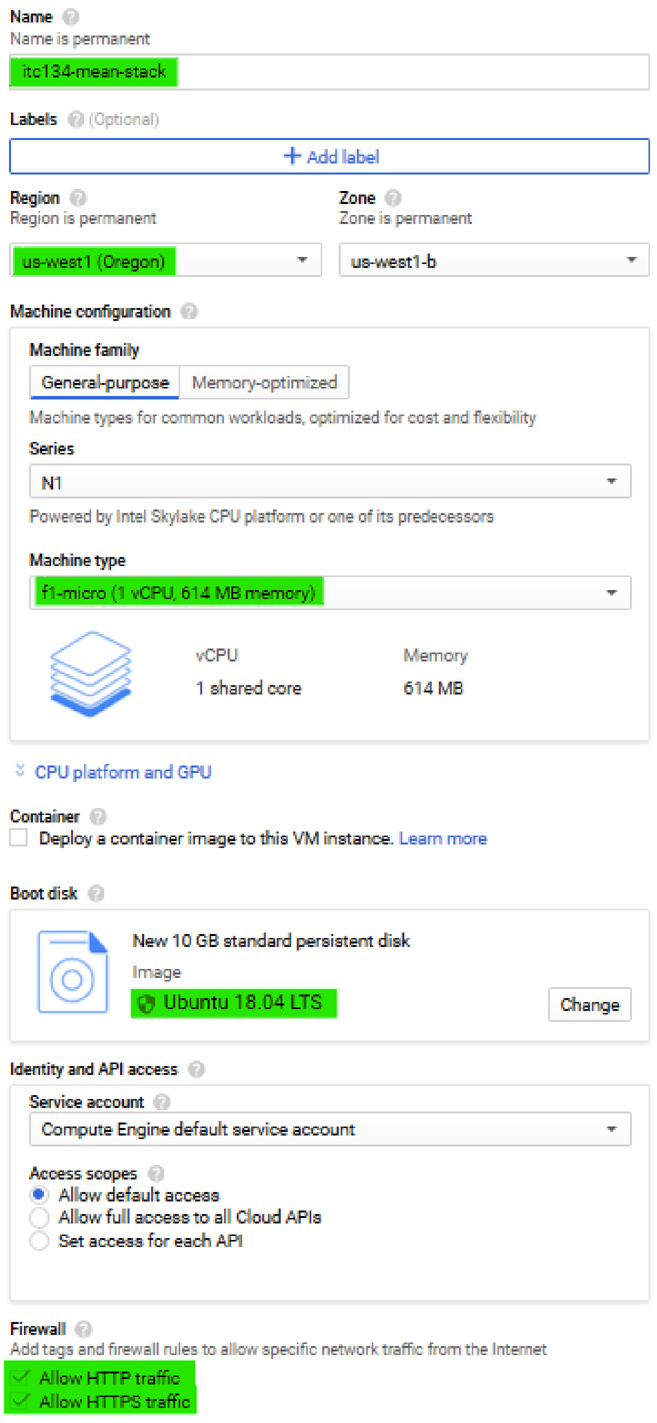
MEAN Stack Tutorial

G-Cloud - Ubuntu

- Enter the link to go to google cloud,(**Link**: https://cloud.google.com)
- Create an account
- On the bottom right look for the linked called Console
- On the top nav bar, you would create a “**Project**”
- In the nav bar on the left find “**Compute engine**”
- Once you are there select “**VM Instances**”
- Then click on the “**Create Instance**” icon or the “**Create**” button
- Name the Instance
- Select your “**Region**”
- Change the “**Machine type**” to the smallest one “**f1-micro**”
- For “**Bootdisk**” click the “**Change**” button > select the Operating System: **“Ubuntu**” and the Version > “**Ubuntu 18.04 LTS**” click “**Select”**
- For “**Firewall**” select “**Allow HTTP Traffic**” and “**Allow HTTPS Traffic**”
- Then hit “**Create**” and wait
- Once it’s done loading, check to see if your instance is running by looking at the green dot with a white checkmark.
- Click “**SSH**” to the right of the instance that you want to use, keep in mind it will take some time to load.
- Continue to Node.js…...

Screenshot for creating an instance on G-Cloud



Name
Name is permanent
itc134-mean-stack

Labels (Optional)
+ Add label

Region
Region is permanent
us-west1 (Oregon)

Zone
Zone is permanent
us-west1-b

Machine configuration

Machine family
General-purpose | Memory-optimized
Machine types for common workloads, optimized for cost and flexibility

Series
N1
Powered by Intel Skylake CPU platform or one of its predecessors

Machine type
f1-micro (1 vCPU, 614 MB memory)

vCPU	Memory
1 shared core	614 MB

CPU platform and GPU

Container
☐ Deploy a container image to this VM instance. Learn more

Boot disk
New 10 GB standard persistent disk
Image
Ubuntu 18.04 LTS
Change

Identity and API access

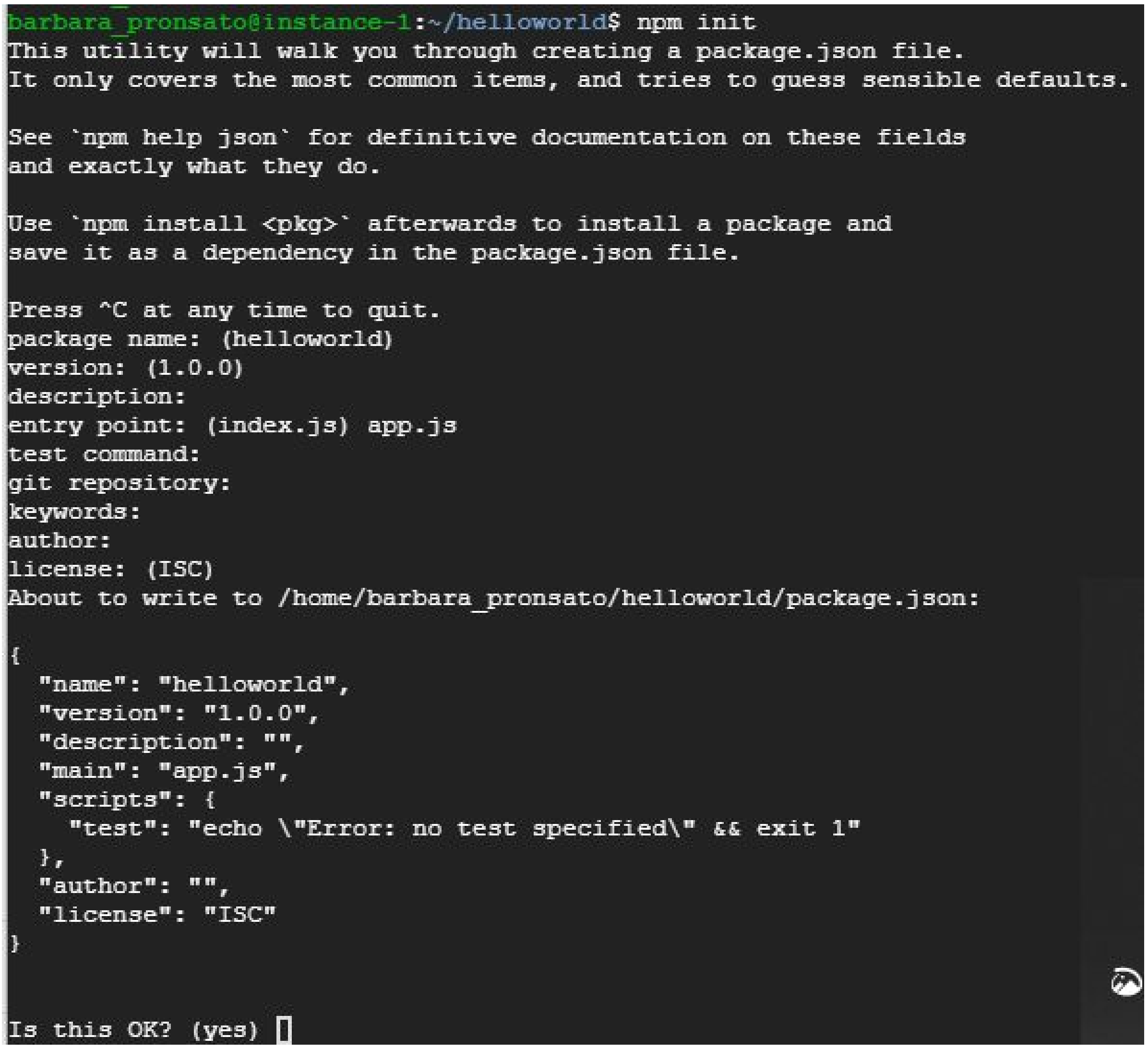
Service account
Compute Engine default service account

Access scopes
● Allow default access
○ Allow full access to all Cloud APIs
○ Set access for each API

Firewall
Add tags and firewall rules to allow specific network traffic from the Internet
✓ Allow HTTP traffic
✓ Allow HTTPS traffic

SoftWare

- Once the Command Prompt opens you'll need to install **Node.js**, in the command line Paste these commands:
    - `sudo apt-get update`
    - `sudo apt-get install -y curl apt-transport-https ca-certificates && curl --fail -ssL -o setup-nodejs https://deb.nodesource.com/setup_10.x && sudo bash setup-nodejs && sudo apt-get install -y nodejs build-essential`
- To install **Angular.js**
    - Paste this into the command line.
        - `sudo -i npm install -g @angular/cli`
    - Type "y" or "n" and hit enter when asked to share data with Google (makes no difference for this tutorial)
- To install **Express.js**
    - Create the Directory
        - `mkdir helloworld`
    - Then change the directory to helloworld.
        - `cd helloworld`
    - After that, initialize your node files and your package.json.
        - `npm init`
        - On first prompt (package name): press Enter, on second prompt (version): press Enter, on third prompt (description): press, Enter.
        - When asked for the "entry point" enter: `app.js`
        - When asked for test command: press enter.
        - When asked for git repository: press enter.
        - Keywords: press enter
        - Author: press enter
        - License (ISC): press enter
        - Is this OK? (yes): press enter



```
barbara_pronsato@instance-1:~/helloworld$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (helloworld)
version: (1.0.0)
description:
entry point: (index.js) app.js
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/barbara_pronsato/helloworld/package.json:

{
  "name": "helloworld",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

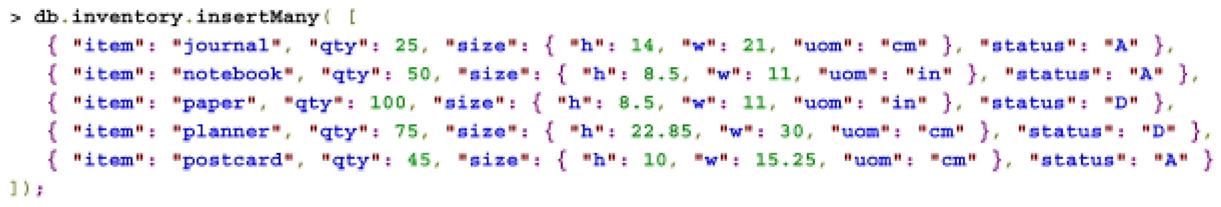
- Run express.js in your project and directory
    - `npm install express --save`
- Finally, open the nano text editor:
    - `nano app.js`
- Once you're in nano copy and paste this below

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
 res.send('Hello World!');
});
app.listen(3000, function () {
 console.log('Example app listening on port 3000!');
});
```

    - To exit the nano Ctrl+x, then press y and enter
- **MongoDB**

    - Import the public key used by the package management system
        - `wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -`
    - Create a list file for MongoDB
        - `echo "deb [ arch=amd64,arm64 ] https://repo.mongodb.org/apt/ubuntu bionic/mongodb-org/4.2 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list`

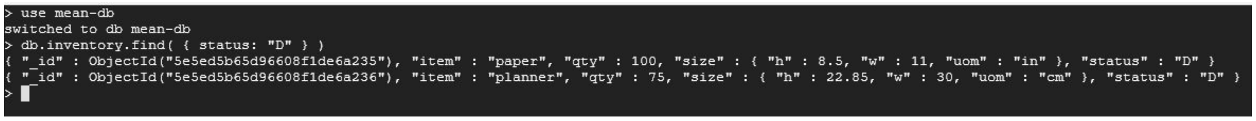
    - Reload local package database
        - `sudo apt-get update`
    - Install MongoDB packages
        - `sudo apt-get install -y mongodb-org`
    - Run MongoDB
        - `sudo systemctl start mongod`
    - To begin using mongo shell
        - `mongo`
    - Type **`use mean-stack-db`** to create and switch to a new collection named "mean-stack-db".
    - Now you enter data with **`db.inventory.insertMany();`**
    - Inside the insertMany() function you must enter data in the **JSON** format, for instance:



```
> db.inventory.insertMany( [
   { "item": "journal", "qty": 25, "size": { "h": 14, "w": 21, "uom": "cm" }, "status": "A" },
   { "item": "notebook", "qty": 50, "size": { "h": 8.5, "w": 11, "uom": "in" }, "status": "A" },
   { "item": "paper", "qty": 100, "size": { "h": 8.5, "w": 11, "uom": "in" }, "status": "D" },
   { "item": "planner", "qty": 75, "size": { "h": 22.85, "w": 30, "uom": "cm" }, "status": "D" },
   { "item": "postcard", "qty": 45, "size": { "h": 10, "w": 15.25, "uom": "cm" }, "status": "A" }
]);
```

- Finally you can search stuff inside your database by creating queries. For instance:

```
> db.inventory.find( { status: "D" } )
```



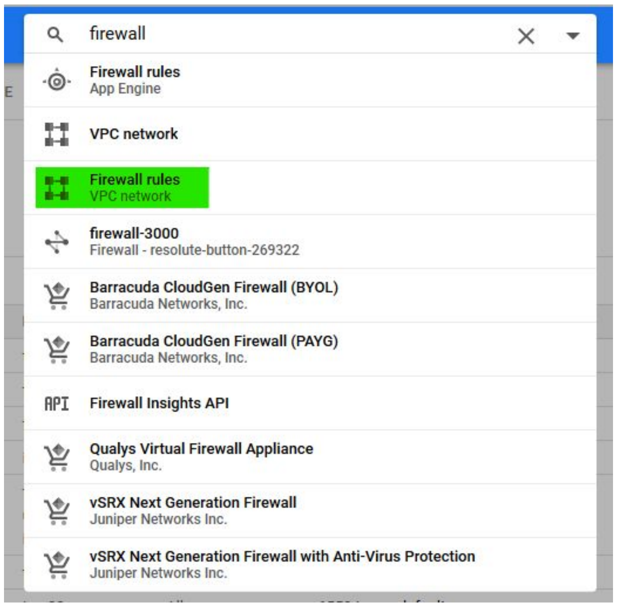
```
> use mean-db
switched to db mean-db
> db.inventory.find( { status: "D" } )
{ "_id" : ObjectId("5e5ed5b65d96608f1de6a235"), "item" : "paper", "qty" : 100, "size" : { "h" : 8.5, "w" : 11, "uom" : "in" }, "status" : "D" }
{ "_id" : ObjectId("5e5ed5b65d96608f1de6a236"), "item" : "planner", "qty" : 75, "size" : { "h" : 22.85, "w" : 30, "uom" : "cm" }, "status" : "D" }
> 
```

Firewall Rule

From command line:
"sudo iptables -I INPUT -p tcp --dport 3000 -j ACCEPT"

OR:

- Back in Google Cloud Platform, in the search bar search for: Firewall
- From the results select: Firewall rules



firewall

Firewall rules
App Engine

VPC network

Firewall rules
VPC network

firewall-3000
Firewall - resolute-button-269322

Barracuda CloudGen Firewall (BYOL)
Barracuda Networks, Inc.

Barracuda CloudGen Firewall (PAYG)
Barracuda Networks, Inc.

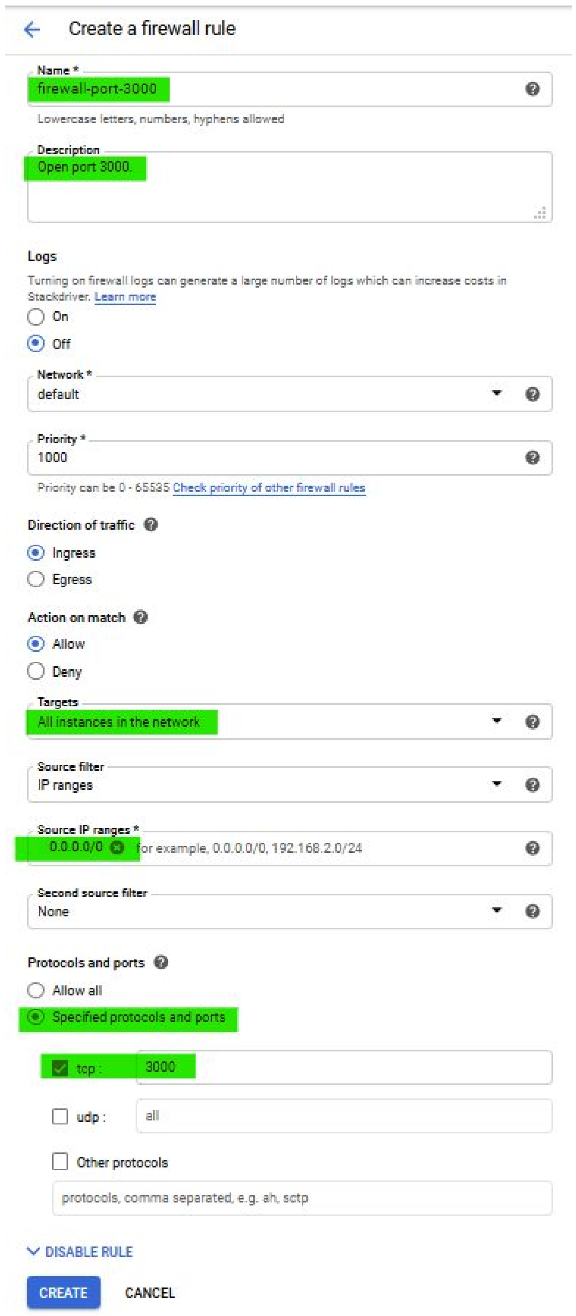
Firewall Insights API

Qualys Virtual Firewall Appliance
Qualys, Inc.

vSRX Next Generation Firewall
Juniper Networks Inc.

vSRX Next Generation Firewall with Anti-Virus Protection
Juniper Networks Inc.

- Click on "**CREATE FIREWALL RULE**"
- Name: **firewall-port-3000**
- Description: **Open port 3000**
- Targets: **All instance in the network**
- Source IP ranges: **0.0.0.0/0**
- Protocols and ports: Click on checkbox: **"tcp"** and in entry box type: **"3000"**
- Click **CREATE** button


Create a firewall rule

Name *
firewall-port-3000
Lowercase letters, numbers, hyphens allowed

Description
Open port 3000.

Logs
Turning on firewall logs can generate a large number of logs which can increase costs in Stackdriver. Learn more
On [✓ Off]

Network *
default

Priority *
1000
Priority can be 0 - 65535 Check priority of other firewall rules

Direction of traffic
[✓ Ingress] Egress

Action on match
[✓ Allow] Deny

Targets
All instances in the network

Source filter
IP ranges

Source IP ranges *
0.0.0.0/0 for example, 0.0.0.0/0, 192.168.2.0/24

Second source filter
None

Protocols and ports
Allow all [✓ Specified protocols and ports]

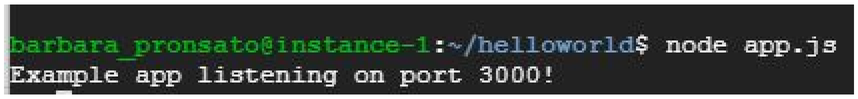
- [x] tcp : 3000
- [ ] udp : all
- [ ] Other protocols

protocols, comma separated, e.g. ah, sctp

Running the Server

- Back in the GNU Bash Shell run the server with the following command: (if still in the mongo db type `exit`
    - `node app.js`

If you see this you're good:



```
barbara_pronsato@instance-1:~/helloworld$ node app.js
Example app listening on port 3000!
```

- Back in the Google Cloud Platform go to:
    - Compute Engine
    - Click on VM instances:
    - Type the External IP address into the URL bar, **removing** the "**s**" from "**https**" and adding "**:3000**" to the end
- Example: **http://34.82.213.84:3000**