Setting up a Node express application on Digital Ocean

For this exercise you can use an existing Droplet, or create a new one, using instructions from previous semesters.
The following will assume you have a droplet with a non-root user as the starting point.

The following will guide you through the necessary steps to host an express-generator generated application on port 80 on Digital Ocean.

1) Getting your Express app ready for Digital Ocean Hosting

Either create a new Express application with the generator, or just use an existing application.

Open the file `bin\www` and find the line that sets the port number. Add the highlighted text.

```
var port = normalizePort(process.env.PORT || '3000');
var ip = process.env .IP || "localhost";
```

Now find the line that actually starts the server, and add the highlighted text:
```
server.listen(port ,ip);
```

In order to run the server on DO we need to to bind the server to the public interface. We will provide the variables for this as Environment Variables.
Add a new file process.json to the root of the application and paste this JSON into the file:

```
{
 "apps" : [{
   "name"        : "app",
   "script"      : "./bin/www",
   "watch"       : true,
   "env": {
     "NODE_ENV": "development",
     "IP" : "0.0.0.0",
     "PORT": 80
   }
 }]
}
```

This file will be used by a process manager pm2 which we will use to start the app. Read more about this here:https://expressjs.com/en/advanced/pm.html

2) Push you Express Application to github (remember to ignore node_modules)

3) Install node.js on your Droplet

Use this tutorial
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-16-04
and the section"*How To Install Using a PPA*" to install node (this is all you need from this tutorial).

4) Allow your node-apps to use port 80

Only root users are allowed (by default) to use ports under 1024. A quick and easy fix for this (and probably not a solution for a real production system) is the following:

Type these two commands:

```
sudo setcap cap_net_bind_service=+ep /usr/bin/node
sudo setcap 'cap_net_bind_service=+ep' /usr/bin/nodejs
```

5) Install the process manager PM2

Type: `sudo npm install pm2 -g`

6) Deploy your Express App to the server

Since your application has been pushed to github, this is a simple as; just cloning the project into a folder on your droplet

Type:

```
cd ~
mkdir express
cd express
git clone xxx    (replace with the url for your git-repo)
```

Navigate into the git project and type:

```
npm install
```

Now your Express application is ready to run. Type: `npm start`

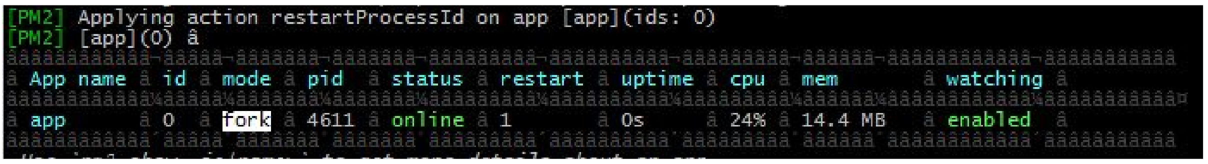
This will block the terminal until you shutdown the server (CTRL+C). So open a new terminal and ssh into it, just as you did above, and in this terminal type:

```
curl http://localhost:3000
```

If this works, go back to the first terminal and shut-down the server. Now we will start the server with the PM2 process manager (and use the environment variables from the file process.json). Type:

```
pm2 start process.json
```

You should see something like this (and observe that the terminal is no longer blocked)



```
[PM2] Applying action restartProcessId on app [app](ids: 0)
[PM2] [app](0) â
ââââââââââââ¬âââââ¬âââââââ¬âââââââ¬ââââââââ¬ââââââââââ¬âââââââââ¬ââââââ¬âââââââââââ¬ââââââââââ
â App name â id â mode â pid  â status  â restart â uptime â cpu â mem        â watching â
ââââââââââââ¼âââââ¼âââââââ¼âââââââ¼ââââââââ¼ââââââââââ¼âââââââââ¼ââââââ¼âââââââââââ¼ââââââââââ¤
â app      â 0  â fork â 4611 â online  â 1       â 0s     â 24% â 14.4 MB    â enabled  â
ââââââââââââ´âââââ´âââââââ´âââââââ´ââââââââ´ââââââââââ´âââââââââ´ââââââ´âââââââââââ´ââââââââââ
```

Open a browser and verify that you can access your Express Application running via your droplet's IP and the default web-port 80.