



Node.js Programming on Ubuntu



Rashmi Purandare



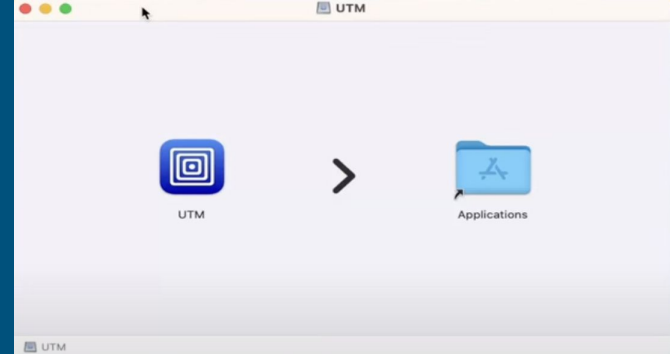
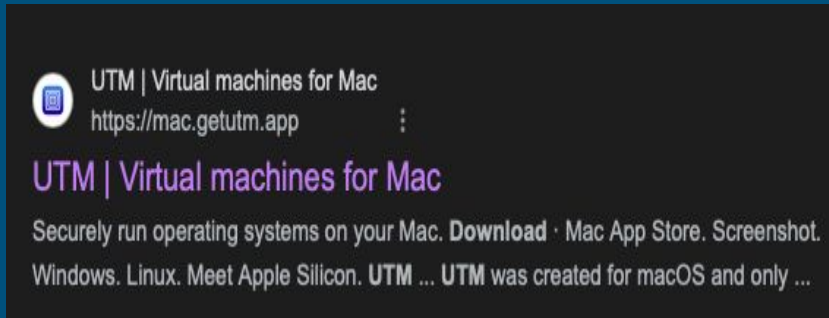
Table of content

1. How to install Ubuntu
 - a. Setup Ubuntu on MacBook virtually
 - b. Configuration
2. How to install JDK on Ubuntu
 - a. Set up Java Path
3. How to check Ubuntu version
4. How to install & uninstall Nodejs on Ubuntu
5. How to set up time server
6. Understanding JSON?
7. Understanding HTTP JSON API Server

How to install Ubuntu 24.04 on Mac OS virtually.

Download & Install UTM

- Here we will use a tool called UTM virtual machine for MacBook.
- Firstly, we will install UTM on you Mac which runs securely on the OS.



Download Ubuntu ARM

- After installing UTM, we will download the ARM image of Ubuntu 24 since MacBook supports the ARM architecture.
- On clicking below link you will be able to download Ubuntu server for ARM
<https://ubuntu.com/download/server/arm>

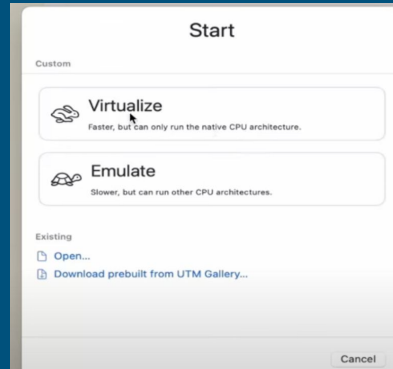
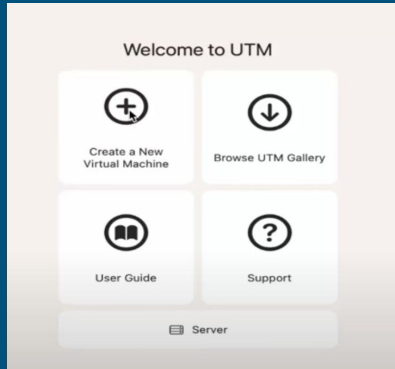
Ubuntu Server

This is the default ISO image of the Ubuntu Server installer.

Download 24.04.1 LTS

Creating New Virtual Machine in UTM

- Open UTM application now and select create new Virtual Machine then select virtualize option and then select Linux OS.



- Then, browse the downloaded ARM file and select it.
- Allocate sufficient resources (CPU, RAM) to the virtual machine to ensure smooth performance and select any shared directory from your system eg. Downloads so that file transferring will be easy between Ubuntu and Mac.

Install Ubuntu 24.04 LTS

- Start the virtual machine and follow the on-screen instructions to install Ubuntu 24.04 LTS.
- Complete the installation process, setting up your user account and preferences.
- Once you reach the below screen , the installation is completed and you can click on reboot.

```
writing install sources to disk
running 'curtin extract'
curtin command extract
  acquiring and extracting image from cp:///tmp/tap0zesqek1/mount
configuring keyboard
curtin command in-target:
executing curtin install curthooks step
curtin command install
configuring installed system
  running 'curtin curthooks'
  curtin command curthooks
    configuring apt
    configuring apt
    installing missing packages
    installing packages on target system: ('efibootmgr', 'grub-efi-arm64', 'grub-efi-arm64-signed', 'shim-signed')
    configuring iscsi service
    configuring raid (mdadm) service
    configuring NVMe over TCP
    installing kernel
    setting up swap
    apply networking config
    writing etc/fstab
    configuring multipath
    updating packages on target system
    configuring pollinate user-agent on target
    updating initramfs configuration
    configuring target system bootloader
    installing grub to target devices
    copying metadata from /cdrom
final system configuration
calculating extra packages to install
installing openssh-server
retrieving openssh-server
curtin command system-install
unpacking openssh-server
curtin command system-install
configuring cloud-init
downloading and installing security updates
curtin command in-target:
restoring apt configuration
curtin command in-target:
subiquity/late/run:
[ View full log ]
[ Reboot Now ]
```

Set Up GUI for Ubuntu

- Once you restart and enter your credentials you will be logged in to the Ubuntu server and below screen will be displayed.

```
Welcome to Ubuntu 24.04 LTS (GNU/Linux 6.8.0-31-generic aarch64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/pro

System information as of Fri May 17 04:47:01 PM UTC 2024

System load:            0.05
Usage of /:              20.9% of 29.82GB
Memory usage:           2%
Swap usage:             0%
Processes:              166
Users logged in:        0
IPv4 address for enp0s1: 192.168.64.3
IPv6 address for enp0s1: fd7d:59d1:baab:ad46:c06c:3ff:fe4b:b908

Expanded Security Maintenance for Applications is not enabled.

Updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

The programs included with the Ubuntu system are free software:
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

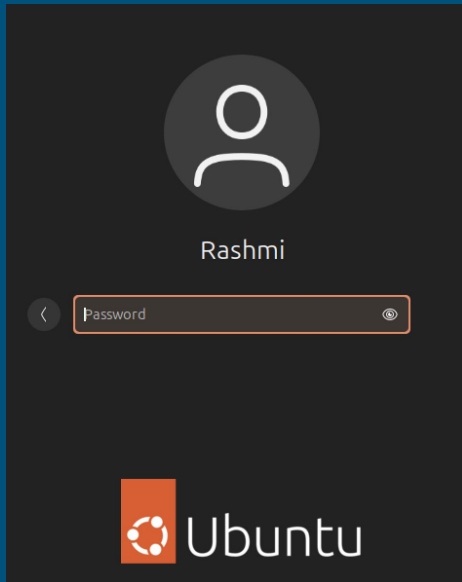
Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
```

Commands to set up the GUI and Desktop

- `sudo apt update` : It fetches the latest package lists from the repositories, ensuring that you can install the latest versions of software and dependencies.
- `sudo apt install taskel` : A utility that simplifies the installation of predefined tasks in Ubuntu
- `sudo apt install ubuntu-desktop` : This command installs the Ubuntu desktop environment (GUI) if it's not already installed. It includes the graphical interface and necessary software packages to provide a full desktop experience.
- `sudo reboot now` : This command restarts your virtual machine so that the changes (like the installation of the desktop environment) can be applied.

- Once you restart, you will be directed to the Graphical User Interface.
- Login with your credentials.
- Run below command for better integration and then reboot again.



```
sudo apt install spice-vdagent  
sudo apt install spice-webdavd
```

Welcome to Ubuntu 24.04 LTS!

Complete your setup with additional settings and we'll
have you up and running in no time

Download Oracle JDK

- Open your web browser and navigate to the Oracle JDK Downloads page <https://www.oracle.com/java/technologies/downloads/>
- Find the section for JDK and select the appropriate package for your system.
- Accept the Oracle license agreement and download the file to your computer.
- Here we install the ARM package since installed Ubuntu is also ARM compatible.

JDK 23 **JDK 21** **JDK 17** **GraalVM for JDK 23** **GraalVM for JDK 21** **GraalVM for JDK 17**

JDK Development Kit 23 downloads

JDK 23 binaries are free to use in production and free to redistribute, at no cost, under the [Oracle No-Fee Terms and Conditions \(NFTC\)](#).

JDK 23 will receive updates under these terms, until March 2025, when it will be superseded by JDK 24.

Linux **macOS** **Windows**

Product/file description	File size	Download
ARM64 Compressed Archive	228.80 MB	https://download.oracle.com/java/23/latest/jdk-23_linux-aarch64_bin.tar.gz (sha256)

Installing JDK on Ubuntu

- Open the terminal and navigate to the directory where you downloaded the JDK file.
- Select the downloaded file and extract it using below commands

```
tar -xvf jdk-23_linux-aarch64_bin.tar.gz : Extracts the file
```

```
sudo dpkg -i jdk-23_linux-aarch64_bin.deb : Installs the extracted file.
```

- To confirm JDK23 was installed correctly you can use the command

```
java -version
```

Set up JAVA Path

You can set up Java Path by two ways :

Temporary Path	Permanent Path
A temporary path is set for the current session only. Once you close the terminal or log out, the changes will be lost	A permanent path is set in a configuration file (like ~/.bashrc, ~/.bash_profile) and remains in effect even after closing the terminal or logging out.
Useful for testing or running a single command without modifying system-wide settings	Ideal for setting up environment variables that need to be accessible every time you open a new terminal session.
<code>export JAVA_HOME=/usr/lib/jvm/java-<version></code>	<code>export JAVA_HOME=/usr/lib/jvm/java-<version></code> <code>export PATH=\$JAVA_HOME/bin:\$PATH</code>

Checking Ubuntu Version

- After opening the terminal, use the `lsb_release -a` command to check the Ubuntu version.

```
rashmi@mac:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 24.04.1 LTS
Release:        24.04
Codename:       noble
```

- Instead of printing all of the above information, you can display the description line, which shows your Ubuntu version passing the `-d` switch.

```
rashmi@mac:~$ lsb_release -d
No LSB modules are available.
Description:    Ubuntu 24.04.1 LTS
```

Installation and Uninstallation of Node.js

- For installing Node.js, enable the NodeSource repository by running the following curl
 - For node.js version 19.x
 - `$ curl -sL https://deb.nodesource.com/setup_19.x | sudo -E bash -`
 - Followed by `$ sudo apt install nodejs` and `$ sudo apt-get install -y nodejs` for npm (Node Package Manager)
- Verify that the Node.js and npm were successfully by using below commands
 - `$ node --version`
 - `$ npm --version`
- If you wish to uninstall Node.js from your Ubuntu system, run the command below:
 - `sudo apt-get remove nodejs`
- The command will remove the package but retain the configuration files. To remove both the package and the configuration files run:
 - `sudo apt-get purge nodejs`
- As a final step, you can run the command below to remove any unused files and free up the disk space:
 - `sudo apt-get autoremove`

How to set up Time Server on Ubuntu

- After installing node.js, create a JavaScript file that contains the code for the time server. Then, save the code in a file called time_server.js. `nano time_server.js`
- Open the terminal and navigate to the directory where time_server.js is saved.
- Start the server by providing the port number as the first argument. For example, to run the server on port 8000 by using the command `node time_server.js 8000`
- On the client browser enter <http://localhost:8000/> to get the below output.

JavaScript

```
var net = require('net')

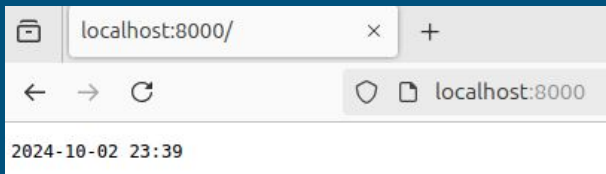
function zeroFill(i) {
  return (i < 10 ? '0' : '') + i
}

function now () {
  var d = new Date()
  return d.getFullYear() + '-'
    + zeroFill(d.getMonth() + 1) + '-'
    + zeroFill(d.getDate()) + ' '
    + zeroFill(d.getHours()) + ':'
    + zeroFill(d.getMinutes())
}

var server = net.createServer(function (socket) {
  // socket.end():
  // Half-closes the socket. i.e., it sends a FIN packet.
  // It is possible the server will still send some data.
  // - If data is specified, it is equivalent to calling
  //   socket.write(data, encoding) followed by socket.end().
  socket.write('HTTP/1.1 200 OK\n\n')
  socket.end(now() + '\n')
})

// Listening on the port provided on the command line
server.listen(Number(process.argv[2]))
```

Output



You may encounter a message indicating that port **8000** is already in use, which means another process is listening on that port. This cause your Node.js server to fail when it tries to start on the same port. Use below commands to resolve the issue. Then restart the server.

```
sudo lsof -i :8000
```

```
sudo kill -9 <PID>
```

Understanding JSON (JavaScript Object Notation)

Feature	JSON
Format	<ul style="list-style-type: none">◦ A lightweight data interchange format that is easy for humans to read and write and easy for machines to parse and generate.◦ JSON is a hierarchical data format
File Extension	.json
Data Structure	A collection of key-value pairs, where keys are strings and values can be strings, numbers, booleans, null, arrays, or other JSON objects.
Applications	Used for data exchange between web servers and clients, as well as for storing and transmitting data in various applications.
Advantages	Easy to read and write, widely supported by programming languages and web services, and can be used for complex data structures.
Disadvantages	Not suitable for large datasets, not optimized for storage or transmission efficiency, and can be verbose for simple data structures.
Examples	[{ "name": "John", "age": 30, "city": "New York" }, { "name": "Jane", "age": 25, "city": "San Francisco" }, { "name": "Bob", "age": 40, "city": "Los Angeles" }]

- A text-based open standard designed for human-readable data interchange.
- Derived from the JavaScript scripting language, JSON is a language for representing simple data structures and associative arrays, called objects.
- Despite its relationship to JavaScript, JSON is language-independent, with parsers available for many languages.

Understanding HTTP JSON API Server

-Let's consider below question.

1. Create an HTTP server in Node.js that responds with JSON data.
2. The server should handle **two specific endpoints**:
 - **/api/parsetime**: The client will send a query string with an ISO-format date (e.g., `?iso=2013-08-10T12:10:15.474Z`), and the server should return the hour, minute, and second from that date as a JSON object.
 - **/api/unixtime**: The server should return the UNIX timestamp (epoch time) for the same ISO-format date.
3. The server should **listen on a port** passed as the first argument when you run the program.

```
var http = require('http')
var url = require('url')

// - Expect the request to contain a query
// string with a key 'iso' and an ISO-format time as
// the value. For example
// /api/parsetime?iso=2013-08-10T12:10:15.474Z
// - The JSON response should contain only 'hour', 'minute'
// and 'second' properties. For example:
// {
//   "hour": 14,
//   "minute": 23,
//   "second": 15
// }
function parsetime (time) {
  return {
    hour: time.getHours(),
    minute: time.getMinutes(),
    second: time.getSeconds()
  }
}

// Add second endpoint for the path '/api/unixtime' which
// accepts the same query string but returns UNIX epoch
// time under the property 'unixtime'. For example:
// {
//   "unixtime": 1376136615474
// }
function unixtime (time) {
  return { unixtime : time.getTime() }
}

var server = http.createServer(function (req, res) {
  // req.url = /api/parsetime?iso=2013-08-10T12:10:15.474Z
  // or
  // req.url = /api/unixtime?iso=2013-08-10T12:10:15.474Z
  var parsedUrl = url.parse(req.url, true)

  // time = 2013-08-10T12:10:15.474Z
  var time = new Date(parsedUrl.query.iso)
  var result

  // match req.url with the string /api/parsetime
  if (/^\/api\/parsetime/.test(req.url))
    // e.g., of time "2013-08-10T12:10:15.474Z"
    result = parsetime(time)
  // match req.url with the string /api/unixtime
  else if (/^\/api\/unixtime/.test(req.url))
    result = unixtime(time)

  if (result) {
    res.writeHead(200, { 'Content-Type': 'application/json' })
    res.end(JSON.stringify(result))
  } else {
    res.writeHead(404)
    res.end()
  }
})

server.listen(Number(process.argv[2]))
```

Step 1: On the server use below command

`$ node http_json_api_server.js 8000`

Step 2: on the client browser enter

`http://localhost:8000/api/parsetime?iso=2013-08-10T12:10:15.474Z`

Step 3: on the client browser enter

`http://localhost:8000/api/unixtime?iso=2013-08-10T12:10:15.474Z`

Output

The first screenshot shows a browser window with the address bar containing `localhost:8000/api/parsetime?iso=2013-08-10T12:10:15.474Z`. The page displays the JSON response: `{ "hour": 12, "minute": 10, "second": 15 }`.

The second screenshot shows a browser window with the address bar containing `localhost:8000/api/unixtime?iso=2013-08-10T12:10:15.474Z`. The page displays the JSON response: `{ "unixtime": 1376136615474 }`.

Modifying HTTP JSON API Server to get the current time

- We will get the current time by modifying the JSON file previously used . Few changes in the parsetime() function will give us our required output.

```
function parsetime(time) {  
  return {  
    year: time.getFullYear(),  
    month: String(time.getMonth() + 1).padStart(2, '0'), // Months are 0-based  
    date: String(time.getDate()).padStart(2, '0'),  
    hour: String(time.getHours()).padStart(2, '0'),  
    minute: String(time.getMinutes()).padStart(2, '0') ; }  
}
```

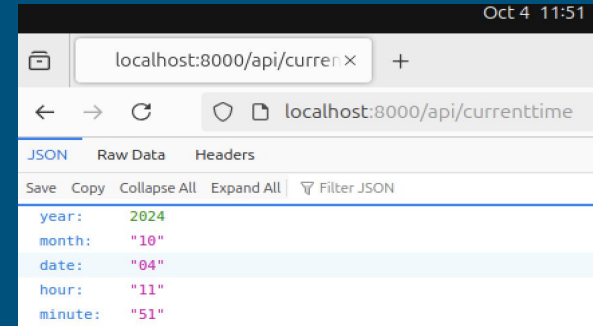
- And then adding below logic.

```
// Handle /api/parse_currenttime  
else if (/^\/api\/parse_currenttime\/.test(req.url)) {  
  var currentTime = new Date(); // Get the current time  
  result = parseCurrentTime(currentTime); // Use the current time parsing function }
```

On the client browser enter:

<http://localhost:8000/api/currenttime>

Output



Thank You!

