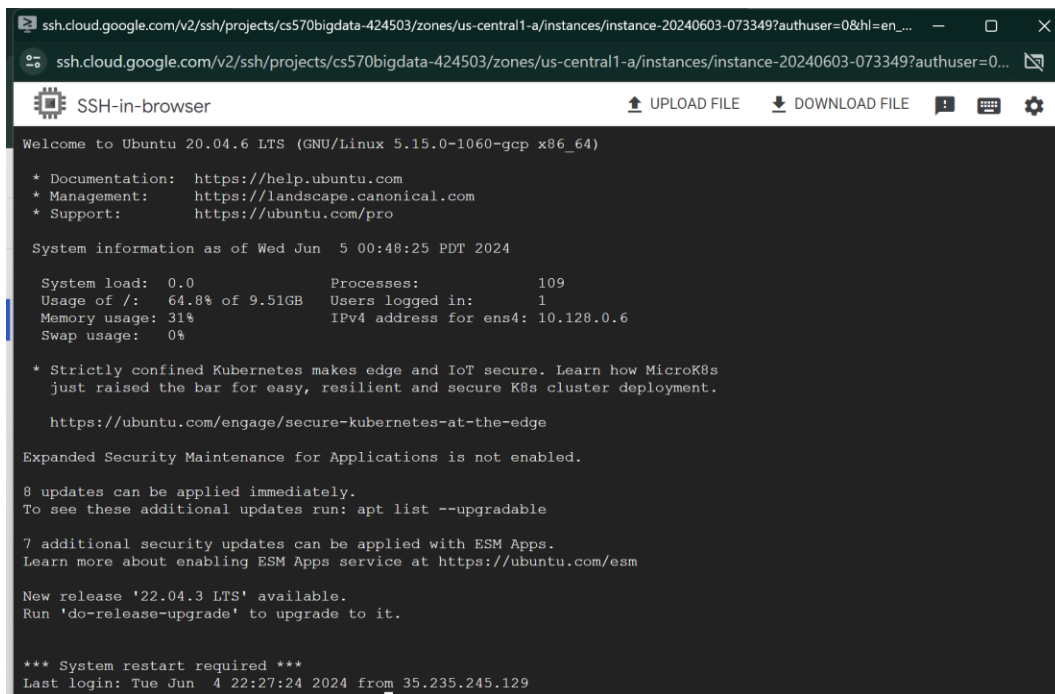
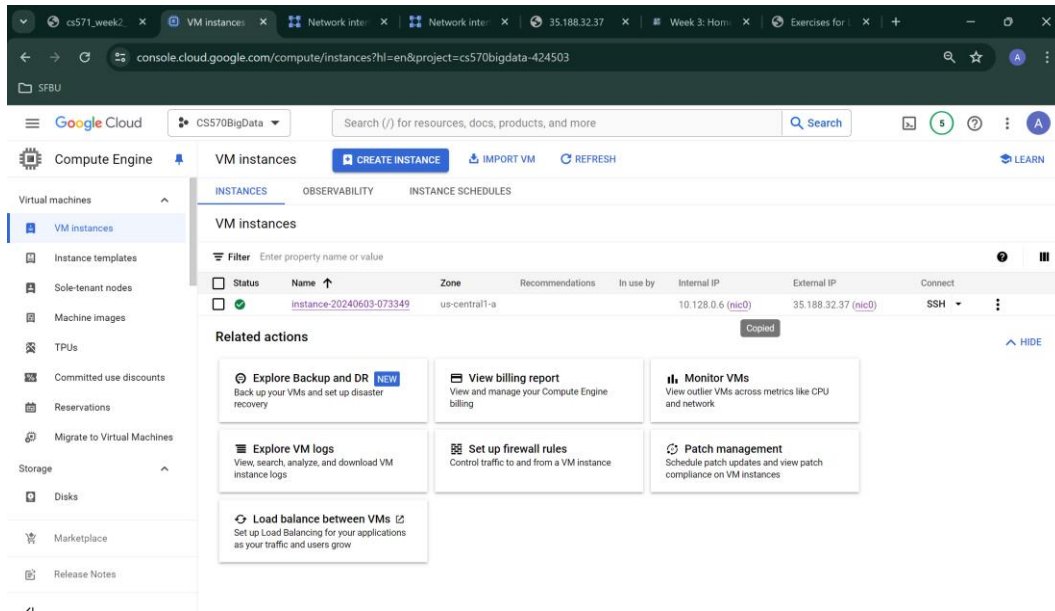


## Step 1: Install Node.js on Ubuntu

For this project, I am going to use a Google Cloud Platform (GCP) virtual instance with an Ubuntu operating system. First, I will SSH to connect to the virtual machine.



\$ sudo apt install nodejs

```
adagniew407@instance-20240603-073349:~$ sudo apt install nodejs
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer require
d:
  genders libgenders0
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  libc-ares2 libnode64 nodejs-doc
Suggested packages:
  npm
The following NEW packages will be installed:
  libc-ares2 libnode64 nodejs nodejs-doc
0 upgraded, 4 newly installed, 0 to remove and 8 not upgraded.
Need to get 6806 kB of archives.
After this operation, 30.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
Get:1 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/main amd
64 libc-ares2 amd64 1.15.0-1ubuntu0.5 [36.9 kB]
Get:2 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe
amd64 libnode64 amd64 10.19.0~dfsg-3ubuntu1.6 [5764 kB]
Get:3 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe
amd64 nodejs-doc all 10.19.0~dfsg-3ubuntu1.6 [943 kB]
Get:4 http://us-central1.gce.archive.ubuntu.com/ubuntu focal-updates/universe
amd64 nodejs amd64 10.19.0~dfsg-3ubuntu1.6 [61.6 kB]
Fetched 6806 kB in 0s (30.4 MB/s)
Selecting previously unselected package libc-ares2:amd64.
(Reading database ... 77798 files and directories currently installed.)
Preparing to unpack .../libc-ares2 1.15.0-1ubuntu0.5 amd64.deb ...
```

\$ node -v

```
adagniew407@instance-20240603-073349:~$ node -v
v10.19.0
```

\$ sudo apt install npm

```
adagniew407@instance-20240603-073349:~$ sudo apt install npm
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer require
d:
  genders libgenders0
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential cpp
  cpp-9 dpkg-dev fakeroot g++ g++-9 gcc gcc-9 gcc-9-base gyp
  javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libasan5 libatomic1 libauthen-sasl-perl
  libbinutils libc-dev-bin libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0
  libctf0 libdata-dump-perl libdpkg-perl libencode-locale-perl libfakeroot
  libfile-basedir-perl libfile-desktopentry-perl libfile-fcntllock-perl
  libfile-listing-perl libfile-mimeinfo-perl libfont-afm-perl libgcc-9-dev
  libgomp1 libhtml-form-perl libhtml-format-perl libhtml-parser-perl
  libhtml-tagset-perl libhtml-tree-perl libhttp-cookies-perl
  libhttp-daemon-perl libhttp-date-perl libhttp-message-perl
  libhttp-negotiate-perl libio-html-perl libio-socket-ssl-perl
  libio-stringy-perl libipc-system-simple-perl libisl22 libitm1
  libjs-inherits libjs-is-typedarray libjs-psl libjs-typedarray-to-buffer
  liblsan0 liblwp-mediatypes-perl liblwp-protocol-https-perl
  libmailtools-perl libmpc3 libnet-dbus-perl libnet-http-perl
  libnet-smtp-ssl-perl libnet-ssleay-perl libnode-dev libpython2-stdlib
```

## Step 2: Study Time Server

In Node.js, the Date object is used to work with dates and times. It provides various methods for retrieving and manipulating date and time information. For example, the `getFullYear()`, `getMonth()`, `getDate()`, `getHours()`, and `getMinutes()` methods can be used to extract individual components of the current date and time.

Reference:

[https://hc.labnet.sfbu.edu/~henry/npu/classes/javascript/node\\_js/course/nodeschool/learnyounode/time\\_server.html](https://hc.labnet.sfbu.edu/~henry/npu/classes/javascript/node_js/course/nodeschool/learnyounode/time_server.html)

## Step 3: Study JSON

JSON, or JavaScript Object Notation, is a widely used format for storing and exchanging data. Its simplicity, based on key/value pairs, lists, and scalars, aligns well with the data structures of dynamic programming languages, making it easy to work with. Unlike XML, JSON does not require a schema, enabling straightforward data interchange and processing.

Reference: [https://hc.labnet.sfbu.edu/~henry/npu/classes/javascript/json/slide/index\\_slide.html](https://hc.labnet.sfbu.edu/~henry/npu/classes/javascript/json/slide/index_slide.html)

## Step 4: Study HTTP JSON API Server

Create a file called `timeServer.js`, and write the content of the js file.

### 1. Imports Required Modules:

- `http`: Used to create the HTTP server.
- `url`: Used to parse the URL and query parameters.

### 2. Defines Utility Functions:

- `zeroFill(i)`: Adds a leading zero to numbers less than 10.
- `now()`: Returns the current date and time as an object with year, month, date, hour, and minute.
- `parsetime(time)`: Parses a Date object and returns an object with hour, minute, and second.
- `unixtime(time)`: Converts a Date object to UNIX epoch time and returns it.

### 3. Creates an HTTP Server:

- Parses the incoming request URL to determine the endpoint and query parameters.
- Handles requests to three main endpoints:
  - `/api/currenttime`: Responds with the current date and time in JSON format.

- /api/parsetime: Parses the provided ISO-format time and responds with hour, minute, and second.
- /api/unixtime: Converts the provided ISO-format time to UNIX epoch time and responds with it.
- For other endpoints, it responds with a plain text representation of the current date and time or an error message.

#### 4. Starts the Server:

- Listens on a port provided as a command-line argument.
- Logs a message indicating that the server is running.:wq::

## Step 5: Modify HTTP JSON API Server to support this request from the client side

\$ vi timeServer.js

```
adagniew407@instance-20240603-073349:~$ vi timeServer.js
```

Now based what we studied in step 4 and 5 lets modify the code.

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

// Function to add leading zero to numbers less than 10
function zeroFill(i) {
  return (i < 10 ? '0' : '') + i;
}

// Function to format the current date and time
function now() {
  var d = new Date();
  return {
    year: d.getFullYear(),
    month: zeroFill(d.getMonth() + 1),
    date: zeroFill(d.getDate()),
    hour: zeroFill(d.getHours()),
    minute: zeroFill(d.getMinutes())
  };
}

// Function to parse time into hour, minute, and second
function parsetime(time) {
  return {
```

```
    hour: time.getHours(),
    minute: time.getMinutes(),
    second: time.getSeconds()
  };
}

// Function to get UNIX epoch time
function unixtime(time) {
  return { unixtime: time.getTime() };
}

// Create an instance of the HTTP server to handle HTTP requests
let server = http.createServer((req, res) => {
  const parsedUrl = url.parse(req.url, true);
  const time = new Date(parsedUrl.query.iso);
  let result;

  if (req.url === '/api/currenttime') {
    result = now();
  } else if (/^\/api\/parsetime/.test(req.url)) {
    result = parsetime(time);
  } else if (/^\/api\/unixtime/.test(req.url)) {
    result = unixtime(time);
  } else {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    const nowObj = now();
    result = `${nowObj.year}-${nowObj.month}-${nowObj.date}
    ${nowObj.hour}:${nowObj.minute}`;
    //return;
  }

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

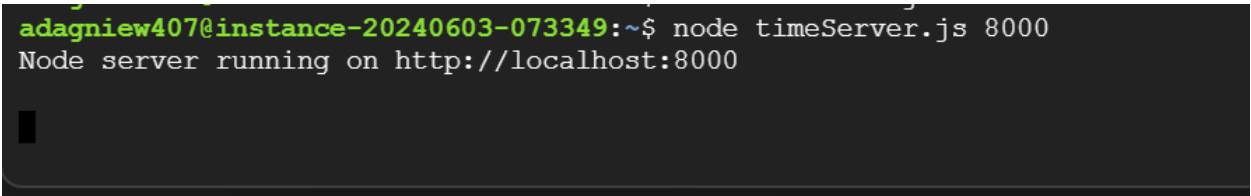
// Start the server listening on the port provided on the command line
server.listen(Number(process.argv[2]), () => {
  console.log('Node server running on http://localhost:' + process.argv[2] +
  '\n');
});
```

## Step 6: Test the code

Since we don't have a GUI we are going to use a second SSH terminal to check the results of the website below:

First lets run the timeServer.js:

```
$ node timeServer.js 8000
```



```
adagniew407@instance-20240603-073349:~$ node timeServer.js 8000
Node server running on http://localhost:8000
```

Then, lets open another SSH terminal:

we are going to use:

```
$ curl http://localhost:8000/currenttime which will give like the below result
```

```
*** System restart required ***
Last login: Wed Jun  5 00:48:26 2024 from 35.235.244.32
adagniew407@instance-20240603-073349:~$ curl http://localhost:8000/curenttime
"2024-06-05 01:13"
```

The image shows two side-by-side SSH-in-browser windows. The left window displays system update information for Ubuntu 20.04.6 LTS, including a message about a system restart required. The right window shows the same system information but with a Node.js error message: 'RangeError [ERR\_SOCKET\_BAD\_PORT]: Port should be >= 0 and < 65536. Received NaN.' The error message is followed by a stack trace. The bottom of the image shows a Windows taskbar with the date and time 11:13 AM 6/5/2024.

```
*** System restart required ***
Last login: Tue Jun  4 22:27:24 2024 from 35.235.245.129
adagniew407@instance-20240603-073349:~$ ls
InvertedIndex  PiCalculation  hadoop-3.4.0  hadoop-3.4.0.tar.gz  student_id.js  timeSe
rver.js
adagniew407@instance-20240603-073349:~$ vi timeServer.js
adagniew407@instance-20240603-073349:~$ vi timeServer.js
adagniew407@instance-20240603-073349:~$ node timeServer.js
net.js:1406
    throw new ERR_SOCKET_BAD_PORT(options.port);
          ^
RangeError [ERR_SOCKET_BAD_PORT]: Port should be >= 0 and < 65536. Received NaN.
    at Object.<anonymous> (/home/adagniew407/timeServer.js:64:8)
    at Module.compile (internal/modules/cjs/loader.js:778:30)
    at Object.Module._extensions..js (internal/modules/cjs/loader.js:789:10)
    at Module.load (internal/modules/cjs/loader.js:653:32)
    at tryModuleLoad (internal/modules/cjs/loader.js:593:12)
    at Function.Module._load (internal/modules/cjs/loader.js:585:3)
    at Function.Module.runMain (internal/modules/cjs/loader.js:831:12)
    at startup (internal/bootstrap/node.js:283:19)
    at bootstrapNodeCore (internal/bootstrap/node.js:623:3)
adagniew407@instance-20240603-073349:~$ vi timeServer.js
adagniew407@instance-20240603-073349:~$ node timeServer.js 8000
Node server running on http://localhost:8000
```

```
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.15.0-1060-gcp x86_64)

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

System information as of Wed Jun  5 01:13:14 PDT 2024

System load:  0.01               Processes:    108
Usage of /:   64.8% of 9.51GB    Users logged in: 1
Memory usage: 31%               IPv4 address for ens4: 10.128.0.6
Swap usage:   0%

* Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
  just raised the bar for easy, resilient and secure K8s cluster deployment.
  https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

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

7 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Wed Jun  5 00:48:26 2024 from 35.235.244.32
adagniew407@instance-20240603-073349:~$ curl http://localhost:8000/curenttime
"2024-06-05 01:13"
adagniew407@instance-20240603-073349:~$
```