SHORT GUIDE FOR DEPLOYING

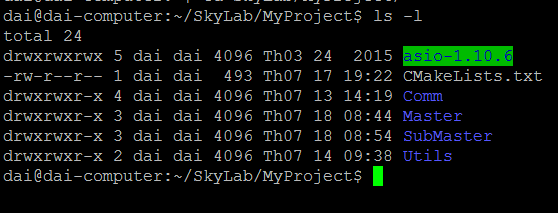
I have developed, debugged and preliminary tested the firmware on the PC running Ubuntu. The following is the steps for transferring the code to BBB and then compile, run the application on real Beagle-Bone Black.

1. First, I need to transfer the source code from my Ubuntu PC to BBB board. I assume that we have installed Debian Distribution on BBB. My BBB have IP address 10.10.1.144 and I will use default account in this guide.

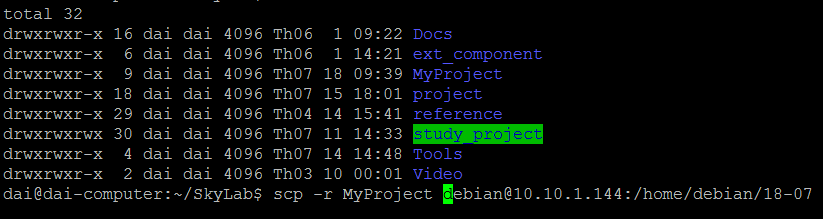
Login name: debian

Password: temppwd

This is the directory structure of project on my Ubuntu PC

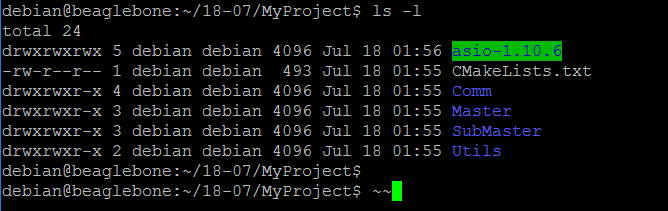


I will use the scp command to copy code to BBB



1. After transferring the source code to the BBB. It is ready to deploy the application on the real hardware. Again, I will use the putty from my Windows PC to do tasks

Showing the source code on the BBB



1. As of now, I have used cmake to re-compile the source code on the BBB. So there are some dependencies we need to install on the BBB: cmake, boost library (for some utilities)

The followings were the commands to install

debian@beaglebone:$ sudo apt-get update

debian@beaglebone:$ sudo apt-get install cmake

debian@beaglebone:$ sudo apt-get install cmake

debian@beaglebone:$ sudo apt-get install libboost-dev

1. When all dependencies were installed, we are ready to compile our project

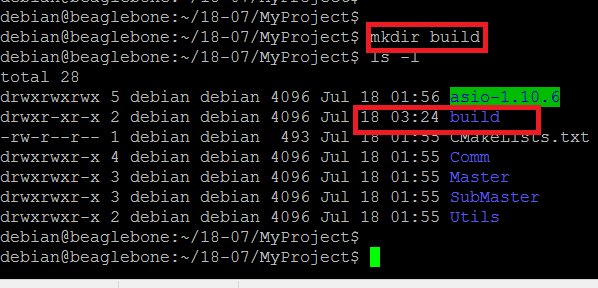
NOTE:

Before compiling the source code, you must update IP address and port settings on source code matching with current settings of your testing environment.

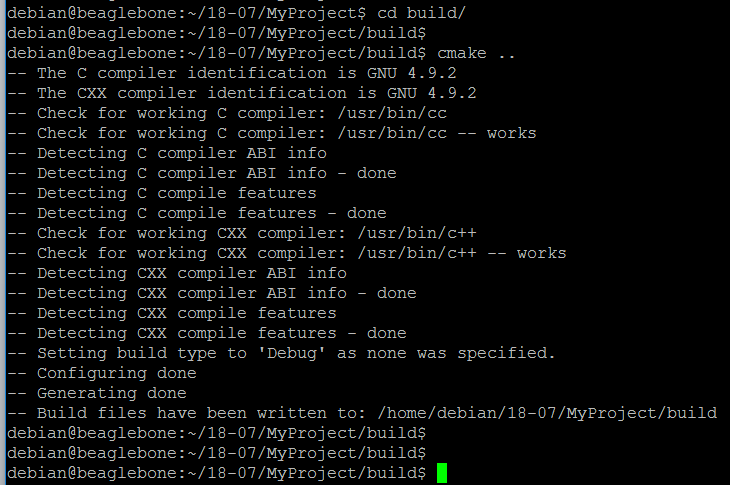
+ Open file main.cpp in “master” directory and update IP address and port information to permit it connects to submaster application

+ Open file main.cpp in “submaster” directory and update IP address setting to IP address of submaster itself.

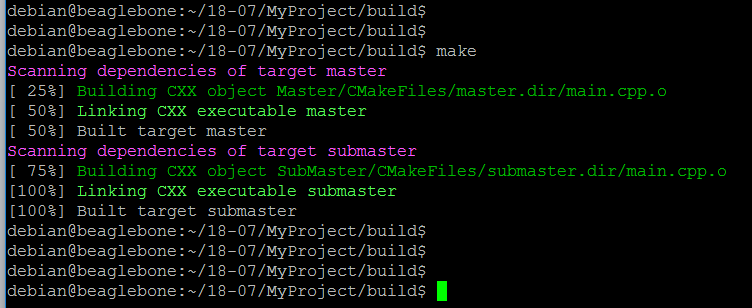
* Make sure we were in the root directory of project
* Make new directory for building progress, for example “build”



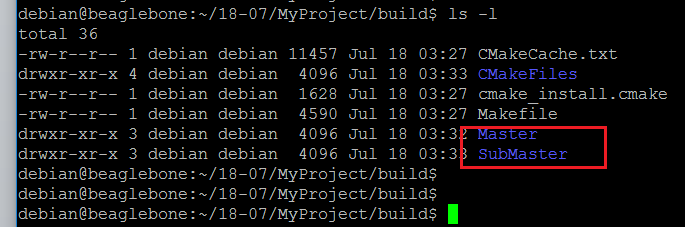
* Move to just-created “build” directory and run “cmake ..” command. If everything was right, we will have similar screen as below

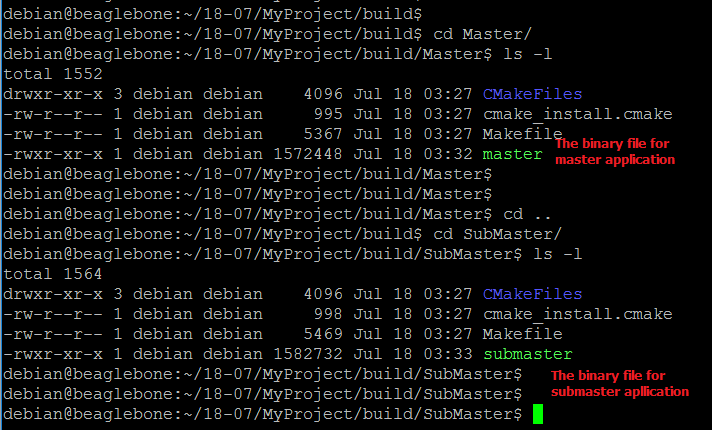


* Next, run “make” command and wait for a while (2 or 3 minutes) before compiling progress to finish
* Again, If everything was right, we will have similar screen as below



The compile will create 2 sub-directories under “build” directory. The one was used to store binary file for master application, the other was used to store binary file for submaster application.





Now, we have binary files for both master and submaster applications. You can test them with the simulator and real external serial devices.

JSON FORMAT OF SETTINGS

{

"version": "1.0.0",

"self\_id": 0,

"units": [

{

"unit\_type": "master",

"unit\_id": 0,

"unit\_ipaddr": "192.168.1.1"

},

{

"unit\_type": "slave",

"unit\_id": 1,

"unit\_ipaddr": "192.168.1.2"

},

.

.

.

{

"unit\_type": "submaster",

"unit\_id": 9,

"unit\_ipaddr": "192.168.1.10"

}

]

}

The application on each BBB should have its own settings.json file was on the same directory with binary file of the application

+ The "self\_id" element in JSON string used to identify the unit

+ The settings.json file for Master should have "self\_id" assigned to 0

+ The settings.json file for Submaster should have "self\_id" assigned to 9

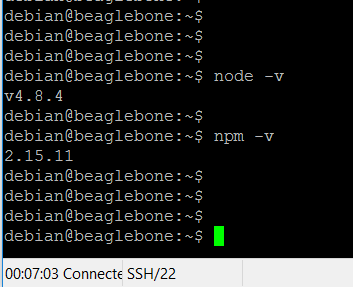
+ The settings.json file for Slave should have "self\_id" assigned to 1 – 8

+ The "units" array stores all information of all units in details and this array should be the same on all BBBs

WEB SETUP

1. Install node.js and Express.js

Latest debian distribution on Beagle Bone Black have pre-installed node.js and Node Package Manager (npm) as picture below:

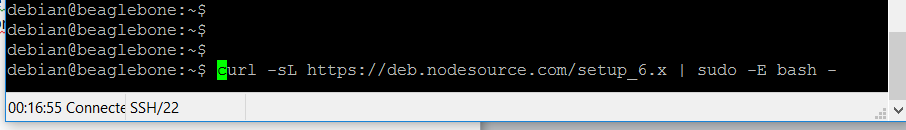


But, this is not the latest version of node.js as well as npm component

So, we need to run some commands to update node.js and npm to latest versions.

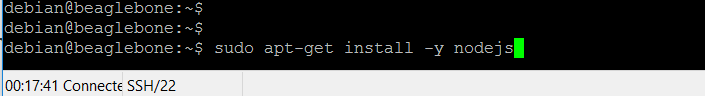
* First, run this command

curl -sL https://deb.nodesource.com/setup\_6.x | sudo -E bash –

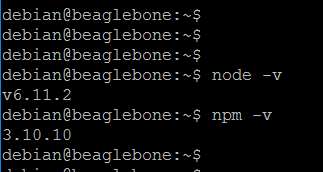


* Next, run

sudo apt-get install -y nodejs



After installation process was done, both node.js and npm were updated to latest version



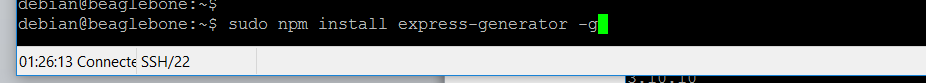
When you go at here, you are ready to deploy the web application. The production web application should be self-contained and have integrated all required components and dependencies (included Express.js)

However, it is better for you know the process creating the Express.js web application. The following sections will detail the steps that I have done.

1. Install Express Application Generator

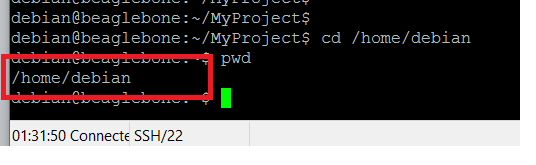
This tool was used to create skeleton Express.js web application. So run this command to install Express Application Generator for system global

* sudo npm install express-generator –g



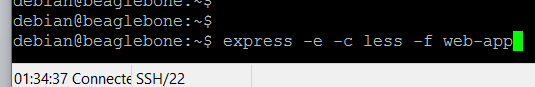
* Next, go to the directory that you want the new web application was located. Now, I will use home directory of current account - /home/debian.

Run command cd /home/debian

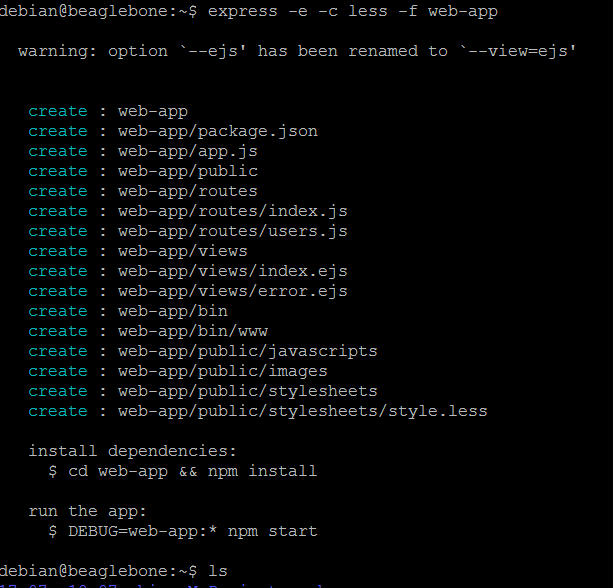


* Next, we will use application generator to create web application named web-app at /home/debian.

Run command express -e -c less -f web-app



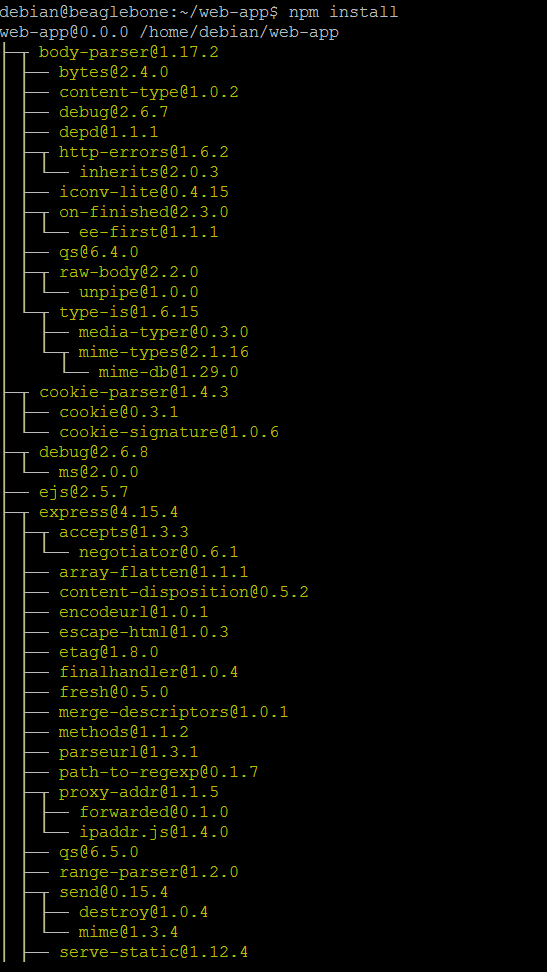
We will see the screen when the command was completed



As the hints on the screen. Next step should be go to new created directory (web-app) and run command ‘nmp install)

* Go to web-app directory and run npm install

When the command was completed you will see the screen like this



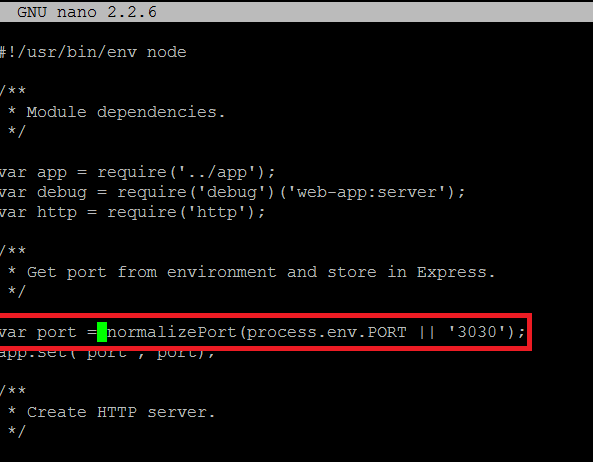




* At that time, you can start web application. But with my experience, it looks like the 3000 port have used by some other application, I don’t know which application now. So we need to change the default port from 3000 to 3030 for example.
* Go to bin directory and open file www as follow

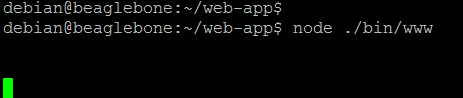


And change 3000 value to 3030 as follow

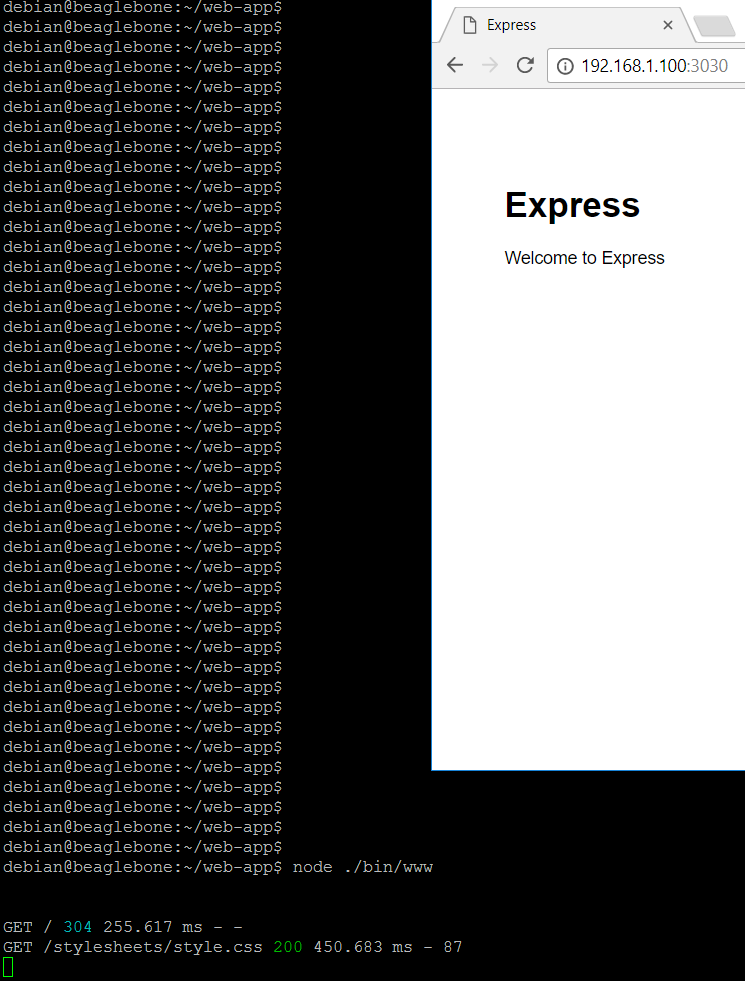


Save and exit

* Go back to parent of the bin (it is web-app directory) and start the web with command node ./bin/www



Now the web have started and you can access to the skeleton application with your browser

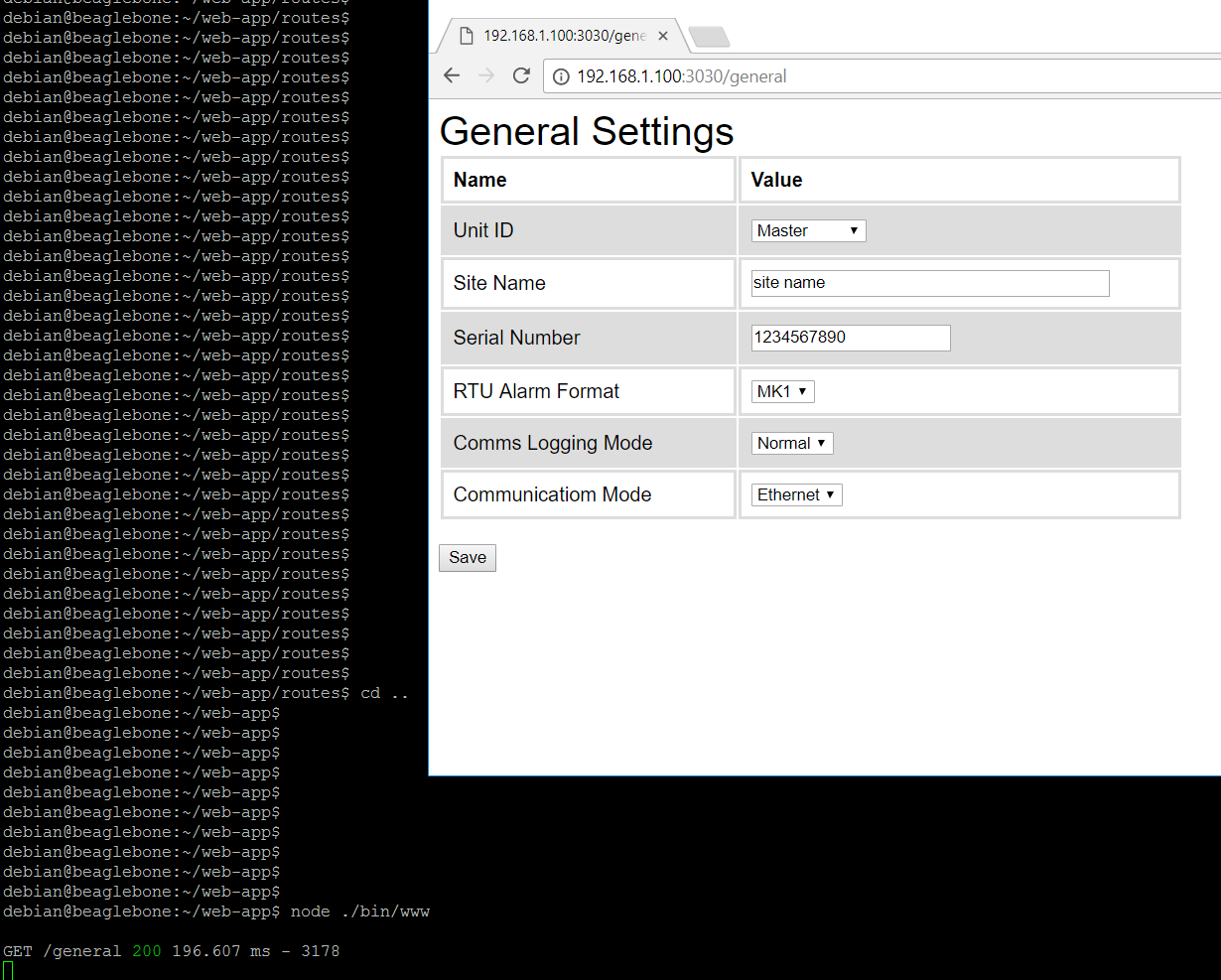


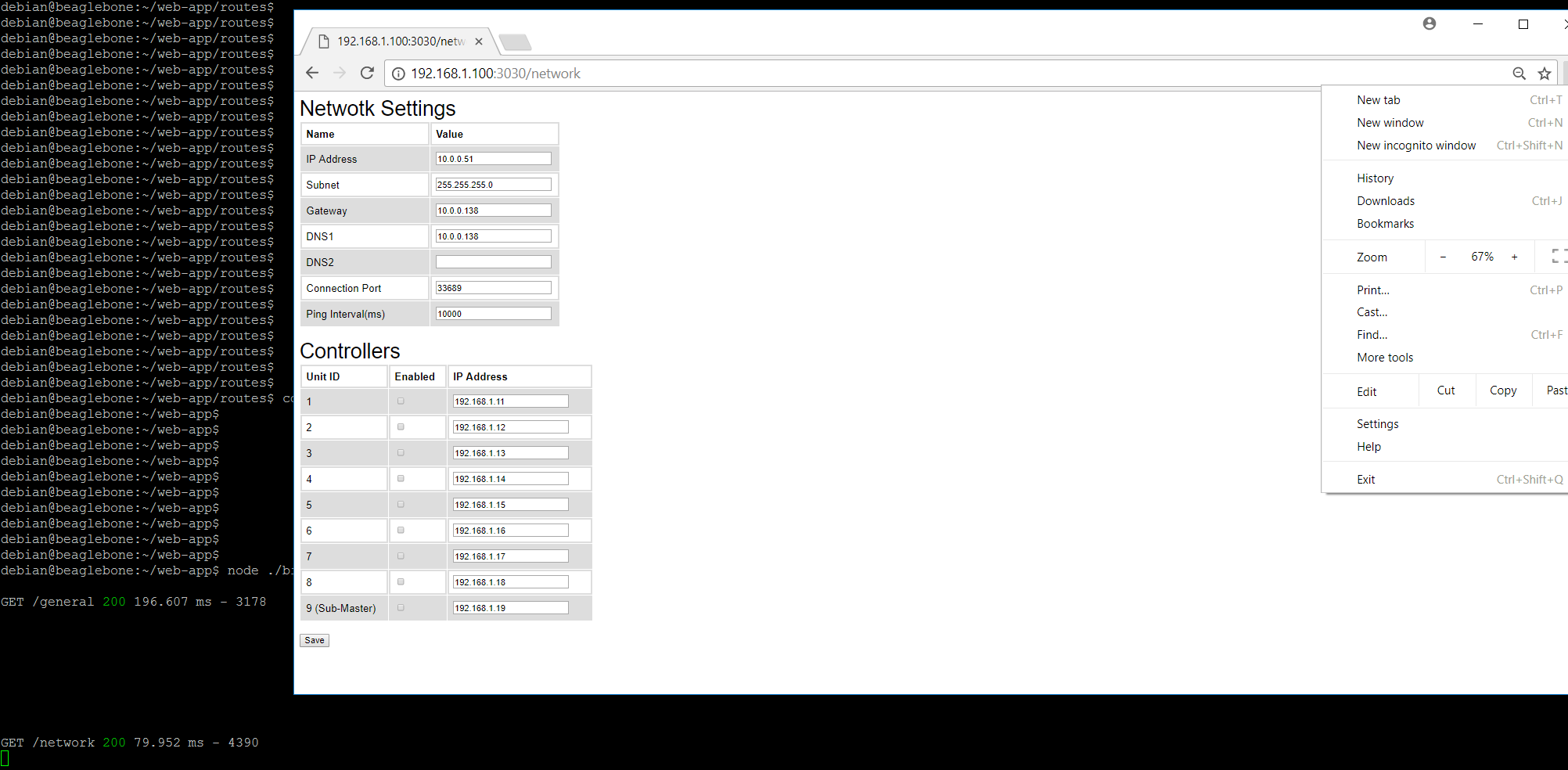
* At that time, we have setup the web application. The final steps were to pull 3 files from GIT and copy them into the web application

Copy the file index.js from web-app/routes on GIT and paste it on web-app/route on the BBB.

Copy files index1.ejs, index2.ejs from web-app/views on GIT and paste them on web-app/views on the BBB.

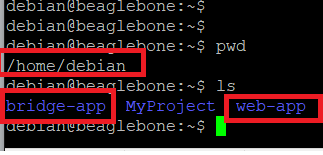
Stop the web and re-run it. And now you can access two web pages for our system.

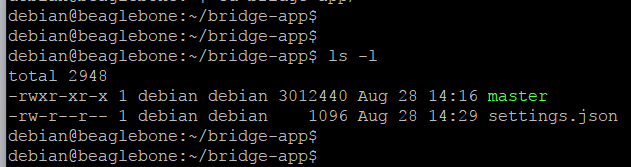


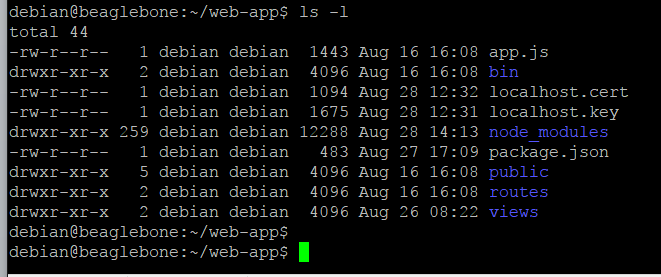


I will continue with data-binding and more features !!!

Because configurations of the application was stored in JSON/text file (settings.json), and this file was accessed from both serial-tcp convertor and web-app. So we need to assume for the location of this file. Assuming that we will store the convertor application in ‘bridge-app’ directory and web application in ‘web-app’ directory. Boths were location under default user directory – debian.

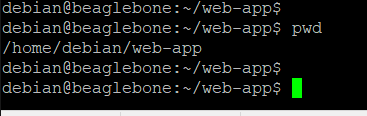






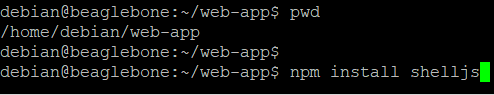
The settings.json file was located in bridge-app.

The web-app used some modules from node.js. We need install them. First, go to the root directory of web application (./web-app)



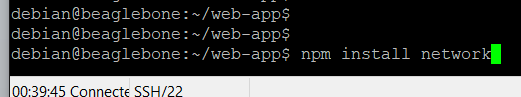
And then run npm install shelljs

To install Unix shell commands



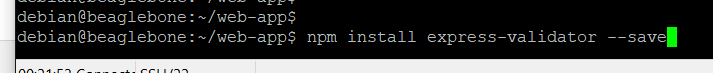
Next run npm install network

To install network utilities

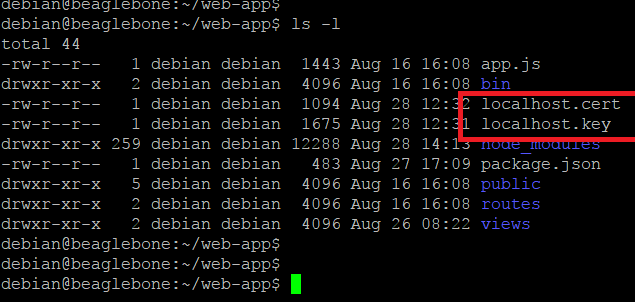


And finally run npm install express-validators –save

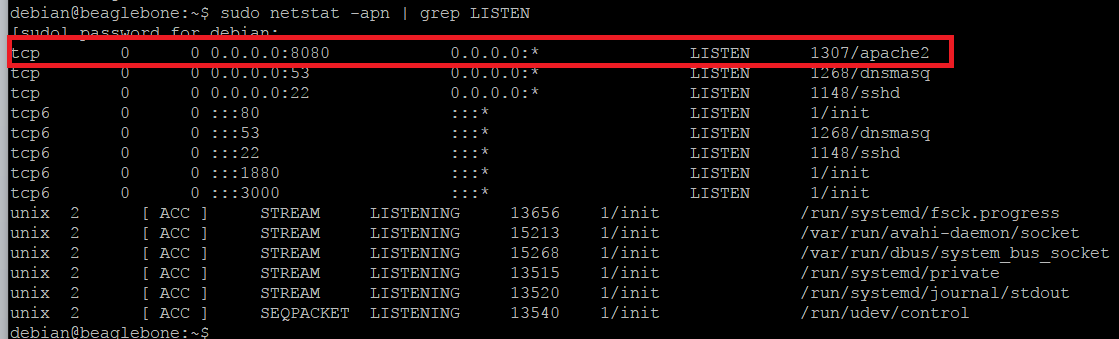
To install validation and sanitization



Now that, we can pull new files from GIT for web-app. There are two files for https were to make the security connections



By default, the Debian image for BBB have installed and started Apache2 and was used port 8080 for listing to connections.

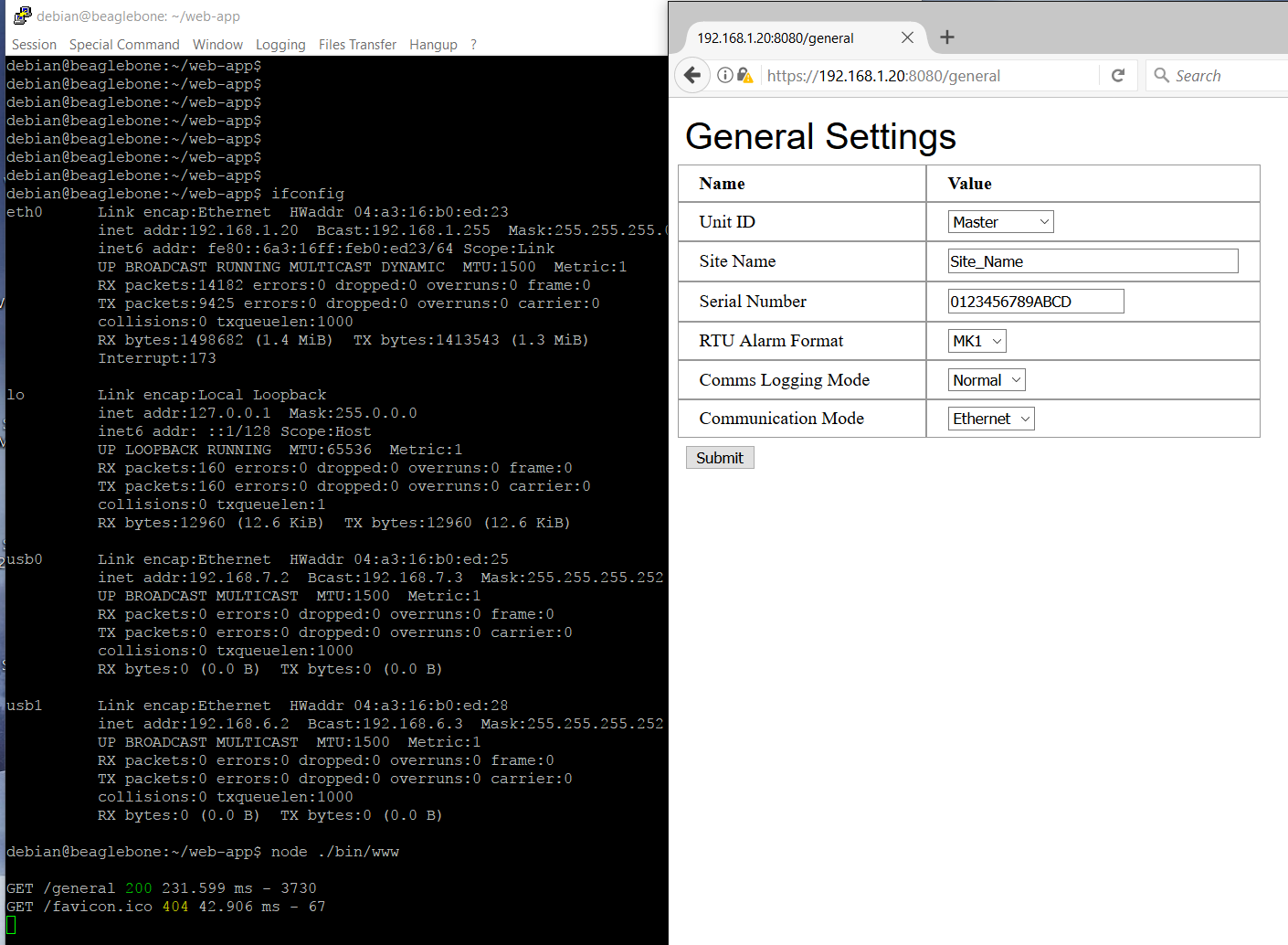


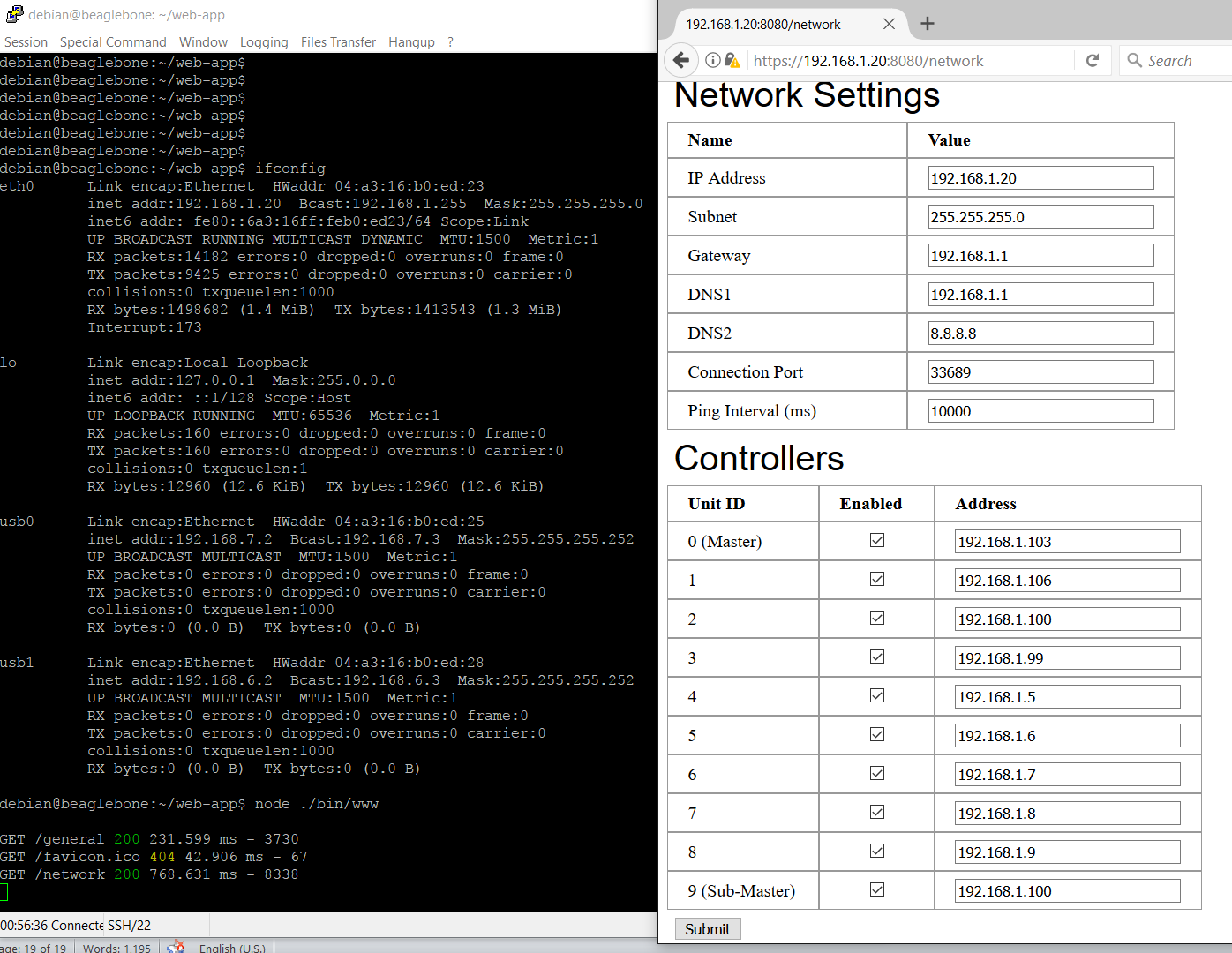
So, we need to remove apache2 from BBB by running these commands

sudo apt-get purge apache2 apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common

sudo apt-get autoremove

Now, we can run our web-app application



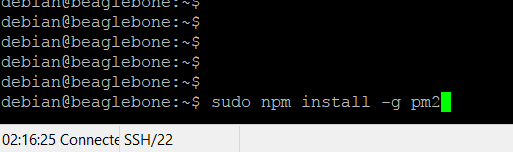


Deploying Web and SerialTcp Application

1. Web application

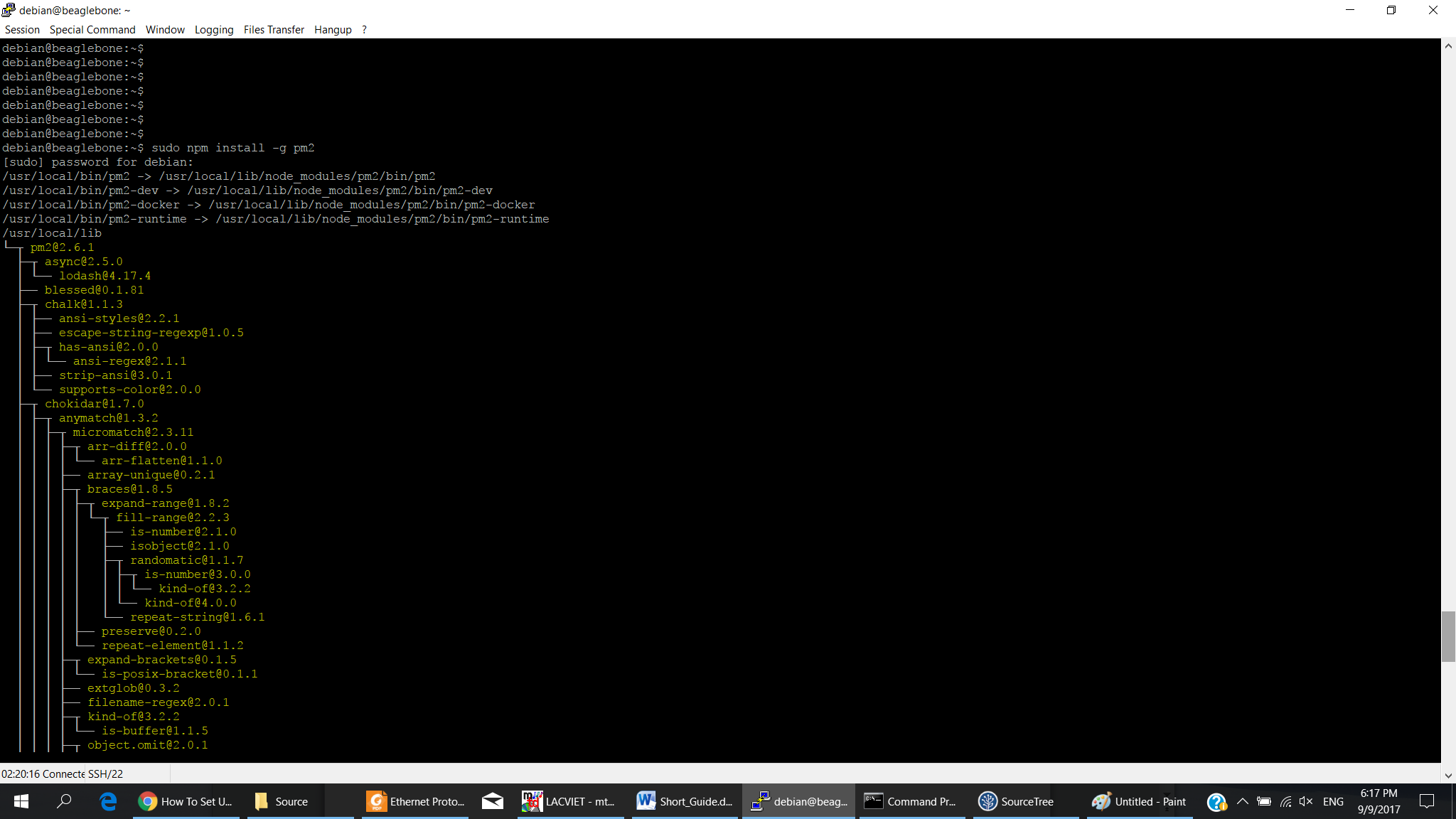
We will use PM2, which is a process manager for Node.JS applications.

Run command sudo npm install -g pm2 to install PM2

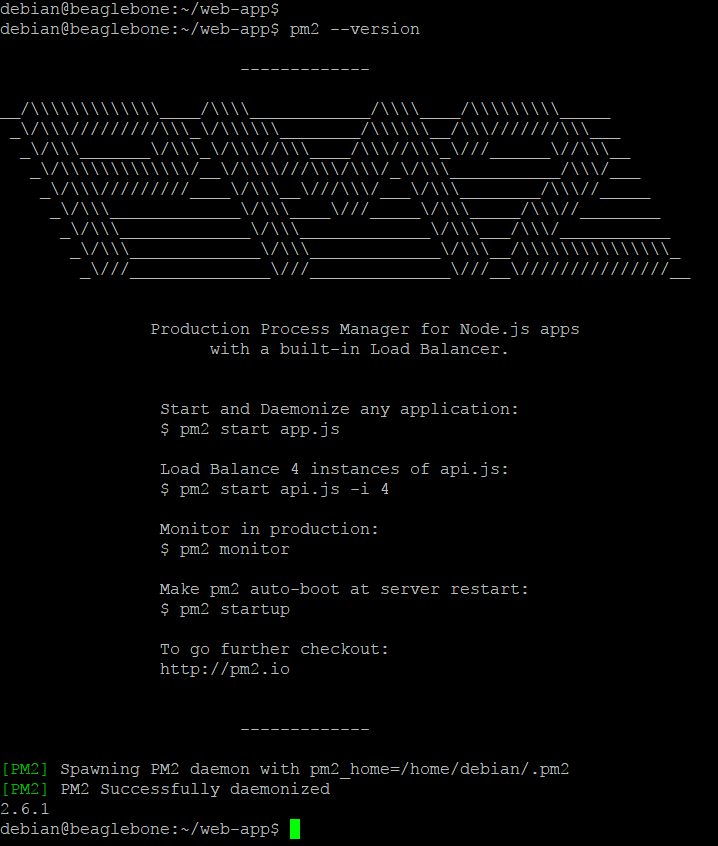


The -g option tells npm to install the module globally, so that it's available system-wide.

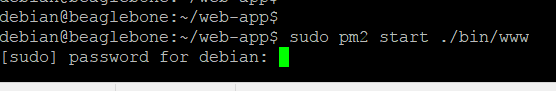
If the PM2 was installed successfully, we will see the output that is similar to below



We can show version of PM2 as follow:



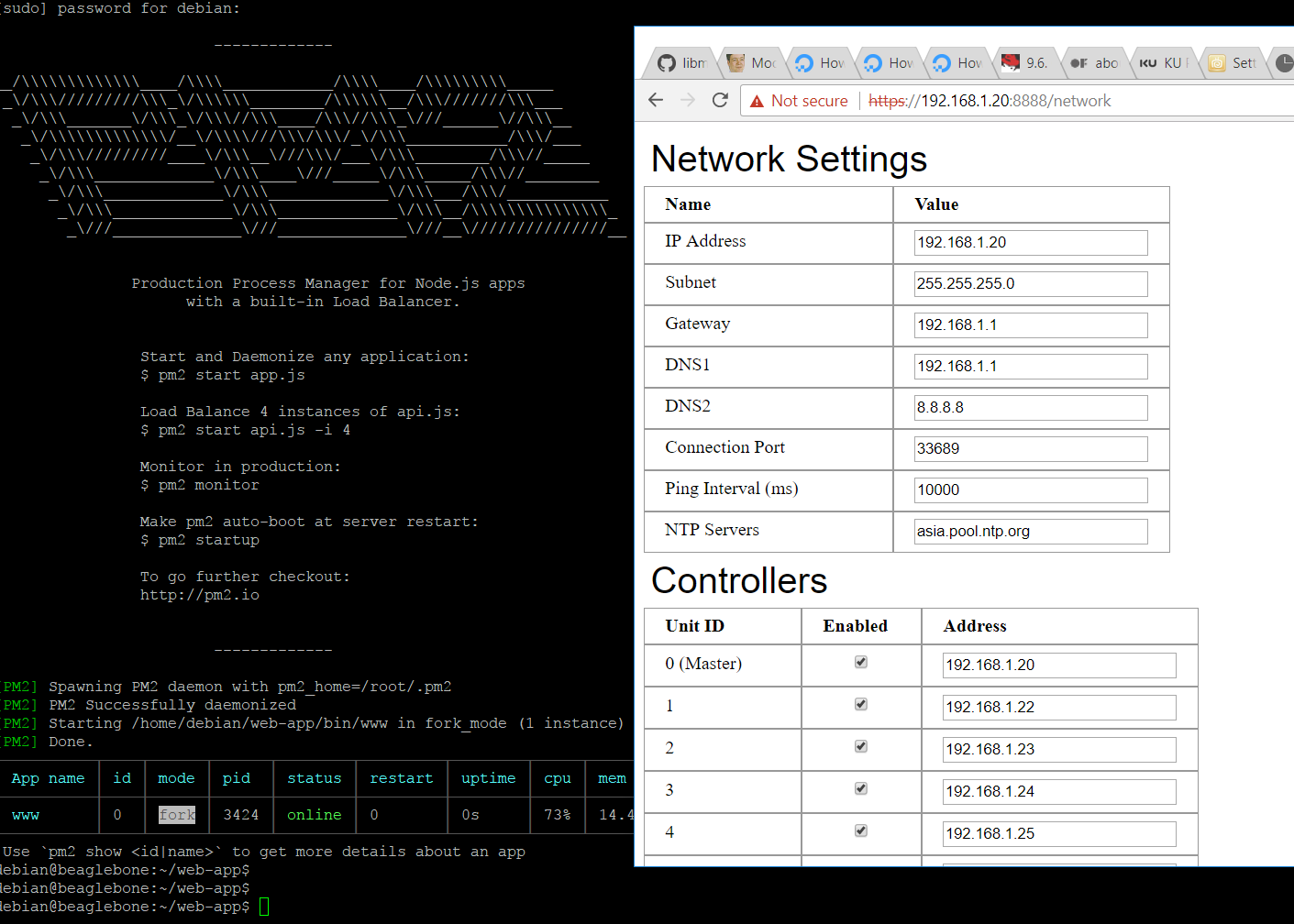
Next, make sure you are in the root directory of the web application. We will use PM2 to start our web application by using command sudo pm2 ./bin/www



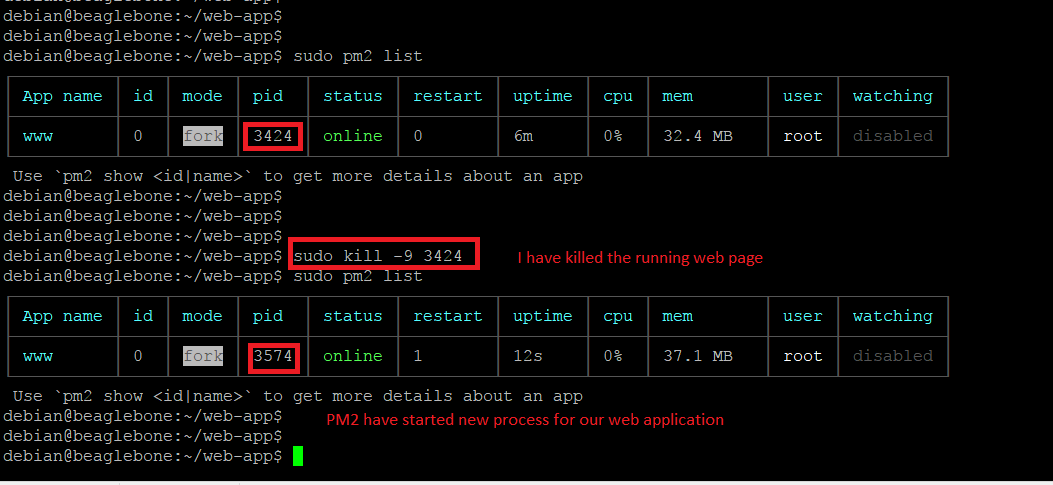
You will see the screen as below if PM2 started our web application



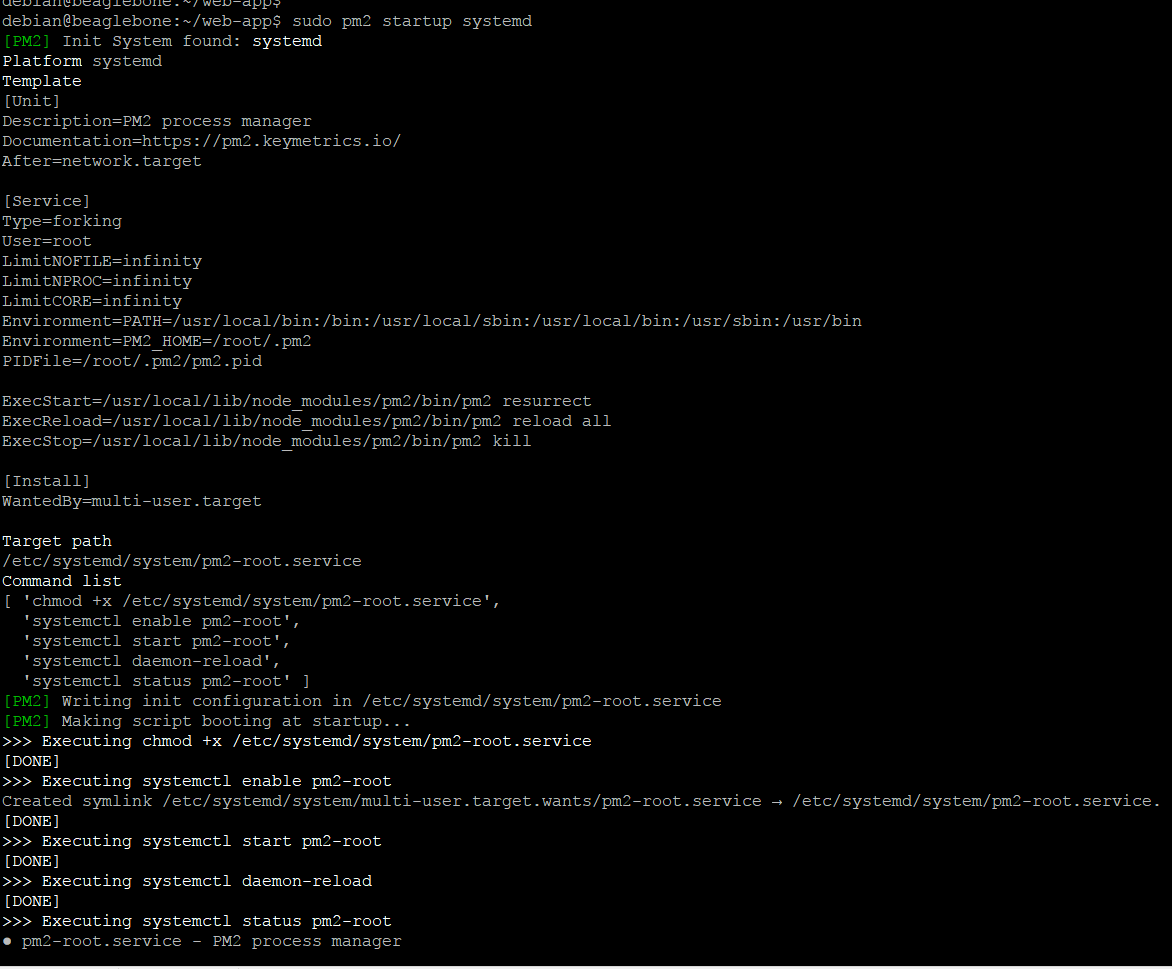
Now, you can try to access to our web to make sure it is working.



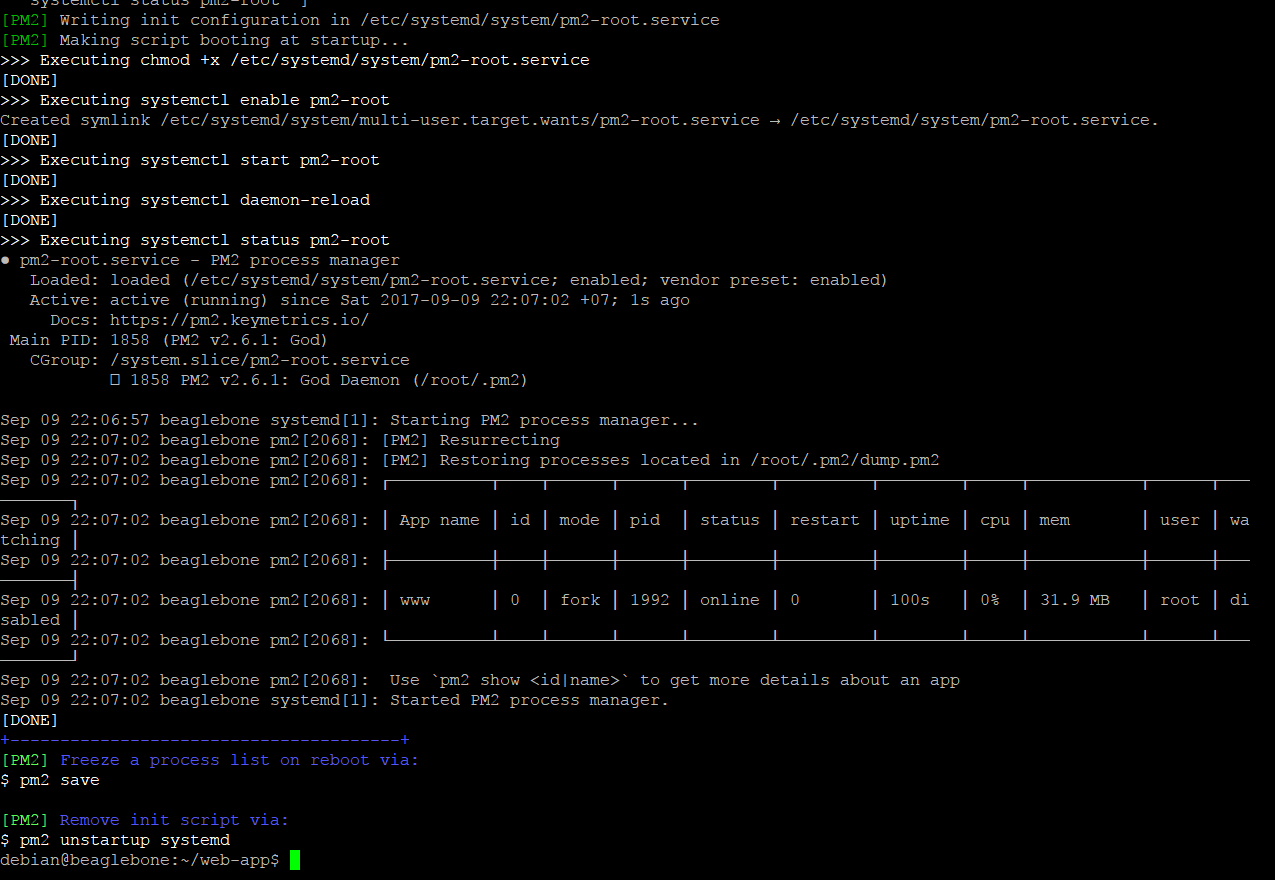
Applications that are running under PM2 will be restarted automatically if the application crashes or is killed, as you can see my screen-shot



We need to run an additional command to start our web application on startup or reboot. sudo pm2 startup systemd



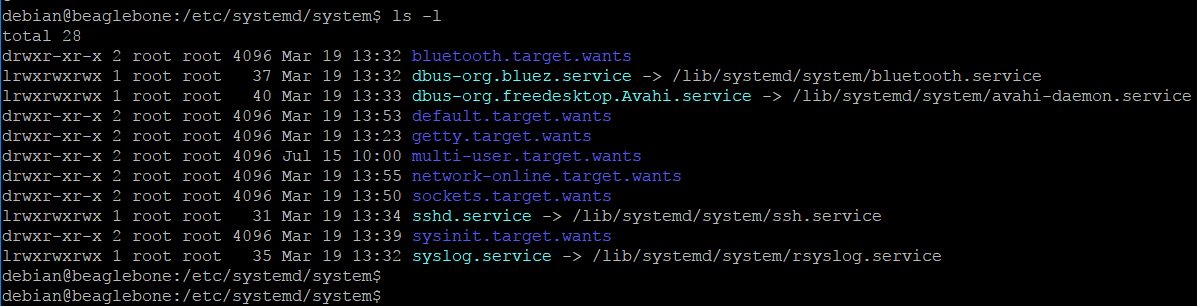
Now, our web application will been restarted when it was killed or crashed or started up on the system boot and reboot.



1. Serial to Tcp application

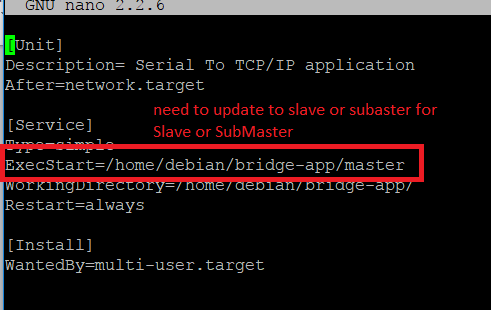
Auto-start serialtcp using systemd of Operating System

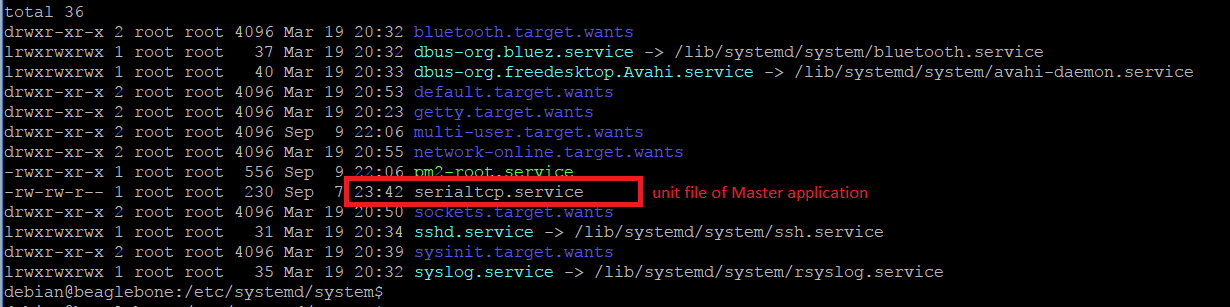
1. Move to /etc/system/system



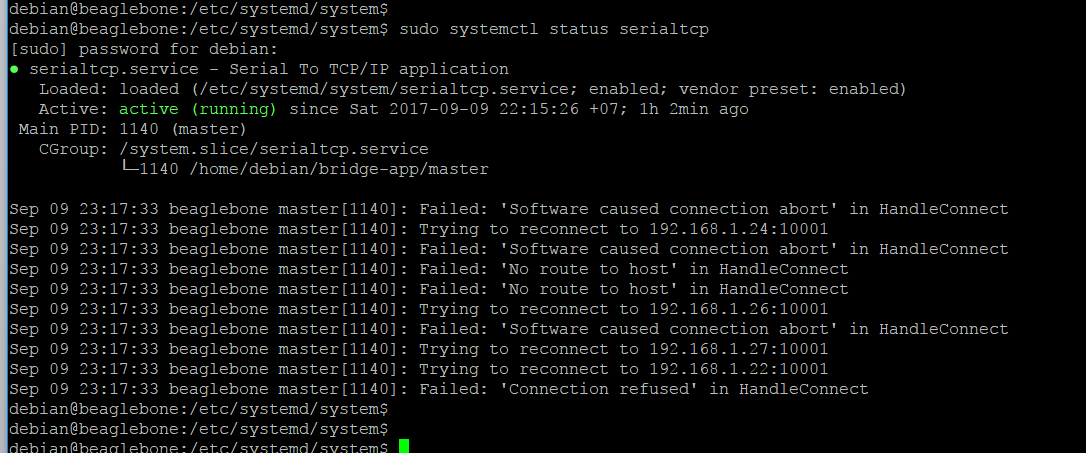
1. Create new unit file serialtcp.service with contents as follows

The contents were used for auto-starting Master application. You have to change ExecStart variable for Slave and SubMaster





1. Start serialtcp.service. sudo systemctl start serialtcp
2. Check status of serialtcp.service to make sure it was started. sudo systemctl status serialtcp



1. Finally, Enable auto-start on reboot and start-up. sudo systemctl enable serialtcp.service
2. Check to know if the service was enabled. sudo is-enabled serialtcp.service

