HTTP JSON API Node.js Time Server

Belsabel Woldemichael

TABLE OF CONTENTS

Introduction

Objective: Create an HTTP JSON API server using Node.js to provide the current date and time in JSON format.

Goal: Develop a simple HTTP server accessible via a specific endpoint, returning time data formatted in JSON.

Skills Demonstrated:

- Node.js for building lightweight and efficient API servers.
- Handling HTTP requests and responses.
- JSON for data interchange.

Design

1. System Overview

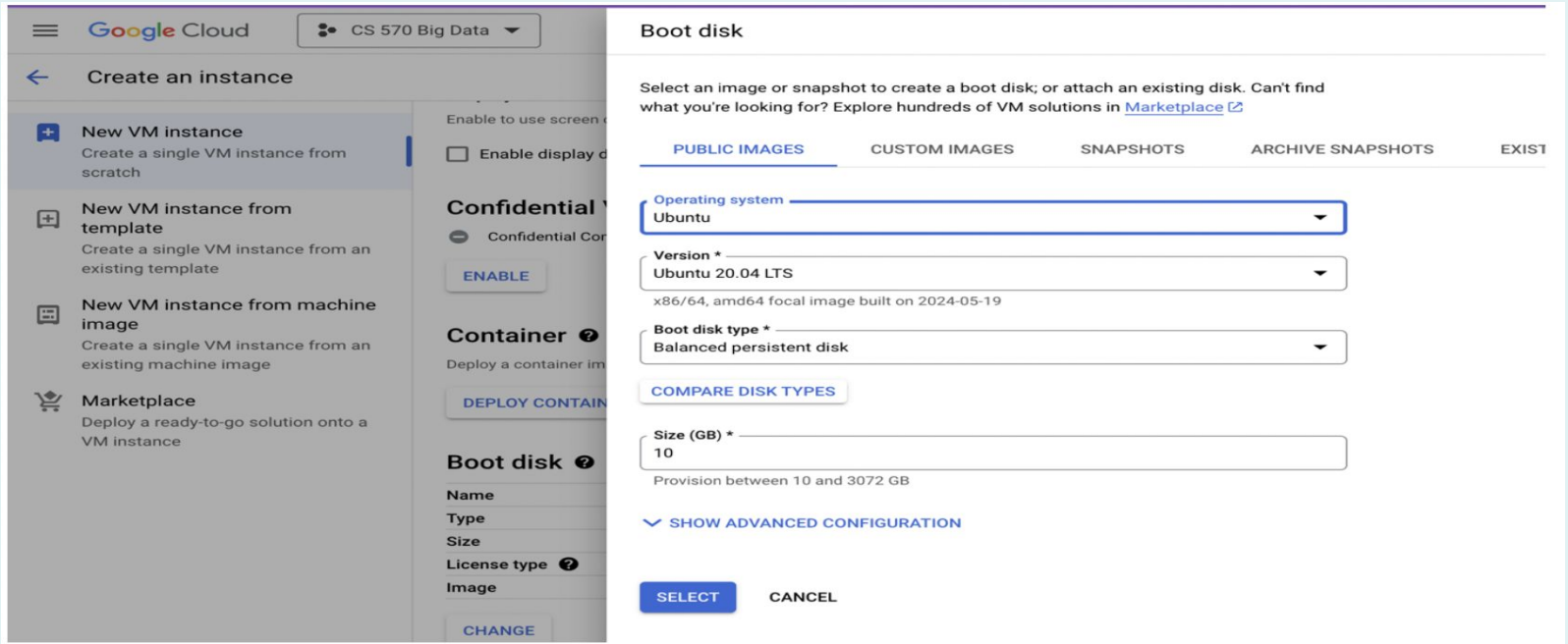
- **Client**: A web browser or any HTTP client that sends a request to the server.
- **Server**: A Node.js application that listens for HTTP requests and responds with the current date and time in JSON format.

2. Components

- **Node.js**: The runtime environment used to build the server.
- **HTTP Module**: Built-in Node.js module to create the server and handle HTTP requests.
- **JSON**: Format for the response data.

Implementation

1. Create an instance on GCP project with Ubuntu OS



2. Set Up Your Environment

- Update your package index

```
permitted by applicable law.
belsabelteklemariam@ubuntu:~$ sudo apt update
Get:1 file:/etc/apt/mirrors/debian.list Mirrorlist [30 B]
Get:3 file:/etc/apt/mirrors/debian-security.list Mirrorlist [39 B]
Get:2 https://deb.debian.org/debian bookworm InRelease [151 kB]
Get:7 https://packages.cloud.google.com/apt google-compute-engine-bookworm-stable InRelease [1321 B]
Get:4 https://deb.debian.org/debian bookworm-updates InRelease [55.4 kB]
Get:5 https://deb.debian.org/debian bookworm-backports InRelease [56.5 kB]
Get:6 https://deb.debian.org/debian-security bookworm-security InRelease [48.0 kB]
Get:8 https://packages.cloud.google.com/apt cloud-sdk-bookworm InRelease [1652 B]
Get:9 https://packages.cloud.google.com/apt google-compute-engine-bookworm-stable/main amd64 Packages [3128 B]
Get:10 https://deb.debian.org/debian bookworm-backports/main Sources.diff/Index [63.3 kB]
```

- Install Node.js and npm

```
belsabelteklemariam@ubuntu:~$ sudo apt install nodejs npm -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential bzip2 cpp cpp-12 dpkg-dev eslint fakeroot fontconfig-config fonts-dejavu-core g++ g++-12 gcc gcc-12 git
  git-man gyp handlebars javascript-common libabsl20220623 libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl libaom3 libasan8 libatomic1
  libauthen-sasl-perl libavif15 libbinutils libc-ares2 libc-dev-bin libc-devtools libc6-dev libcc1-0 libclone-perl libcrypt-dev libctf-nobfd0 libctf0 libdata-dump-perl libdav1d
  libde265-0 libdeflate0 libdpkg-perl libdrm-amdgpu1 libdrm-common libdrm-intel1 libdrm-nouveau2 libdrm-radeon1 libdrm2 libegl-mesa0 libegl1 libencode-locale-perl liberror-perl
  libfakeroot libfile-basedir-perl libfile-desktopentry-perl libfile-fcntllock-perl libfile-listing-perl libfile-mimeinfo-perl libfont-afm-perl libfontconfig1 libfontenc1
  libgav1-1 libgbm1 libgcc-12-dev libgd3 libgdk-pixbuf-2.0-0 libgdk-pixbuf2.0-bin libgdk-pixbuf2.0-common libgl1 libgl1-mesa-dri libglapi-mesa libgles2 libglvnd0 libglx-mesa0
  libglx0 libgomp1 libgprofng0 libheif1 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 libice6 libio-html-perl libio-socket-ssl-perl libio-stringy-perl libipc-system-simple-perl libisl23 libitm1
  libjansson4 libjbig0 libjpeg62-turbo libjs-async libjs-events libjs-inherits libjs-is-typedarray libjs-prettify libjs-regenerate libjs-source-map libjs-sprintf-js
  libjs-typedarray-to-buffer libjs-util liblerc4 libllvm15 liblocale-gettext-perl liblsan0 liblwp-mediatypes-perl liblwp-protocol-https-perl libmailtools-perl libmpc3 libmpfr6
  libnet-dbus-perl libnet-http-perl libnet-smtp-ssl-perl libnet-ssleay-perl libnode-dev libnode108 libnotify-bin libnotify4 libnsl-dev libnuma1 libpciaccess0 libquadmath0
```

- Verify Installation

```
belsabelteklemariam@ubuntu:~$ node -v
v18.19.0
belsabelteklemariam@ubuntu:~$ npm -v
9.2.0
```

3. Create project directory

```
belsabelteklemariam@ubuntu:~$ mkdir time-server
belsabelteklemariam@ubuntu:~$ cd time-server
```

4. Initialize Node.js

```
belsabelteklemariam@ubuntu:~/time-server$ npm init -y
Wrote to /home/belsabelteklemariam/time-server/package.json:

{
  "name": "time-server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

5. Create server.js file

```
belsabelteklemariam@ubuntu:~/time-server$ touch server.js
belsabelteklemariam@ubuntu:~/time-server$ nano server.js
```

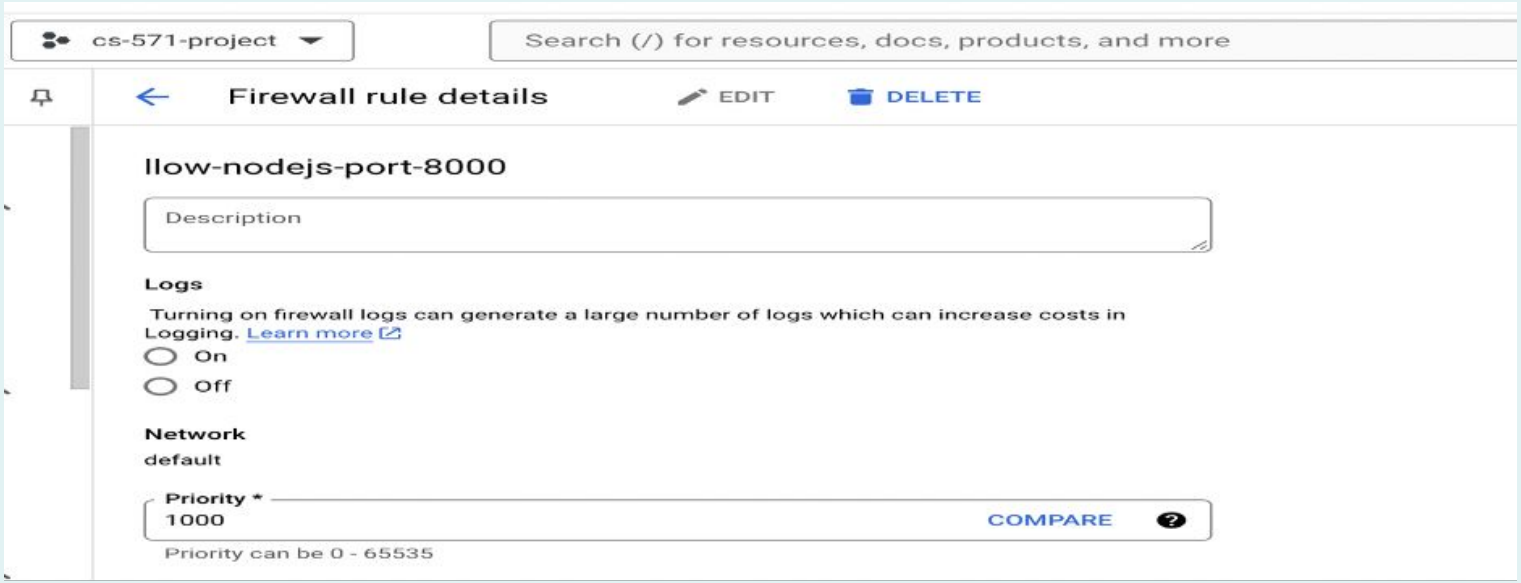
6. Put the code in server.js

```
  GNU nano 7.2                                                    server.js *
const http = require('http');
const url = require('url');

// Function to get the current time and format it as a JSON object
function getCurrentTime() {
    const now = new Date();
    return {
        year: now.getFullYear(),
        month: ('0' + (now.getMonth() + 1)).slice(-2), // Months are zero-based, add 1 and pad with zero
        date: ('0' + now.getDate()).slice(-2),         // Pad with zero
        hour: ('0' + now.getHours()).slice(-2),        // Pad with zero
        minute: ('0' + now.getMinutes()).slice(-2)     // Pad with zero
    };
}

// Create the HTTP server
const server = http.createServer((req, res) => {
    const parsedUrl = url.parse(req.url, true);

    // Check if the request is for the current time API
    if (parsedUrl.pathname === '/api/currenttime') {
        const currentTime = getCurrentTime();
        res.writeHead(200, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify(currentTime));
    } else {
        res.writeHead(404, { 'Content-Type': 'application/json' });
        res.end(JSON.stringify({ error: 'Endpoint not found' }));
    }
});

// Start the server
const port = 8000;
server.listen(port, () => {
    console.log(`Server running at http://localhost:${port}/`);
});
```

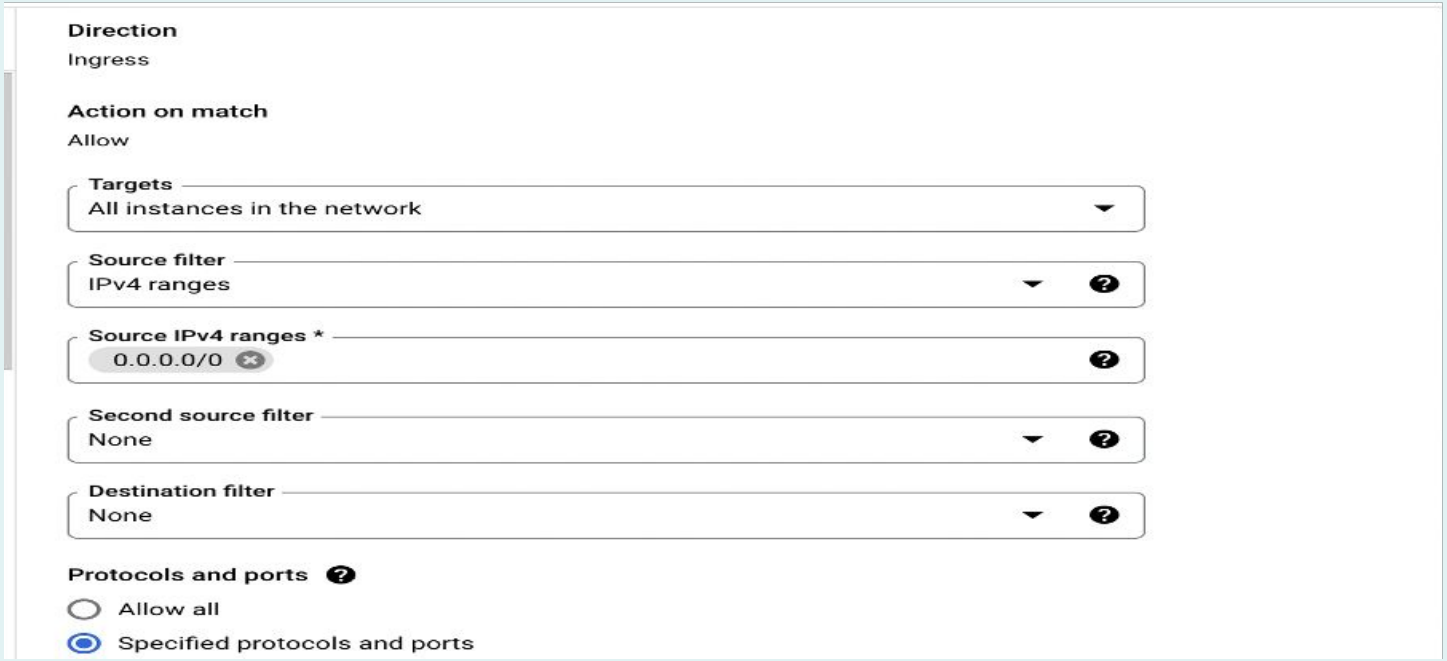
Check Firewall Rules

Ensure that your VM instance allows incoming connections on port 8000. You might have already set this up, but to verify:

- Name: allow-node js-port-8000



- Targets: All instances in the network
- Source IP ranges: 0.0.0.0/0



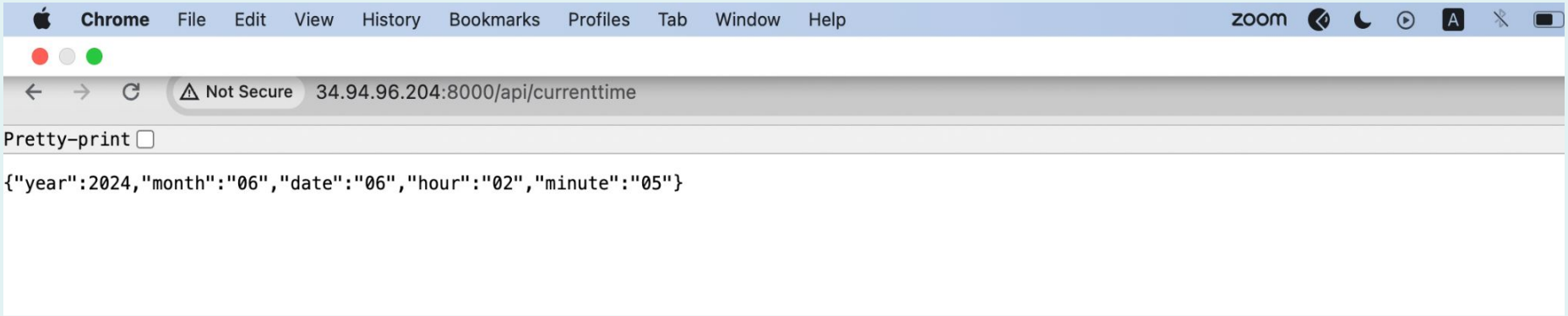
- Protocols and ports: Specified protocols and ports: tcp:8000

Test

6. Run the server.js file

```
belsabelteklemariam@ubuntu:~/time-server$ node server.js
Server running at http://localhost:8000/
```

7. Put http://34.94.96.204:8000/api/currenttime to request from client side
   - The browser displays the below output



Enhancement Ideas

- Enhance the existing time server to support different time zones and allow clients to request the current time in different formats.
- Allow clients to specify the desired date format (e.g., YYYY-MM-DD, DD/MM/YYYY).
- Allow clients to request the time for a specific past date and time.
- Add query parameters to specify the date and time, and respond with the corresponding time details.
- Implement API key-based authentication to restrict access to authorized users.

Conclusion

The HTTP JSON API Node.js Time Server project demonstrates the fundamental capabilities of Node.js for building efficient and lightweight web servers. By developing this server, we achieved several key objectives:

- **Basic Functionality**: We successfully created an HTTP server that responds with the current date and time in JSON format. This basic functionality showcases how to handle HTTP requests and format responses in Node.js.
- **Structured Process**: The step-by-step approach—from installing Node.js to modifying the server for specific client requests—ensured a clear and logical development workflow.

Reference

HTTP JSON API Server

JSON

What is Node.js and how it works

Github link

https://github.com/BelsabelTekle/JavaScript/tree/main/Project2