

To install and set up Node.js, npm, and Vue CLI on a WSL (Windows Subsystem for Linux) environment, follow these steps:

## 1. Install WSL

If you haven't installed WSL, follow the instructions from the official Microsoft documentation [Windows Subsystem for Linux Documentation | Microsoft Learn](#) to enable WSL on your Windows machine. [Install WSL | Microsoft Learn](#)

## 2. Open a new WSL Terminal

Launch your WSL terminal (e.g., Ubuntu).

## 3. Install `curl`

To enable the download of content from the internet via Ubuntu, install `curl`:

```
$ sudo apt-get install curl
```

## 4. Install `nvm` (Node Version Manager)

Check the latest version of `nvm` and substitute into the path accordingly. As of this post, v0.39.1 is the latest:

```
$ curl -o-  
https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.1/install.sh | bash
```

## 5. Verify `nvm` Installation

You may need to close and reopen your terminal. Then verify the installation of `nvm`:

```
$ command -v nvm
```

If `nvm` is installed correctly, this command should return "nvm".

## 6. Install Node.js (Current Stable Release)

Install the latest Long Term Support (LTS) version of Node.js:

```
$ nvm install --lts
```

## 7. Confirm Node.js and npm Installation

Verify that you have Node.js and npm installed:

```
$ node --version && npm --version
```

You should see output similar to:

```
v16.16.0  
8.11.0
```

## 8. Uninstall Existing Node.js (If Applicable)

If you already have a version of Node.js installed, it may be beneficial to uninstall it and reinstall using `nvm` to avoid potential issues.

## 9. Install Vue CLI

With Node.js and npm installed, you can now install Vue CLI:

```
$ npm install -g @vue/cli
```

## 10. Create a New Vue CLI Project

Create a new Vue CLI project and follow the prompts:

```
$vue create my-vue-project
```

**During the prompts:**

1. Select `Manually select features`.
2. Choose `Vue 3`.
3. Select `Router`.
4. Choose `ESLint with Prettier`.
5. Say `No` to "Save this as a preset for future projects?".
6. Say `Yes` to "Use history mode for router?".
7. Select `package.json` as the configuration format.

This will complete the Vue CLI project setup and create a Vue CLI project in the directory.

## Additional Resources

For further details, you can refer to the [official Microsoft WSL documentation](#) and the Vue CLI documentation.

## Installation Instructions for macOS

To set up Node.js, npm, and Vue CLI on a macOS environment, follow these steps:

### 1. Install Homebrew

First, install Homebrew if you don't have it installed. Open a terminal and run:

```
$ /bin/bash -c "$(curl -fsSL  
https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

### 2. Install **nvm** (Node Version Manager)

Install **nvm** using Homebrew:

```
$ brew install nvm
```

Create the **nvm** directory:

```
$ mkdir ~/.nvm
```

Add the following lines to your shell profile (`~/.zshrc` for zsh, `~/.bash_profile` for bash):

```
$ export NVM_DIR="$HOME/.nvm"  
[ -s "/usr/local/opt/nvm/nvm.sh" ] && \. "/usr/local/opt/nvm/nvm.sh"
```

Source the profile to apply the changes:

```
$ source ~/.zshrc # or source ~/.bash_profile
```

### 3. Install Node.js (Current Stable Release)

Install the latest Long Term Support (LTS) version of Node.js:

```
$ nvm install --lts
```

#### 4. Confirm Node.js and npm Installation

Verify that you have Node.js and npm installed:

```
$ node --version && npm --version
```

You should see output similar to:

```
v16.16.0  
8.11.0
```

#### 5. Install Vue CLI

With Node.js and npm installed, you can now install Vue CLI:

```
$ npm install -g @vue/cli
```

#### 6. Create a New Vue CLI Project

Create a new Vue CLI project and follow the prompts:

```
$
```

```
vue create my-vue-project
```

**During the prompts:**

1. Select `Manually select features`.
2. Choose `Vue 3`.
3. Select `Router`.
4. Choose `ESLint with Prettier`.
5. Say `No` to "Save this as a preset for future projects?".
6. Say `Yes` to "Use history mode for router?".
7. Select `package.json` as the configuration format.

This will complete the Vue CLI project setup and create a Vue CLI project in the directory.

## Additional Resources

For further details, you can refer to the Vue CLI documentation.

Feel free to ask if you need any more help!

# Preparing for installation of Virtual environment

Before installing the packages, you need to follow [our guide](#) to running Ubuntu Server 22.04 as a standard user.

## Download Python 3

Let's update the package index and run the command to update the packages to the latest releases:

```
sudo apt update && sudo apt upgrade -y
```

The “-y” key means to force update.  
Checking the Python version goes like this:

```
python3 --version
```

Output is going to be like this:

```
#Output
```

```
Python 3.10.6
```

The next step is to install python3-pip in order to manage Python packages. Let's use the built-in command:

```
sudo apt install python3-pip -y
```

## Setting up a virtual environment

A virtual development environment on a production server is considered a great solution compared to running in a main development environment. In a virtual environment, you can edit and not damage the files of the main development environment. We can create as many virtual environments as we need. Each virtual environment is deployed in different directories on our server. The directories contain files for initializing the virtual environment.

The virtual environment is deployed using the installed venv (virtual environment) package:

```
sudo apt install python3-venv -y
```

Then let's create a directory called test:

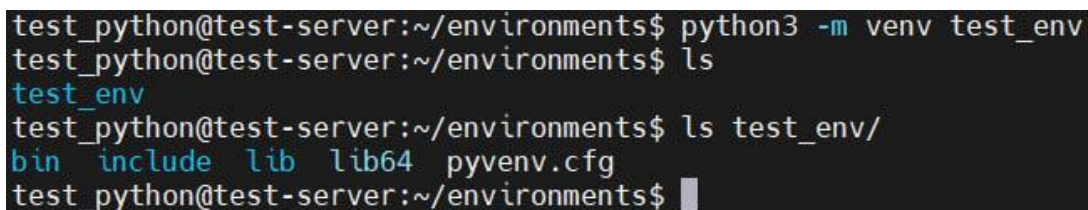
```
mkdir test
```

```
cd test
```

Change to the first directory and use the following command to create a virtual environment called test\_env:

```
python3 -m venv test_env
```

The result is shown in Screen 2.

A terminal window with a dark background and light-colored text. The prompt is 'test\_python@test-server:~/environments\$'. The user enters 'python3 -m venv test\_env'. The prompt changes to 'test\_python@test-server:~/environments\$'. The user enters 'ls'. The output is 'test\_env'. The prompt changes to 'test\_python@test-server:~/environments\$'. The user enters 'ls test\_env/'. The output is 'bin include lib lib64 pyvenv.cfg'. The prompt changes to 'test\_python@test-server:~/environments\$' and a cursor is visible.

```
test_python@test-server:~/environments$ python3 -m venv test_env
test_python@test-server:~/environments$ ls
test_env
test_python@test-server:~/environments$ ls test_env/
bin include lib lib64 pyvenv.cfg
test_python@test-server:~/environments$
```

Screen 2 - Create a virtual environment

The generated files configure the virtual environment to work separately from our host files. Activation of the environment is as follows, and to disable the environment, you must run the deactivate command:

```
source test/test_env/bin/activate
```

OR

```
. test/test_env/bin/activate
```

To disable the virtual environment, run the command:

```
deactivate
```