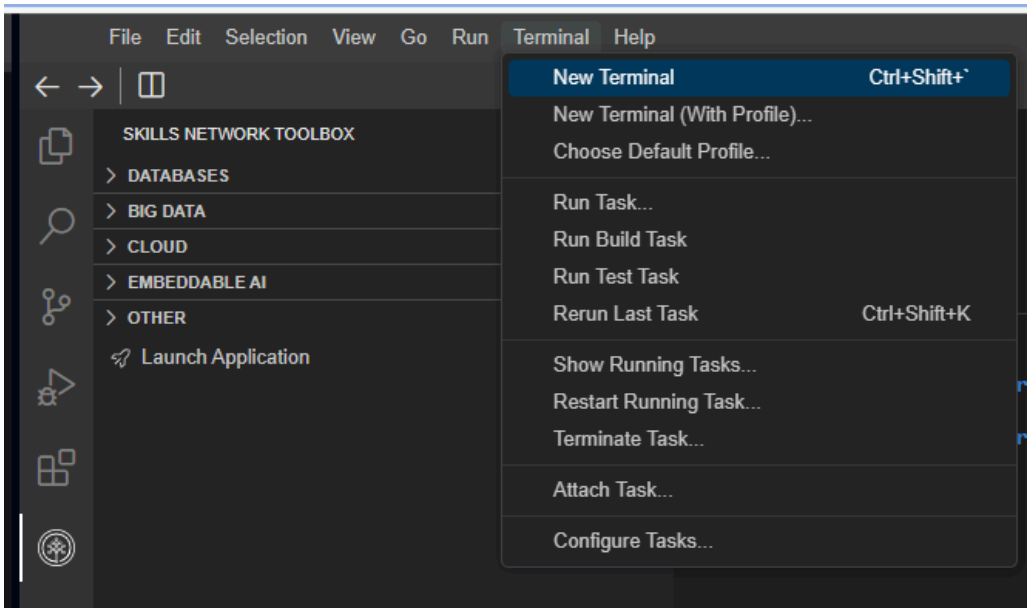
动手实验 – 第一个服务器与服务器端 Java 脚本 (20 分钟)

练习目标：

- 使用终端进行 git clone 并获取 Node.JS 服务器代码
- 使用服务器端 Java 脚本创建一个网络服务器
- 运行服务器
- 从客户端访问服务器并获取服务器的响应

步骤 1：验证环境和命令行工具

1. 通过编辑器菜单打开终端窗口：终端 > 新建终端。



2. 验证 `node` CLI 是否已安装。

```
node --version
```

您应该看到类似于此的输出，尽管版本可能不同：

```
v19.9.0
```

3. 切换到你的项目文件夹。

```
cd /home/project
```

4. 如果尚不存在，请克隆包含本实验所需工件的 git 仓库。

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

5. 切换到此实验的目录。

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

6. 列出此目录的内容以查看本实验的工件。

```
ls
```

7. 检查 index.js 的内容。这是我们将在下一节中运行的服务器端脚本。

```
cat index.js
```

您应该看到类似于此的输出。

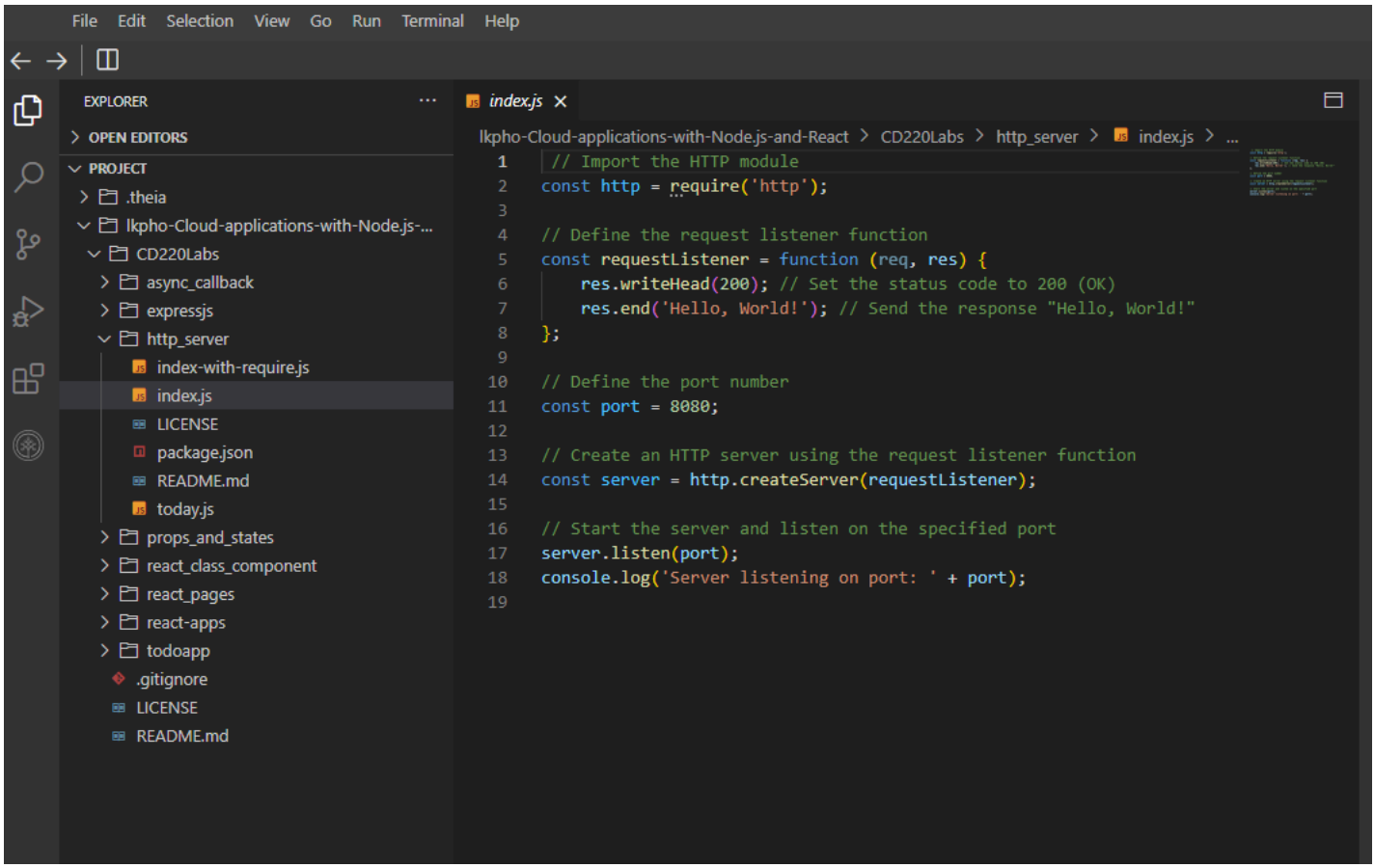
```
 // Import the HTTP module
const http = require('http');
// Define the request listener function
const requestListener = function (req, res) {
    res.writeHead(200); // Set the status code to 200 (OK)
    res.end('Hello, World!'); // Send the response "Hello, World!"
};
// Define the port number
const port = 8080;
// Create an HTTP server using the request listener function
```

```
const server = http.createServer(requestListener);
// Start the server and listen on the specified port
server.listen(port);
console.log('Server listening on port: ' + port);
```

解释：

- `HTTP Module` 从 Node.js 导入 HTTP 模块。
- `Request Listener Function` 定义一个请求监听函数，用于处理传入的 HTTP 请求。在这种情况下，函数将状态码设置为 200（OK），并发送响应“Hello, World!”。
- `Port Definition` 使用 const port = 8080; 定义端口号为 8080。
- `Create HTTP Server` 使用 http.createServer(requestListener); 创建 HTTP 服务器，其中 requestListener 是定义的请求监听函数。
- `Server Start` 启动服务器并在指定端口上监听，使用 server.listen(port);。同时，记录一条消息，指示服务器正在指定端口上监听。

另外，您也可以通过文件资源管理器菜单查看 index.js 的内容。它的显示如下。



步骤 2：使用 node CLI

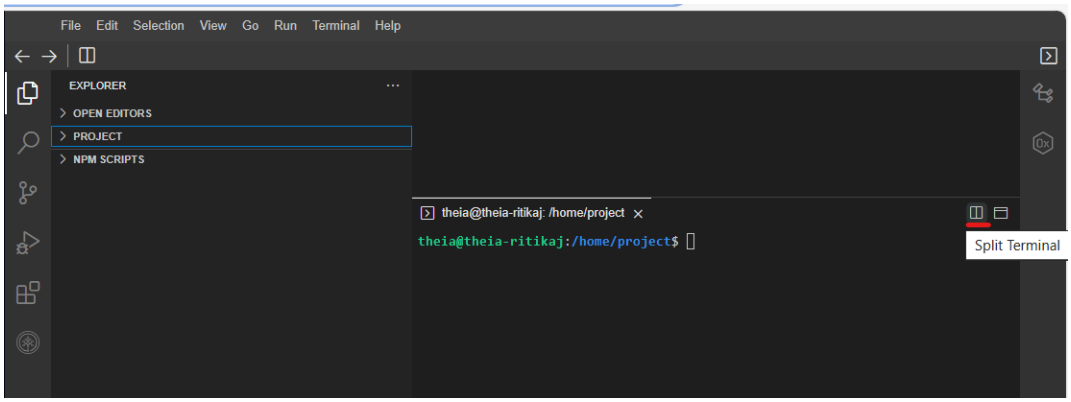
1. 为了启动服务器，我们使用 node 命令运行 index.js 文件。

```
node index.js
```

您应该看到类似于这样的输出。

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

2. 要分割终端，请点击“分割终端”，如下图所示。



3. 在第二个终端窗口中，使用 `curl` 命令来 ping 应用程序。

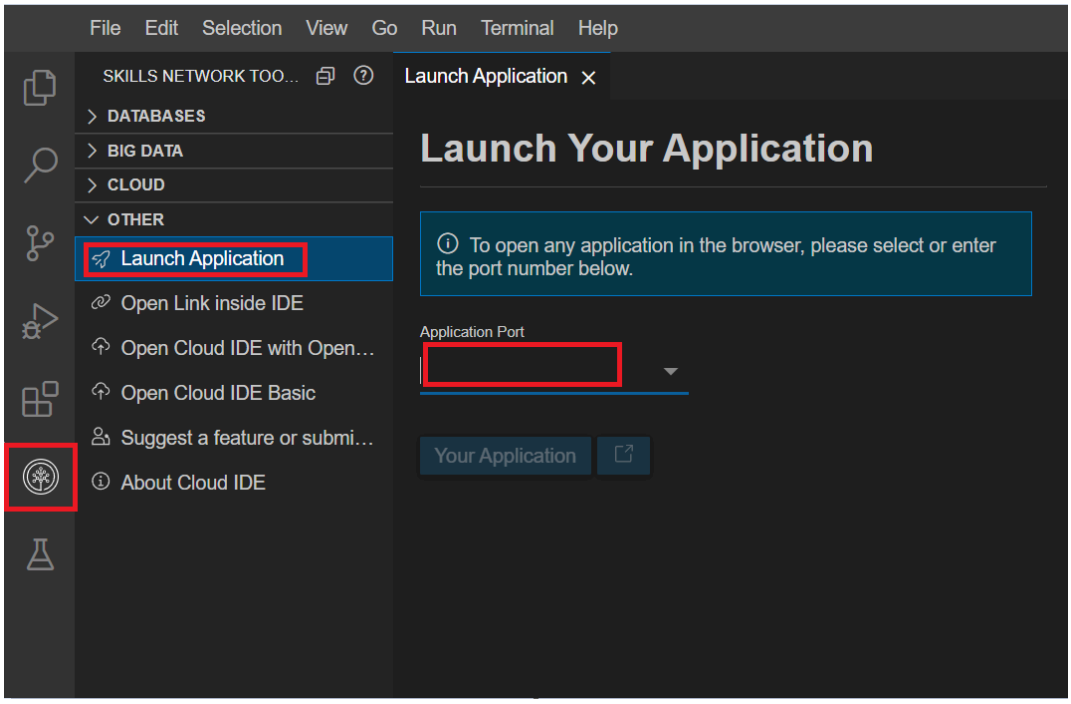
```
curl localhost:8080
```

您应该看到类似于这样的输出。

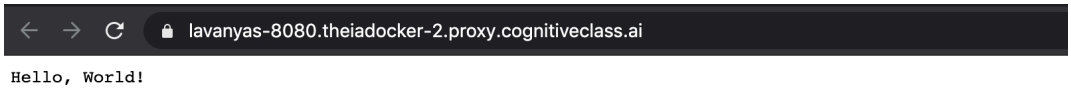
```
Hello, World!
```

它应该表明您的应用程序正在运行。

4. 要通过浏览器窗口验证，请点击左侧的技能网络按钮，这将打开“技能网络工具箱”。然后点击 **其他**，再点击 **启动应用程序**。在这里，您应该能够输入服务器运行的端口号，即 `8080` 并启动。



一个新的浏览器窗口将会打开，如下所示。(*注意：如果您的浏览器设置不允许弹出窗口，则可能不会打开新的浏览器窗口*)




```
Hello, World!
```

5. 要停止服务器，请转到主命令窗口并按 Ctrl+c 停止服务器，并保持在该终端中。

步骤 3：使用 node CLI 运行需要另一个模块的服务器脚本

1. 在同一个终端中，检查 today.js 的内容。

```
cat today.js
```

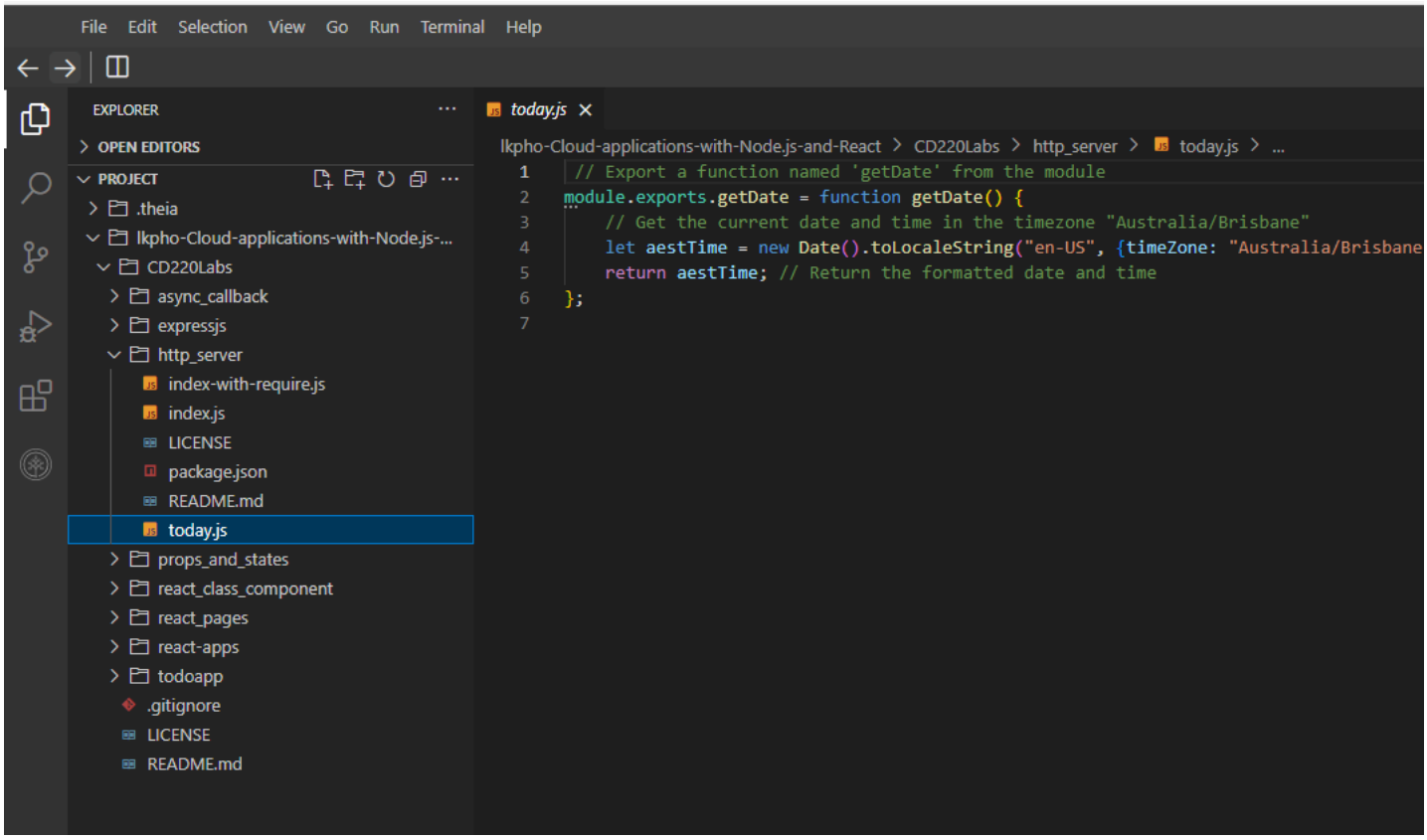
您应该看到类似于这样的输出。

```
 // Export a function named 'getDate' from the module
module.exports.getDate = function getDate() {
    // Get the current date and time in the timezone "Australia/Brisbane"
    let aestTime = new Date().toLocaleString("en-US", {timeZone: "Australia/Brisbane"});
    return aestTime; // Return the formatted date and time
};
```

解释：

- 导出函数 使用 module.exports 导出一个名为 getDate 的函数。
- 日期格式化 在 getDate 函数内部，使用 new Date().toLocaleString("en-US", {timeZone: "Australia/Brisbane"}) 获取当前日期和时间。这会根据时区 "Australia/Brisbane" 格式化日期和时间。
- 返回值 使用 return aestTime; 返回格式化后的日期和时间。

另外，您也可以通过文件资源管理器菜单查看 today.js 的内容。它的显示方式如下。



我们将在服务器端脚本中使用这个导出的模块。

2. 检查 index-with-require.js 的内容。如您所见，这个脚本需要模块 `today`，我们在前一步中看到了它的内容。

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

您应该看到类似于这样的输出。

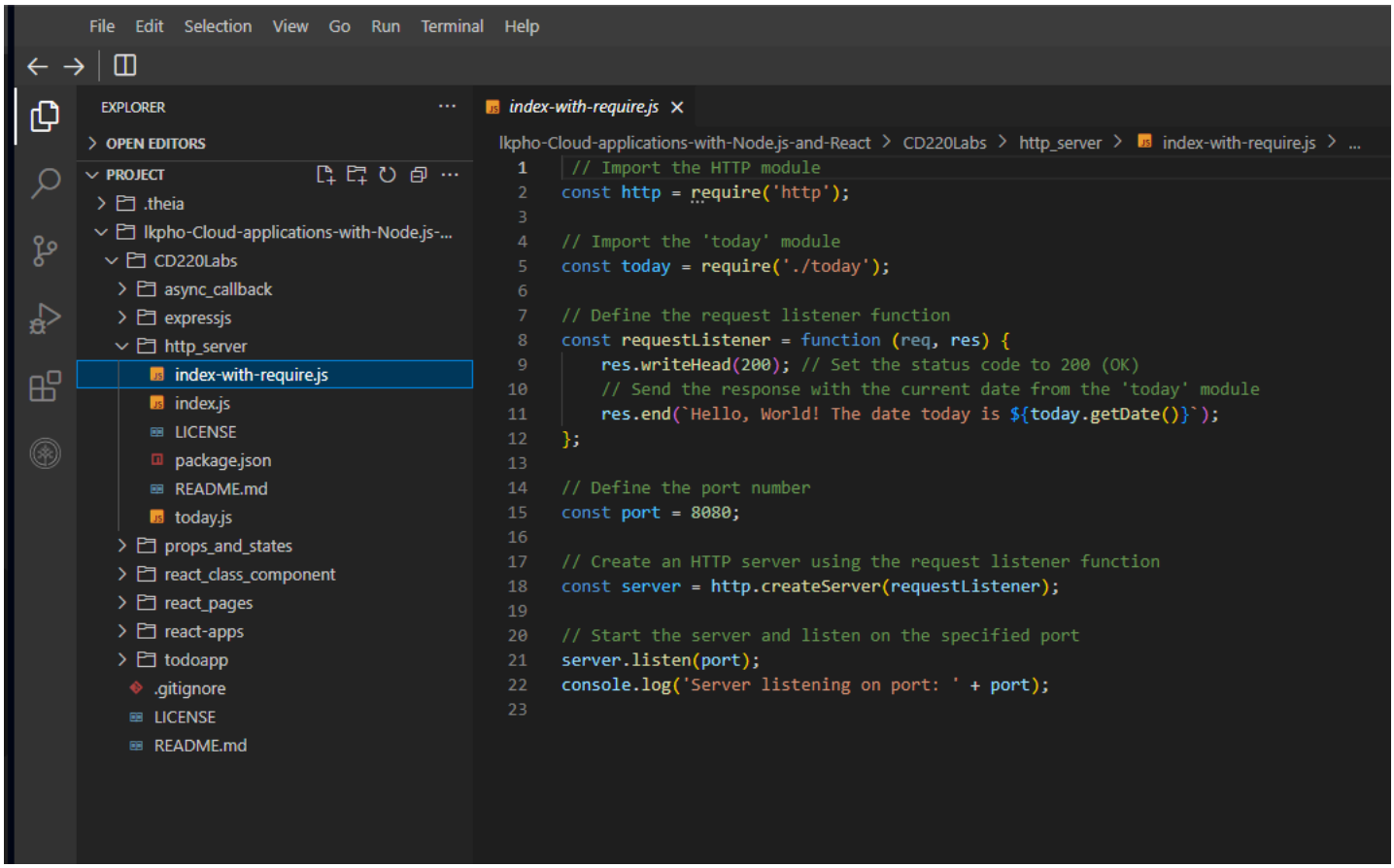
```
 // Import the HTTP module
const http = require('http');
// Import the 'today' module
const today = require('./today');
// Define the request listener function
const requestListener = function (req, res) {
    res.writeHead(200); // Set the status code to 200 (OK)
    // Send the response with the current date from the 'today' module
    res.end(`Hello, World! The date today is ${today.getDate()}`);
};
// Define the port number
const port = 8080;
// Create an HTTP server using the request listener function
const server = http.createServer(requestListener);
// Start the server and listen on the specified port
server.listen(port);
```

```
console.log('Server listening on port: ' + port);
```

解释：

- `HTTP Module` 从 Node.js 导入 HTTP 模块。
- `Module Import` 使用 const today = require('./today'); 导入 'today' 模块。这假设存在一个名为 'today' 的模块，导出一个函数 getDate。
- `Request Listener Function` 定义一个请求监听函数，用于处理传入的 HTTP 请求。在这种情况下，该函数将状态码设置为 200（OK），并使用 'today' 模块中的 getDate 函数发送响应 "Hello, World! 今天的日期是 {current date}"。
- `Port Definition` 使用 const port = 8080; 定义端口号为 8080。
- `Create HTTP Server` 使用 http.createServer(requestListener); 创建一个 HTTP 服务器，其中 requestListener 是定义的请求监听函数。
- `Server Start` 启动服务器并在指定端口上监听，使用 server.listen(port);。同时，记录一条消息，指示服务器正在指定端口上监听。

另外，您还可以通过文件资源管理器菜单查看 index-with-require.js 的内容。它将显示如下。



3. 为了启动服务器，我们使用 node 命令运行 index-with-require.js 文件。

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

您应该看到类似于这样的输出。

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

4. 在您之前打开的第二个终端窗口中，使用 `curl` 命令来 ping 应用程序。

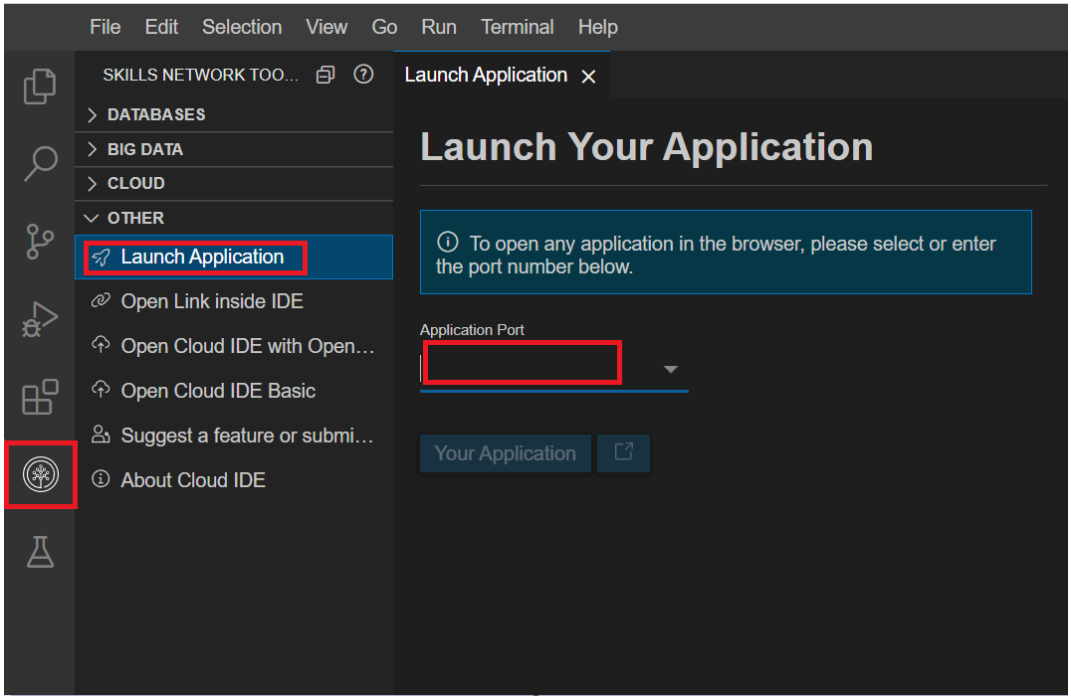
```
curl localhost:8080
```

您应该看到类似于这样的输出。

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

它应该指示您的应用程序已启动并运行。

5. 要通过浏览器窗口验证，请点击左侧的 **Skills Network** 按钮，这将打开“Skills Network Toolbox”。然后点击 **Other** 然后 **Launch Application**。从那里您应该能够输入端口号 `8080` 并启动。



一个新的浏览器窗口将打开，显示 `Hello World!` 以及您所在时区的日期和时间。

挑战：

在 index-with-require.js 中进行更改，以根据一天中的时间问候用户 **取决于**。

▼ 点击这里查看示例解决方案

```
 // Import the HTTP module
const http = require('http');
// Import the 'today' module
const today = require('./today');
// Define the request listener function
const requestListener = function (req, res) {
    res.writeHead(200); // Set the status code to 200 (OK)
    let dateVal = today.getDate(); // Get the current date from the 'today' module
    // Determine the appropriate greeting based on the current time
    let greeting = "It is still not morning";
    if (dateVal.getHours() > 6 && dateVal.getHours() < 12) {
        greeting = "Good morning!";
    } else if (dateVal.getHours() >= 12 && dateVal.getHours() < 18) {
        greeting = "Good afternoon!";
    } else if (dateVal.getHours() >= 18 && dateVal.getHours() < 21) {
        greeting = "Good evening!";
    } else if (dateVal.getHours() >= 21 && dateVal.getHours() < 24) {
        greeting = "Good night!";
    }
    // Send the response with the appropriate greeting
    res.end(`Hello, ${greeting}`);
};
// Define the port number
const port = 8080;
// Create an HTTP server using the request listener function
const server = http.createServer(requestListener);
// Start the server and listen on the specified port
server.listen(port);
console.log('Server listening on port: ' + port);
```

恭喜你！你已经完成了实验。

摘要

现在你已经学会了如何运行服务器，你准备好创建自己的 Node.JS 服务器了。

Author(s)

Lavanya

© IBM Corporation. 保留所有权利。