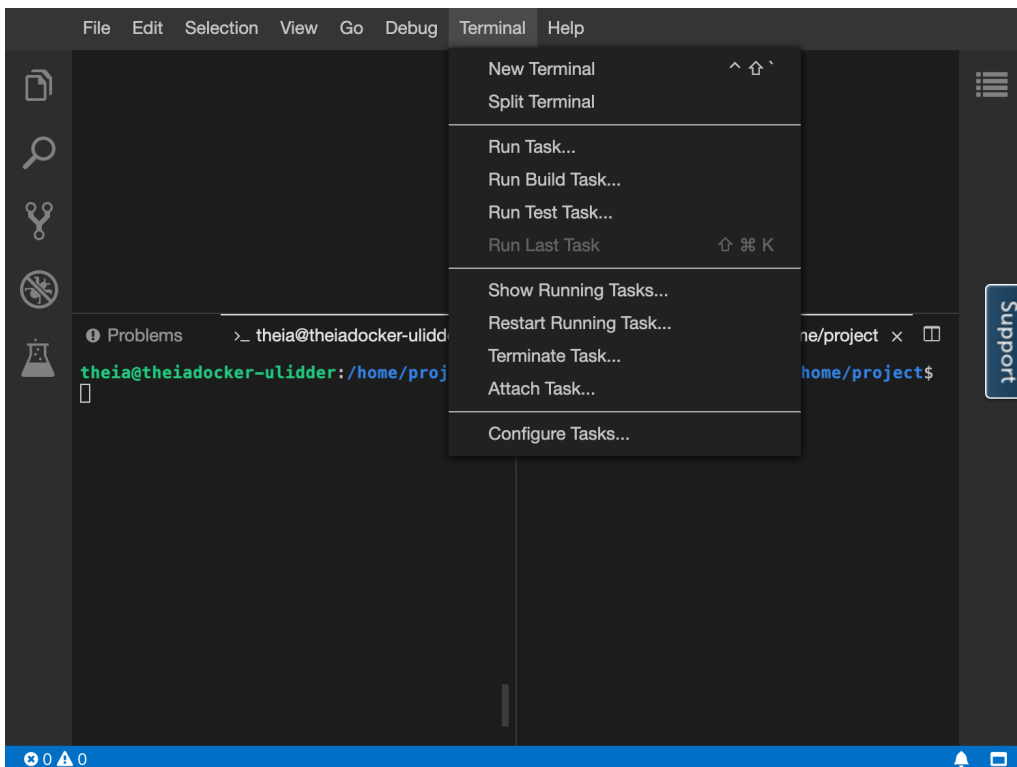

Step 2: Use the node CLI

1. In order to start the server, we run the index.js file with the node command.

```
node index.js
```

You should see output similar to this.

```
server listening on port: 8080
```

2. To split the terminal, click Terminal > Split Terminal.

3. In the second terminal window, use the `curl` command to ping the application.
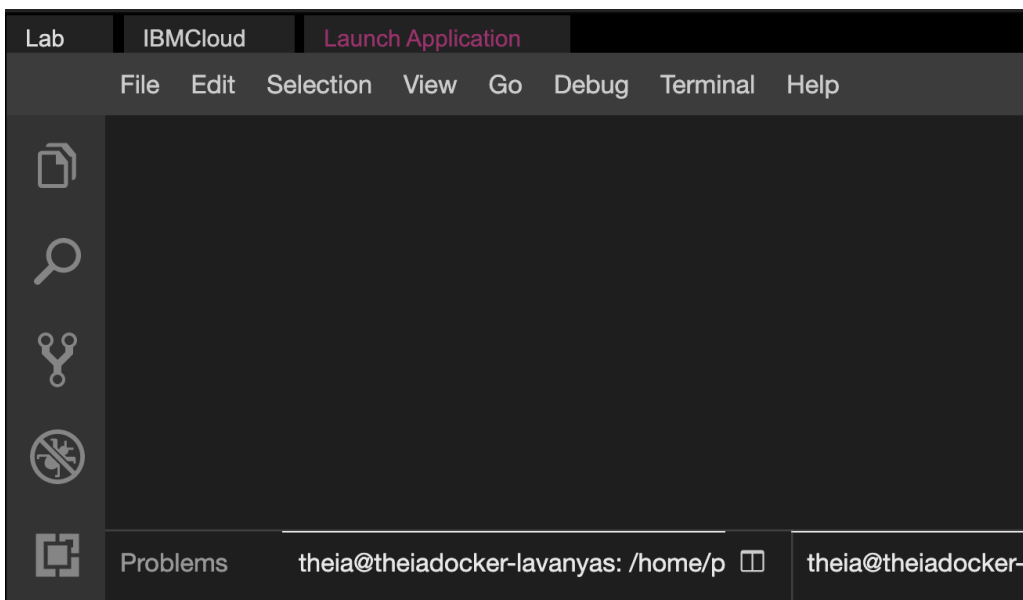
```
curl localhost:8080
```

You should see output similar to this.

```
Hello, World!
```

It should indicate that your app is up and running.

4. To verify the same with browser window, click on `Launch Application` in the menu.

A window will pop up as below for you to enter the port number. Enter the port number the server is running on, which is `8080` in it.

labs.cognitiveclass.ai says

What port is your application running on?

8080

Cancel   OK

A new browser window will open up as below. (*Note: New browser window may not open up if you browser settings does not allow pop-ups*)

← → C   🔒 lavanyas-8080.theiadocker-2.proxy.cognitiveclass.ai

Hello, World!
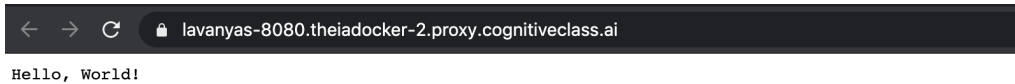
5. To stop the server, go to the main command window and press Ctrl+c to stop the server and stay in that terminal.

# Step 3: Use the node CLI to run server script which requires another module

1. In the same terminal, check the content of today.js.

```
cat today.js
```

You should see output similar to this.

```
module.exports.getDate = function getDate() {
    var aestTime = new Date().toLocaleString("en-US", {timeZone: "Australia/Brisbane"});
    return aestTime;
}
```

Alternatively, you can also view the content of today.js through the file explorer menu. It would appear like this.
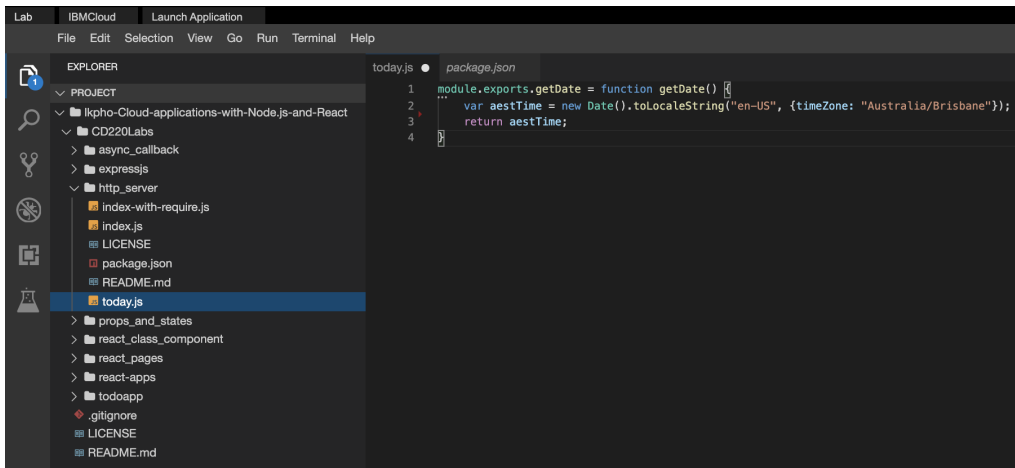
We will use this exported module in the server side script.

> 2. Check the content of index-with-require.js. As you will observe, this script requires the module `today` whose content we
>   saw in the previous step.

```
cat index-with-require.js
```

You should see output similar to this.

```
const http = require('http');
const today = require('./today');

const requestListener = function (req, res) {
  res.writeHead(200);
  res.end(`Hello, World! The date today is ${today.getDate()}`);
}

const port = 8080;
const server = http.createServer(requestListener);
console.log('server listening on port: ' + port);
```

Alternatively, you can also view the content of index-with-require.js through the file explorer menu. It would appear like this.

> 3. In order to start the server, we run the index-with-require.js file with the node command.

```
node index-with-require.js
```

You should see output similar to this.

```
server listening on port: 8080
```

> 4. In the second terminal window which you opened earlier, use the `curl` command to ping the application.

```
curl localhost:8080
```

You should see output similar to this.

```
Hello, World! The date today is Wed Oct 14 2020 14:56:42 GMT+1030 (Australian Eastern Standard Time)
```

It should indicate that your app is up and running.

> 5. To verify the same with browser window, click on `Launch Application` in the menu.

Enter the port number `8080` in the window which pops up.

A new browser window will open up which show `Hello World!` along with the date and time in your time zone.

Challenge:

Make changes in index-with-require.js to wish the user depending on the time of the day.

▶ Click here for a sample solution

Congratulations! You have completed the lab for the first module of this course.

Summary

Now that you have have learnt how to run a server we will go further and extend the capabilities of our server side.

Author(s)

Lavanya

Changelog

Date	Version	Changed by	Change Description
2020-11-04	1.0	Lavanya	Initial version created based videos

Previous   Continue