**Hello world, using Node.js**

This tutorial assumes that all the MEAN stack is installed properly on google cloud **Ubuntu 18.04 LTS**

Node.js is a server-side version of JavaScript. That means all the things all them cool things about JavaScript apply here. It also means if you're already quite familiar with JavaScript you're going to have a nice advantage.

Let's create a hello world. Since we are using Linux in Google cloud, we can use the text editor nano to create hello.js file

**nano hello.js**

Type the following inside the file

**console.log("Hello World");**

Now save the file, and exit and run it with the following command:

**node hello.js**

So, you should get 'Hello World' appear in your terminal. That's all good but I'm sure more importantly you want to know how to print 'Hello World' to new HTTP connections.

Open the file hello.js by tying **nano hello.js** again and modify it by the following code.

**var http = require('http');**

**http.createServer(function (request, response) {**

**response.writeHead(200, {'Content-Type': 'text/plain'});**

**response.end('Hello World\n');**

**}).listen(3000);**

**console.log('Server started');**

Now save your file and run it with:

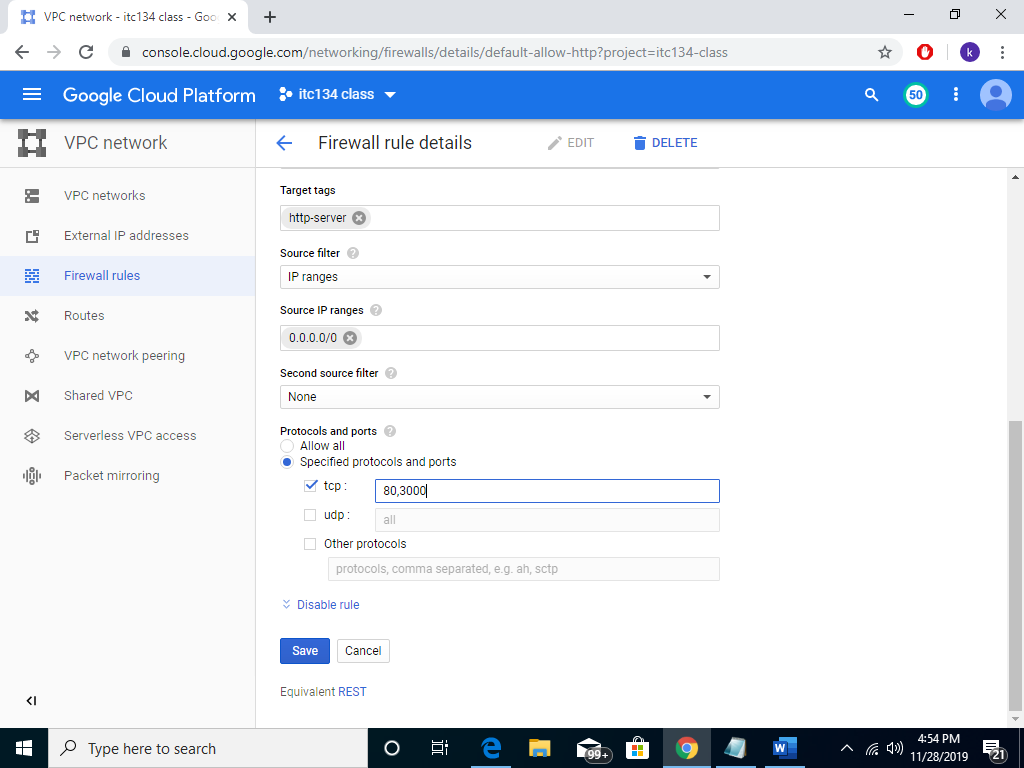
**node hello.js**

You should see 'Server started' in the terminal. Great!

To view the page in our browser we have to configure our port number (3000) using

1. Double click on instance that you created
2. Click on the navigation menu
3. Click on the VPC network
4. Click on the firewall rules
5. Click on default-allow http
6. Click on the edit
7. Modify the tcp/ip protocol 80,3000

**Note** port number in our case 3000 can be any number beside the reserved port numbers(1-1023)



8 Save and exit

Back to the console and clicked the external ip link and change the URL to be insecure(http) [http://35.227.140.10:3000](http://35.227.140.10:3000/) you should see your 'Hello World' message.

**Note** the ip address 35.227.140.10 could be vary depending on your external ip address

**Let's take a closer look at our code**.

The first line is just getting the http module and saving it to the variable 'http'. The http is included with Node.js to make it easy for us to create Node.js applications. We can then use the http module to create our http server by calling its function 'createServer'. This function returns an object and takes a function in a parameter.

We're calling the function 'listen' on our new server object which takes in a numeric value which tells our server what port we want it to listen on. In our case we're using port 3000 which is why we connected our browser to [**http://35.227.140.10:3000**](http://localhost:8080/)

We also create a function and use it as a parameter for the 'createServer' function. This is quite a standard thing to do in JavaScript because functions can be parameters just like variables and objects can be. What's going to happen is that every time our server receives and new connection on port 3000 it's going to run our function, we gave it. Interestingly the function we are passing to it is called an anonymous function, and it's called this because we don't give it a name.

You might have noticed our anonymous function takes two parameters, 'request' and 'response'. These parameters get passed to our anonymous function by the HTTP server when it receives a new connection. They are both objects which we can use in our response to the incoming request.

You notice the first thing we do is call the 'writeHead' function this lets us set the HTTP status as the first parameter and send some response headers as a second parameter. We're setting status code 200 which is telling our web browser everything's OK and we're also passing it a 'Content-Type' header which lets our browser know what we're sending it. In our case it's just plain text.

Next, we're using the response object to write our 'Hello World'. We do this by simply calling its written function and passing it our text. At this point we're done with our response, so we tell the response object by calling its 'end' function.