Author Contacts:

Tim Davidson: tdavidson61@gmail.com
Blake Agrade: blake.agrade@gmail.com
Bisrat Eshetu: bisremichael@gmail.com
Stephen Grose: stephen.grose@protonmail.com
Diego Falcon Costilla: diegofalconcostilla94@gmail.com

Creating a MEAN Stack Using Google Cloud Platform

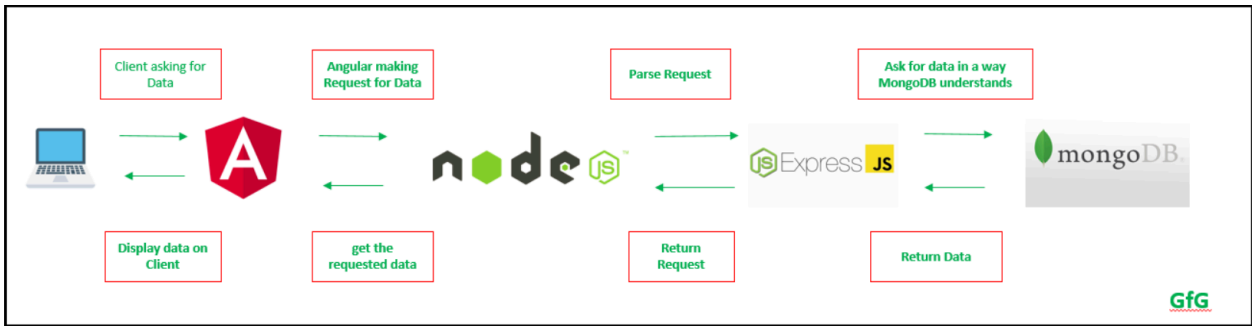
What is a MEAN stack?

A MEAN stack is an open-source framework used for developing dynamic websites and web applications. It is written in a single programming language, JavaScript, which helps to simplify the development of a project.

MEAN is an acronym which stands for: **M**ongoDB, **E**xpress.js, **A**ngularJS and **N**ode.js.

- MongoDB is the database where your information is stored.
- Express.js is the backend or server side framework. It handles all the interactions between the front end and the database.
- AngularJS is the front end or client-side framework and provides the point of contact for the user.
- Node.js provides a JavaScript runtime environment and serves as the backbone for the stack.

The diagram below illustrates the flow of a MEAN stack, from the user's initial request until they receive the data they requested:



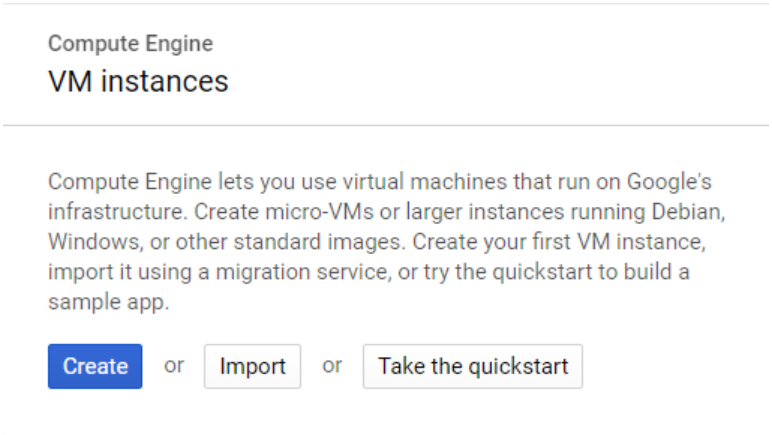
Hopefully you now have a basic understanding of what constitutes a MEAN stack. On the next page we take you through a tutorial that will teach you how to build one of your own.

How to Build a MEAN Stack

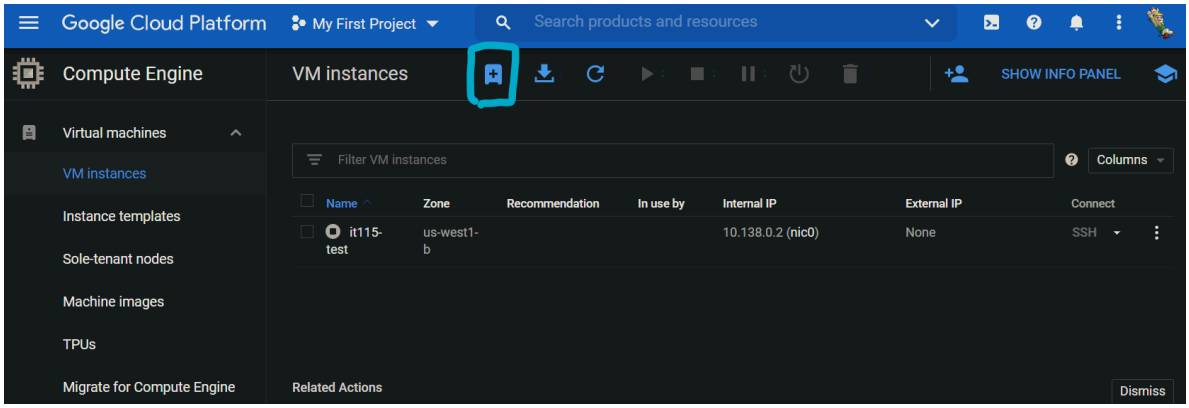
1. On your search engine, type: https://cloud.google.com. This will take you to the Google Cloud platform.
2. If you don't have an account, you can sign up for one for free. Otherwise, sign in to your existing account.
3. Click on **"Go to Console"**
4. On the Console page, you will see this in the upper left hand corner:



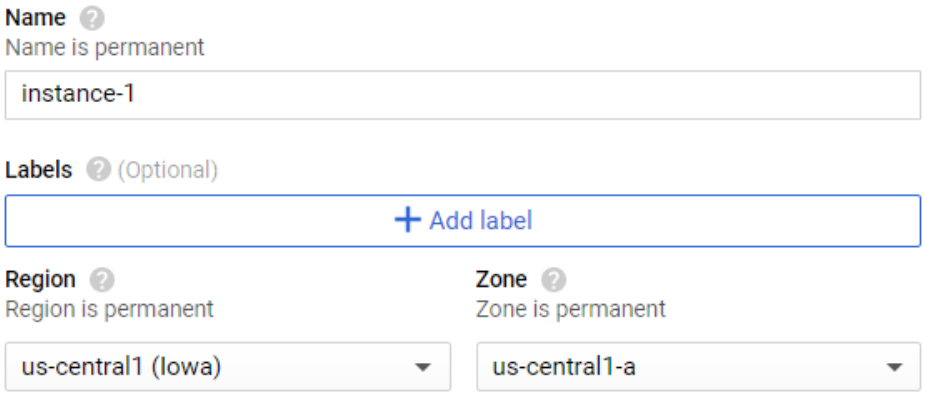
5. Click on the "hamburger" icon on the left and you will see a menu. Scroll down and hover your cursor over **"Compute Engine"** and you will see a drop-down menu. Click on **"VM Instances"**, at the top of that menu.
6. You will see this in the middle of your screen. Click on the blue "Create" rectangle.



If you do not see this screen, please click the small icon next to "VM instances" to start the VM creation process.

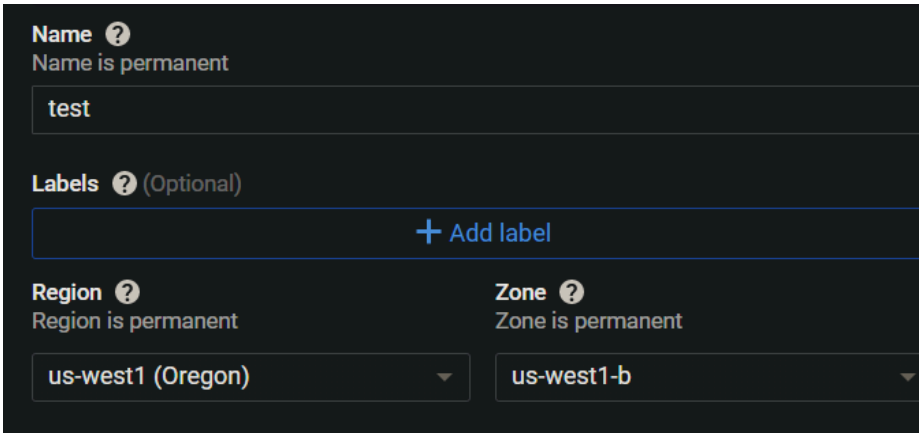


7. You will now see a screen with this at the top:

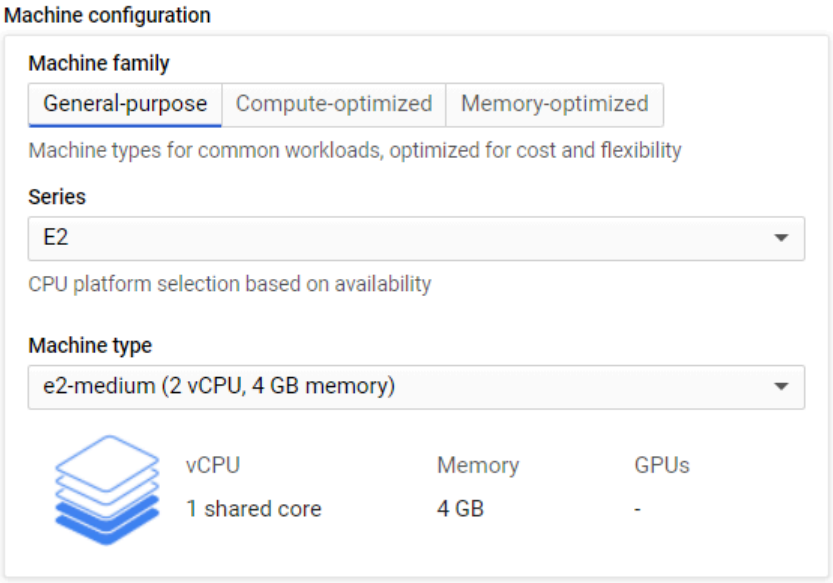


8. Change the name from “instance-1” to whatever you want to name your project. “test” will do for now.

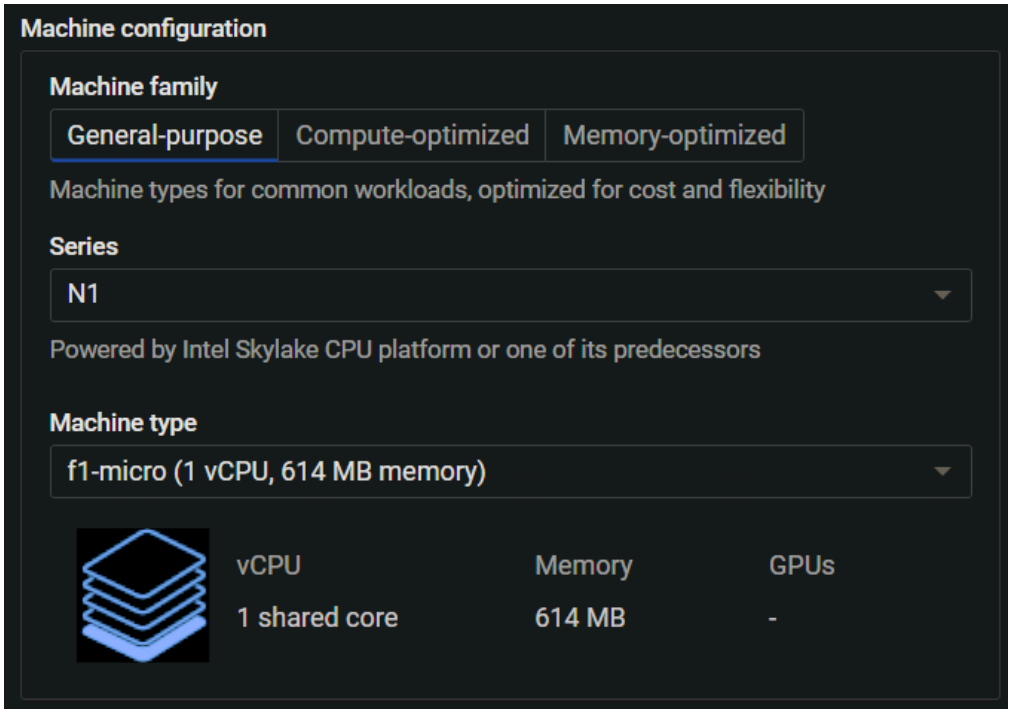
9. Change the region to something close to you, such as “us-west1 (Oregon)”.



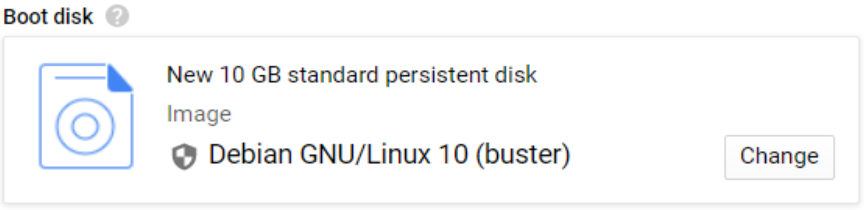
10. Scroll further down the screen until you see this:



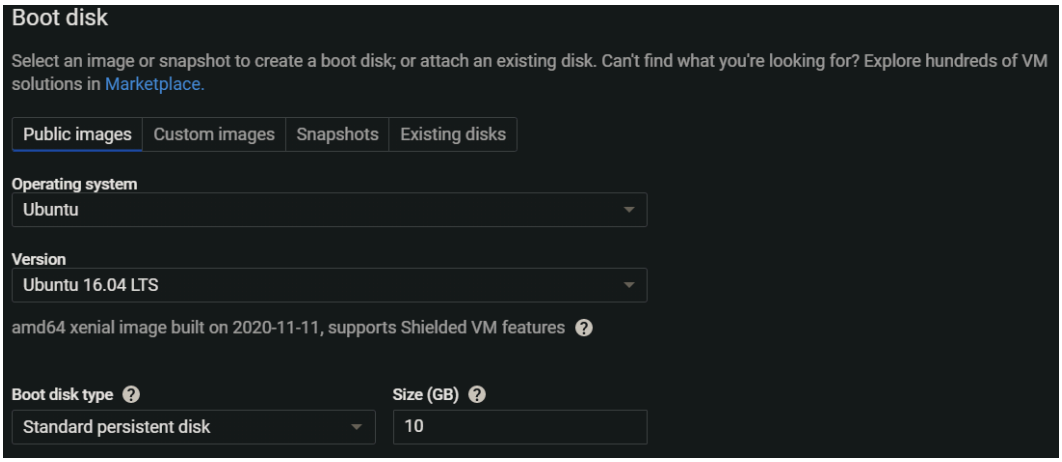
11. Using the drop-down arrow, change the Series to **N1** and the Machine type to **f1-micro**.



12. Then scroll down till you see:



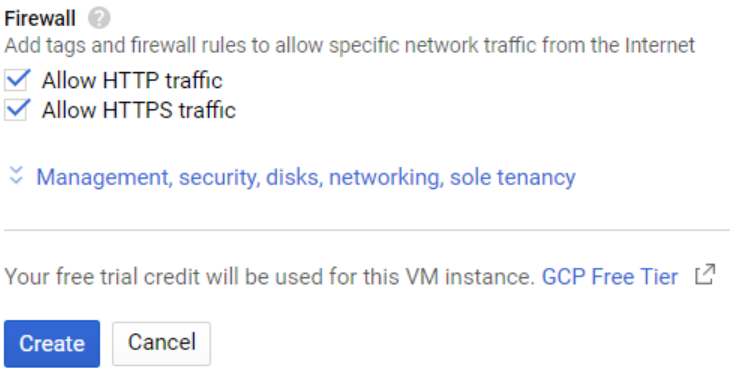
13. Click on “Change”. On the next screen, use the drop-down arrow to change to Ubuntu.



14. Then on the bottom left click:



15. Now you’re back on the previous screen. Scroll down to:

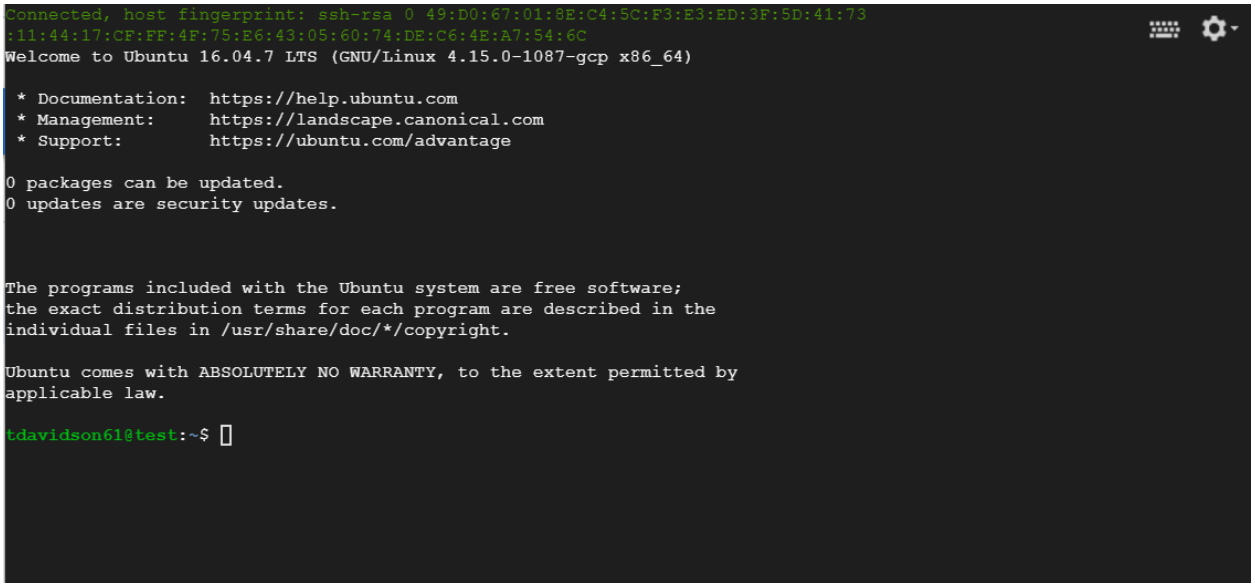


16. Check the Firewall boxes to allow both HTTP and HTTPS traffic. Then click “Create” and wait.

17. After several seconds, this should appear:

Name	Zone	Recommendation	In use by	Internal IP	External IP	Connect
test	us-west1-b			10.138.0.10 (nic0)	35.197.71.60	SSH

18. Go to the far right under “Connect” and click on SSH and wait for your connection. You will see this pop up window on your screen:



19. This is your command line shell. You now have a server on which to build your stack.

Installing Node.js

20. You can check here to see what the latest version is before installing: https://github.com/nodesource/distributions#debinstall

Installation instructions

Node.js v15.x:

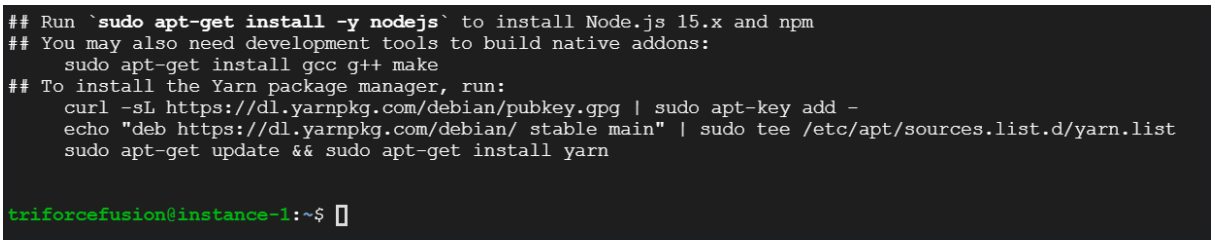
```
# Using Ubuntu
curl -sL https://deb.nodesource.com/setup_15.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root
curl -sL https://deb.nodesource.com/setup_15.x | bash -
apt-get install -y nodejs
```

21. Type (or copy and paste) in the following command to the command line and hit enter:

```
curl -sL https://deb.nodesource.com/setup_15.x | sudo -E bash -
```

After a lot of action, eventually you will see the following that means the command completed:



22. Wait for the command to finish. Type (or copy and paste) in the following command to the command line and hit enter to continue the installation:

```
sudo apt-get install -y nodejs
```

When the command line prompt is up again, if you would like, you can check your version for node.js and the npm (node package manager) by using the the following commands:

```
node -v
```

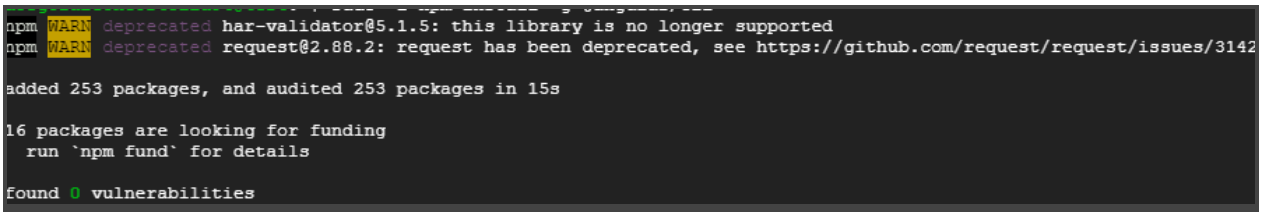
```
npm -v
```

Installing AngularJS

23. Type in the following command and hit enter:

```
sudo -i npm install -g @angular/cli
```

This will take a few seconds, Don't be scared by deprecation warnings, they are normal.



24. When the command line opens up, check your current angular/cli version by typing:

```
ng version
```

The following information will display if the tutorial has been followed as intended:

```
triforcefusion@instance-1:~$ ng version

     _                      _                 ____ _     ___
    / \   _ __   __ _ _   _| | __ _ _ __     / ___| |   |_ _|
   / △ \ | '_ \ / _` | | | | |/ _` | '__|   | |   | |    | |
  / ___ \| | | | (_| | |_| | | (_| | |      | |___| |___ | |
 /_/   \_\_| |_|\__, |\__,_|_|\__,_|_|       \____|_____|___|
                |___/


Angular CLI: 11.0.4
Node: 15.4.0
OS: linux x64

Angular:
...
Ivy Workspace:

Package                      Version
------------------------------------------------------------
@angular-devkit/architect    0.1100.4 (cli-only)
@angular-devkit/core         11.0.4 (cli-only)
@angular-devkit/schematics   11.0.4 (cli-only)
@schematics/angular          11.0.4 (cli-only)
@schematics/update           0.1100.4 (cli-only)

triforcefusion@instance-1:~$
```

Installing Express.js

25. Type the following command and hit enter to create a project folder:

    ```
    mkdir myapp
    ```

    If you don't see an error message and you are back at the command line, that means it worked.

26. To change the directory type the following command and hit enter:

    ```
    cd myapp
    ```

    Your command line will now look like this:

    ```
    triforcefusion@instance-1:~$ mkdir myapp
    triforcefusion@instance-1:~$ cd myapp
    triforcefusion@instance-1:~/myapp$
    ```

27. To initialize your project type the following command:

    ```
    npm init -y
    ```

28. Type and enter the following command to install express and save it:

    ```
    npm install express --save
    ```

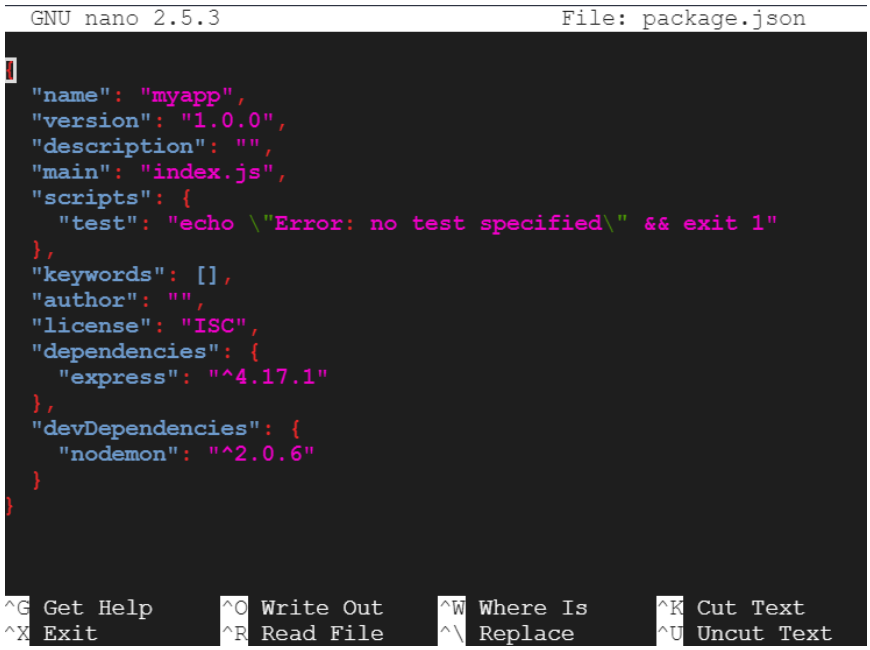
29. When the install finishes, type and enter:

    ```
    npm install -D nodemon
    ```

30. When the install finishes, type and enter:

```
nano package.json
```

You will now see this on your screen:



31. Hit the down arrow until the cursor is next to this line:

```
"test": "echo \"Error: no test specified\" && exit 1"
```

Hit the right arrow until the cursor is covering the first quotation mark.
Now hit “Delete” until

```
"test": "echo \"Error: no test specified\" && exit 1"
```

is removed.
DO NOT HOLD DOWN DELETE
Just erase the single line of code.

32. Select and copy following line of code
(highlight the following code with your cursor, then hold down **Ctrl** and press **C)**:

```
"start": "nodemon index.js"
```

33. Paste the line of code into the shell by placing the cursor in the shell where you deleted the single line of code, then hold down **Ctrl** and press **V.**

This will paste in the line of code you copied in step 32.
Your shell should look like this:



34. Hold down **Ctrl** and press **X** to exit. You will be prompted to save:
    Hit **Y** then **ENTER** to save.

35. Back at the command line, type in:
    ```
    nano index.js
    ```

36. Repeat the "copy and paste" method from steps 32 and 33 to place the following code inside the index.js shell:

    ```
    const express = require('express');
    const app = express();
    const PORT = 3000;
    app.get('/', (request, response) => {
     response.send('Hello');
    });
    app.use(express.json());
    app.listen(PORT, () => console.log(`Express server currently
    running on port ${PORT}`));
    ```

    Hold down **Ctrl** and hit **X** to exit. You will be prompted to save. Hit **Y** and then **ENTER** to save.

37. Type and enter on the command line:
    ```
    npm start
    ```

    You will see the following:

```
triforcefusion@instance-1:~/myapp$ npm start

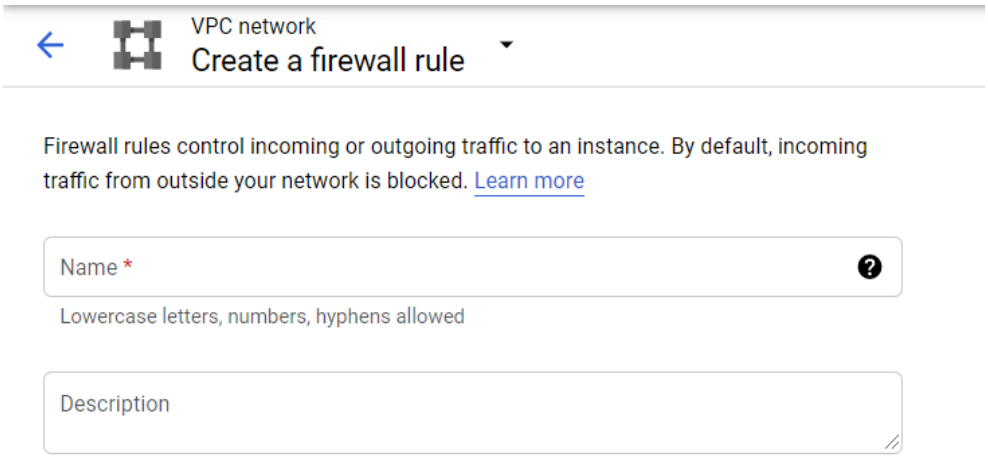
> myapp@1.0.0 start
> nodemon index.js

[nodemon] 2.0.6
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Express server currently running on port 3000
```

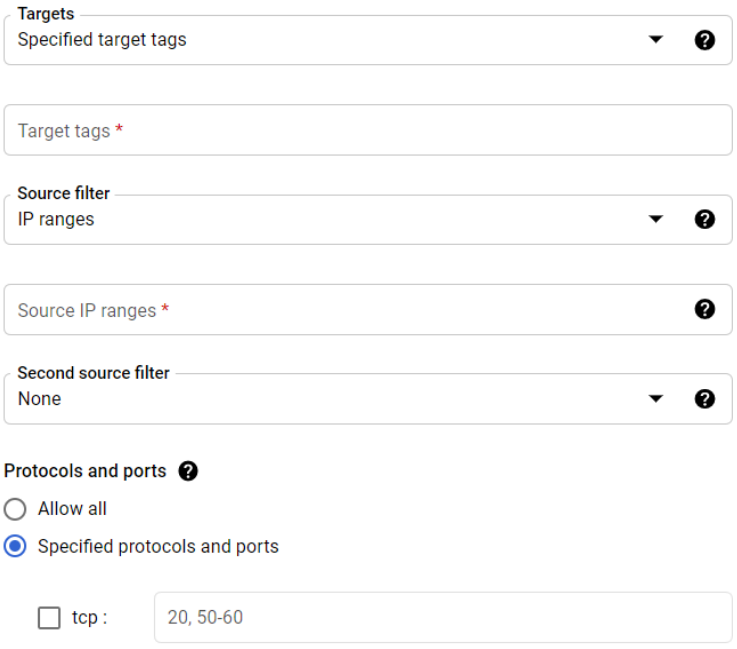
38. Next we have to create a rule that tells our Firewall security to open a port to allow access to the internet.
    Go back to the Google Cloud console.

    Find the magnifying glass icon on the blue search bar and click on it.
    Type in “firewall”. Click on **Firewall VPC network**

39. Near the top under the blue bar click **CREATE FIREWALL RULE.**
    The top part of your screen will look like this:



40. In the space marked “name” enter **“firewall-3000”.**
41. Scroll down until you see “Targets”. Go to the “Targets” rectangle and choose **“All instances in the network”** from the drop-down menu.



42. In the space marked “Source Ip ranges” type the following address and enter:
`0.0.0.0/0`

43. Scroll down to “Protocol and ports”. Click on **“Specified protocols and ports”** and check **“tcp”** and in the rectangle next to tcp type “3000” and hit the **CREATE** button on the bottom left. If you get a message that the name “firewall-3000” is already in use, go back to the previous screen that lists existing Firewall rules and click the box to the left of firewall-3000.

44. Go back to your VM instance; if you don’t remember how to do this, repeat steps 4 and 5 from the beginning of the tutorial. Click on the numbers underneath “External IP”.
A new tab will open where you will see a message that the site can’t be reached. Go to the URL address window.
Type “http://” in front of the IP address and “:3000” after it and hit enter.

    You should see “Hello” in the upper left corner of your screen. Leave this window open as you’ll be coming back to it later.

45. Go back to the command line shell. If it is not still on your screen, go back to your VM instance and click “SSH” again and wait.

46. Back at the shell, hold down **Ctrl** and type **C** in order to leave express
    Make sure the command line is ~/myapp$.
    Type and enter:

```
npm install connect serve-static
```

```
touch server.js
```

```
nano server.js
```

47. Copy and paste this into the server.js shell:

```
var connect = require('connect');
var serveStatic = require('serve-static');
connect()
    .use(serveStatic(__dirname))
    .listen(3000, () => console.log('Server running on
3000...'));
```

    Hold down **Ctrl** and hit **X** to exit. You will be prompted to save. Hit **Y** and then **ENTER** to save.

Installing MongoDB

48. Enter in the following command to move up in the directory:

```
cd ..
```

49. Install MongoDB with the following command:

```
sudo apt install -y mongodb
```

50. The DB will take a few moments to install. Wait for the command line to open back up to continue.

51. Check the install status with the following command:

```
sudo systemctl status mongodb
```

52. If it is working properly you will see the following:

```
triforcefusion@instance-1:~$ sudo systemctl status mongodb
● mongodb.service - An object/document-oriented database
   Loaded: loaded (/lib/systemd/system/mongodb.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-12-11 02:38:09 UTC; 14s ago
     Docs: man:mongod(1)
 Main PID: 4389 (mongod)
   CGroup: /system.slice/mongodb.service
           └─4389 /usr/bin/mongod --config /etc/mongodb.conf

Dec 11 02:38:09 instance-1 systemd[1]: Started An object/document-oriented database.
triforcefusion@instance-1:~$
```

*If your command line did not reappear, hold down **Ctrl** and hit **C** to bring it back.*

53. Check the connection to the MongoDB with the following command:

```
mongo --eval 'db.runCommand({ connectionStatus: 1,
showPrivileges: true})'
```

54. You should see something like the following to confirm it is connected:

```
triforcefusion@instance-1:~$ mongo --eval 'db.runCommand({ connectionStatus: 1, showPrivileges: true})'
MongoDB shell version: 2.6.10
connecting to: test
[object Object]
triforcefusion@instance-1:~$
```

*If your command line did not reappear, hold down **Ctrl** and hit **C** to bring it back.*

55. Start up the MongoDB with the following command:

```
sudo systemctl start mongodb
```

56. Nothing will appear to happen and that is okay. Enter in the next command to start working in the database:

```
mongo
```

57. Successful access to the database will result in this:

```
MongoDB shell version: 2.6.10
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
```

58. Notice the default database is “test.” Enter in sample data into the database named “test” with the following example data:

```
db.products.insert(
   [
     { _id: 11, item: "pencil", qty: 50, type: "no.2" },
     { _id: 12, item: "pen", qty: 20 },
     { _id: 13, item: "eraser", qty: 25 }
```

```
    ]
)
```

59. Ensure that your data is accessible by using the find() command:
    db.products.find()

60. You should see the following:

```
> db.products.find()
{ "_id" : 11, "item" : "pencil", "qty" : 50, "type" : "no.2" }
{ "_id" : 12, "item" : "pen", "qty" : 20 }
{ "_id" : 13, "item" : "eraser", "qty" : 25 }
```

61. Exit the database by typing **exit** and hitting **ENTER**.
    The Mongo database will say "bye" and you will be back at the command line.
    You now have a MEAN stack!

Thank you for using our tutorial. You can continue with our Web app tutorial.

If you choose not to continue with the next tutorial, return to your Google Cloud console VM instance page to stop your server from running when you're not using it. To do this, go the right side of your VM instance and left-click on the three vertical dots next to "SSH". You will get a menu with several choices; choose stop, suspend or delete depending on your needs. ALWAYS STOP OR SUSPEND your instance to prevent incurring charges when not in use.

Links to:

Github Website

Web app tutorial

Web App Github Repo

References

Introduction to MEAN Stack - GeeksforGeeks
An overview of the MEAN stack and a tutorial to run on a local machine

PhoenixNap - Unbuntu 18.x Install Tutorial

A straight-to-the-point tutorial on installing a MEAN stack on Ubuntu ver. 18.x+

Overview of MEAN Stack from MongoDB
Summarizes each part of the MEAN stack and gives you reasons why you would choose MEAN over another stack option.

IT115 FL20 - Google Cloud Platform & LAMP stack (Revised) - Google Docs
Google Cloud Platform tutorial that this stack tutorial was based on

Node.js GitHub
Official Node.js GitHub with lots of resources on how to install Node.js on different platforms, the license agreement, and FAQs

Command Line Interface (CLI) Overview - Official Angular Site
The overview page for Angular's Command Line Interface(CLI). This is from the official site and has many resources on using Angular.

Express.js Install Instructions - Official Site
Short 1 page summary to install Express.js. Node.js needs to be installed first.

MEAN Stack Tutorial
Previous class MEAN stack tutorial for guidance.