Practice Basic Package Management Commands to Add, Update, and Remove Dependencies

Lab Overview

In this lab, you will practice adding, updating, and removing dependencies using package management tools in both Node.js (npm) and Python (pip).

Prerequisites

1. Node.js installed
   - Verify with `node -v` and `npm -v`
2. Python 3.8 installed
   - Verify with `python3 --version` or `python --version`
3. A text editor or IDE (e.g., VS Code)
4. Command-line/terminal access

Part 1: Node.js Package Management with npm

Step 1: Initialize a Node.js Project

1. Open your terminal and create a new directory:

   ```
   cd ~/Desktop
   mkdir node-package-lab
   cd node-package-lab
   ```

2. Initialize a new Node.js project:

   ```
   npm init -y
   ```

   - This creates a `package.json` file.

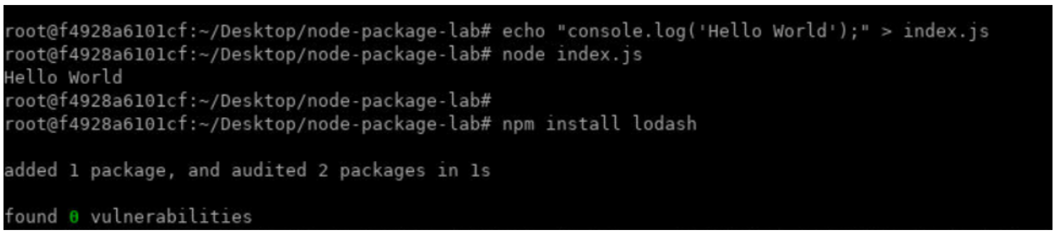
3. Print "Hello World":

   ```
   echo "console.log('Hello World');" > index.js
   node index.js
   ```

Step 2: Add Dependencies

1. Add a dependency (e.g., `lodash`):

   ```
   npm install lodash
   ```



```
root@f4928a6101cf:~/Desktop/node-package-lab# echo "console.log('Hello World');" > index.js
root@f4928a6101cf:~/Desktop/node-package-lab# node index.js
Hello World
root@f4928a6101cf:~/Desktop/node-package-lab#
root@f4928a6101cf:~/Desktop/node-package-lab# npm install lodash

added 1 package, and audited 2 packages in 1s

found 0 vulnerabilities
```

2. Verify that `lodash` was added to your `package.json` under `dependencies`.

3. Check the `node_modules` folder to confirm installation.

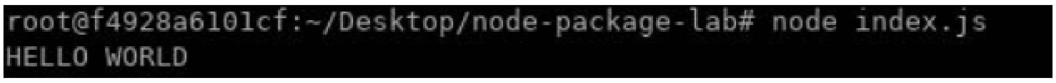
Step 3: Use and Run Installed Package

1. Modify `index.js` to use `lodash`:

   ```
   const _ = require('lodash');
   console.log(_.toUpper('hello world'));
   ```

   Run the foolllowing command from the terminal:

   ```
   node index.js
   ```

   ```
   root@f4928a6101cf:~/Desktop/node-package-lab# node index.js
   HELLO WORLD
   ```

   - This code uses `lodash` to toUpper the string "hello world".

Step 4: Update Dependencies

1. Update the `lodash` package to the latest version:

   ```
   npm update lodash
   ```

2. Verify the updated version in `package.json` and `package-lock.json`.

Step 5: Remove Dependencies

1. Remove the `lodash` package:

   ```
   npm uninstall lodash
   ```

2. Verify that `lodash` has been removed from `package.json` and the `node_modules` folder.

Part 2: Python Package Management with pip

Step 1: Set Up a Python Virtual Environment

1. Create a new directory:

   ```
   cd ~/Desktop
   mkdir python-package-lab
   ```

```
cd python-package-lab
```

2. Create a virtual environment:

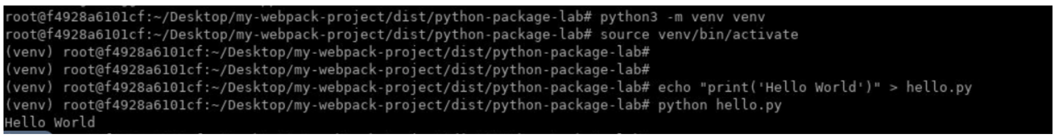
```
python3 -m venv venv
```

3. Activate the virtual environment in Linux:

```
source venv/bin/activate
```

4. Print "Hello World":

```
echo "print('Hello World')" > hello.py
python hello.py
```



```
root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# python3 -m venv venv
root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# source venv/bin/activate
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab#
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab#
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# echo "print('Hello World')" > hello.py
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# python hello.py
Hello World
```

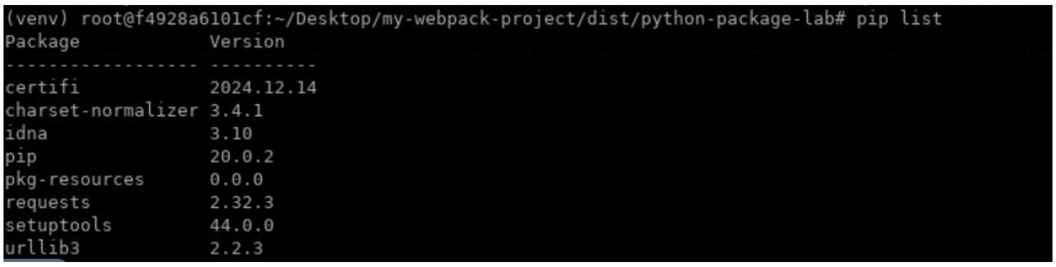
Step 2: Add Dependencies

1. Install a package (e.g., `requests`):

```
pip install requests
```

2. Verify installation:

```
pip list
```



```
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# pip list
Package            Version
------------------ ----------
certifi            2024.12.14
charset-normalizer 3.4.1
idna               3.10
pip                20.0.2
pkg-resources      0.0.0
requests           2.32.3
setuptools         44.0.0
urllib3            2.2.3
```

3. Freeze dependencies to a `requirements.txt` file:

```
pip freeze > requirements.txt
```

Step 3: Use and Run Installed Package

1. Create a `script.py` file to use `requests`:

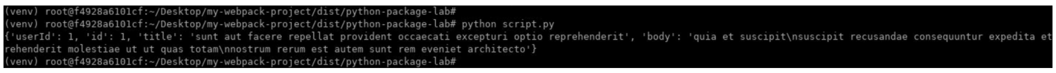
```
import requests

# Sending a GET request to the API endpoint
response = requests.get('https://jsonplaceholder.typicode.com/posts/1')
```

```
# Checking if the request was successful
if response.status_code == 200:
   # Printing the JSON response as a dictionary
   print(response.json())
else:
   # Printing an error message if the request failed
   print(f"Request failed with status code: {response.status_code}")
```

2. Run the script with following command:

```
python script.py
```

   - This code fetches and prints a sample JSON response from a placeholder API.



```
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab#
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab# python script.py
{'userId': 1, 'id': 1, 'title': 'sunt aut facere repellat provident occaecati excepturi optio reprehenderit', 'body': 'quia et suscipit\nsuscipit recusandae consequuntur expedita et
rehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto'}
(venv) root@f4928a6101cf:~/Desktop/my-webpack-project/dist/python-package-lab#
```

Step 4: Update Dependencies

1. Upgrade the `requests` package:

```
pip install --upgrade requests
```

2. Verify the updated version:

```
pip list
```

Step 5: Remove Dependencies

1. Uninstall the `requests` package:

```
pip uninstall requests
```

2. Verify removal:

```
pip list
```

Step 6: Deactivate Virtual Environment

1. Deactivate the virtual environment:

```
deactivate
```

Summary

In this lab, you learned how to:

1. Add, update, and remove dependencies using npm for Node.js.
2. Add, update, and remove dependencies using pip for Python in a virtual environment.
3. Run code using installed packages in both Node.js and Python.