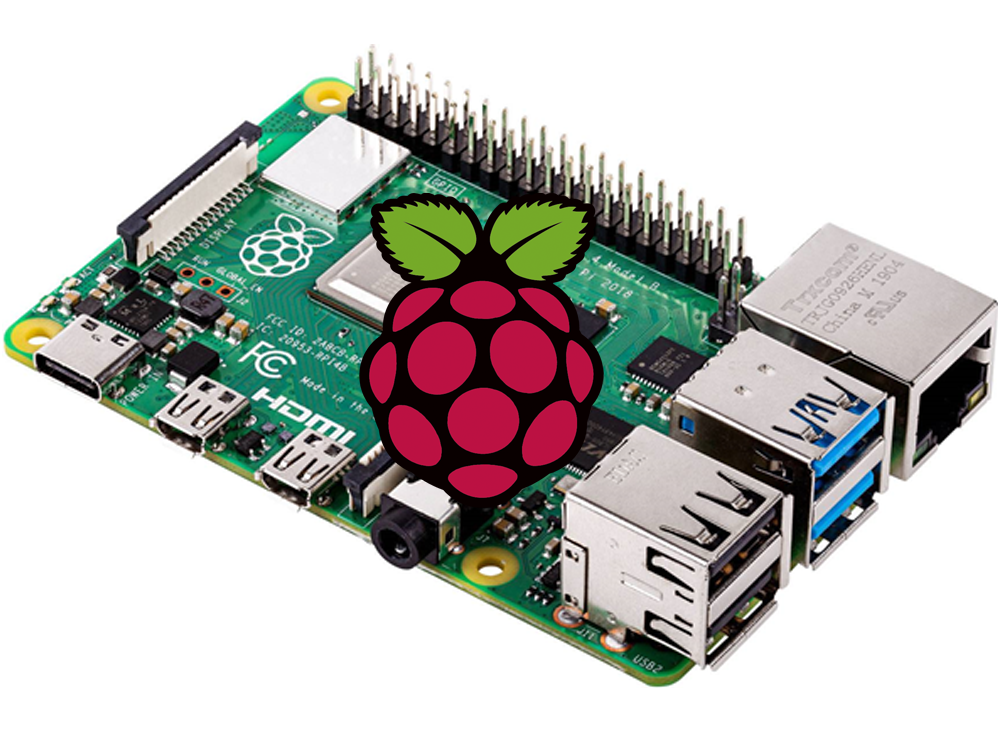
Installation Raspberry Pi 4



Senstable 2

Ziwe Zhang - 0984223

Index

[Requirements 3](#_Toc74438412)

[Installation Wi-Fi Adapter 4](#_Toc74438413)

[Getting started 4](#_Toc74438414)

[Downloading driver 4](#_Toc74438415)

[Installation driver 5](#_Toc74438416)

[Setting up Wi-Fi 5](#_Toc74438417)

[Installation Access Point 6](#_Toc74438418)

[Getting started 6](#_Toc74438419)

[Configuring a static IP 6](#_Toc74438420)

[Configuring the DHCP server (DNSMasq) 6](#_Toc74438421)

[Configuring the access point host software (HostAPD) 7](#_Toc74438422)

[Start it up 8](#_Toc74438423)

[Add routing and masquerade 9](#_Toc74438424)

[Setup Webpage 10](#_Toc74438425)

[Setup NodeJS server 12](#_Toc74438426)

# Requirements

* Raspberry Pi 4
* USB C charger
* Micro HDMI to HDMI cable
* A monitor
* Mouse
* Keyboard
* 16 or 32 GB SD card
* Wi-Fi adapter

You’ll need a mouse, keyboard and a Micro HDMI cable for the setup. It doesn’t matter what kind of mouse or keyboard, but preferably one with cable instead of a cable. Because this can cause issues depending on which ones you’re using.





You’ll need at least a 16GB SD card. I’m using a 32GB SD card because that way I know I will always have enough memory.



And lastly, you’ll need a Wi-Fi adapter, because the Wi-Fi chip on the raspberry pi is being used as an access point. The chip only works as a client (to connect to a Wi-Fi network) or as an Access Point (AP). I’m using the [TP-Link TL-WN725N](https://www.bol.com/nl/p/tp-link-tl-wn725n-wireless-n-nano-usb-adapter-150-mbps/1003004012437373/?bltgh=m69G4YIMjMnbJYtiUyJE-A.2_9.10.ProductTitle), but anything will do. But make sure it is compatible with a raspberry pi.

# Installation Wi-Fi Adapter

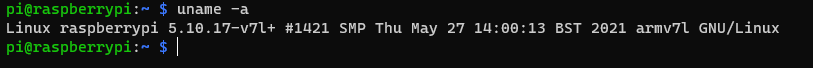
## Getting started

I’m using the [TP-Link TL-WN725N](https://www.bol.com/nl/p/tp-link-tl-wn725n-wireless-n-nano-usb-adapter-150-mbps/1003004012437373/?bltgh=m69G4YIMjMnbJYtiUyJE-A.2_9.10.ProductTitle), depending on which one you’re using it may be a different process to install the adapter.

The TP-Link TL-WN725N needs a driver to be installed on the Raspberry Pi for it to work. We need the driver to load every time when you boot your device.

Firstly, we need to find the kernel version of your system. We can find it using this command:

uname -a

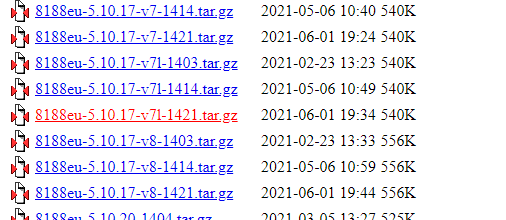
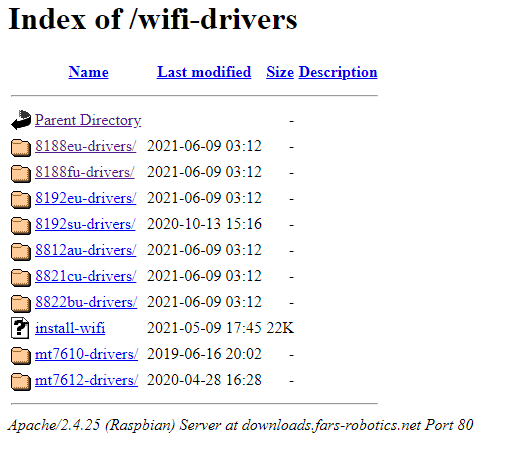


The important numbers are: **5.10.17-v7l+ #1421** (this may change depending on your kernel version)

## Downloading driver

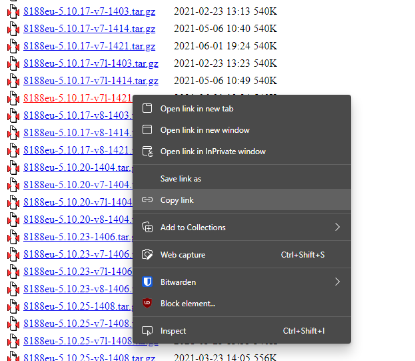
With the kernel version, we can download the correct driver for the Wi-Fi adapter. On this website “[Index of /wifi-drivers (fars-robotics.net)](http://downloads.fars-robotics.net/wifi-drivers/)” we can search for the right driver.

If you’re using the monitor with mouse and keyboard, you can use the browser on your Pi to install the driver.



Go to 8188eu-drivers/

After this search for the driver “8188eu-‘kernel number’.tar.gz”. Kernel number is the number we previously looked up. (5.10.17-v7l+ #1421)



If you’re using an SSH connection, search for the driver and copy the webpage link.

In my case: <http://downloads.fars-robotics.net/wifi-drivers/8188eu-drivers/8188eu-5.10.20-v7l-1404.tar.gz>

You can use this command to download the driver via an SSH connection:

wget http://downloads.fars-robotics.net/wifi-drivers/8188eu-drivers/8188eu-5.10.20-v7l-1404.tar.gz

*replace the link with your own.*

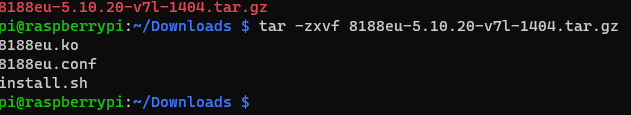
## Installation driver

When you’ve downloaded the driver, use the terminal to go to the directory of the file. Using the browser, it would be in de download folder (/home/pi/Downloads). Using an SSH connection, it will be in the current folder you’re in.

When you’re in the right directory use this command:

sudo tar -xvf 8188eu-5.10.20-v7l-1404.tar.gz

*replace with the file you’ve downloaded.*



This will create 3 files; you need to run ‘install.sh’. You can use this command:

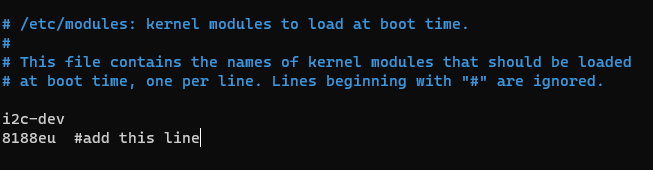
sudo ./install.sh

Once you’ve done this, all the files will be removed. And you’ll need to do 1 more step.

Use this command:

sudo nano /etcmodules

then add ‘8188eu’ on the last line.



Afterwards press “ctrl + x 🡪 y 🡪 Enter”

Now reboot the raspberry pi for your configuration to take effect.

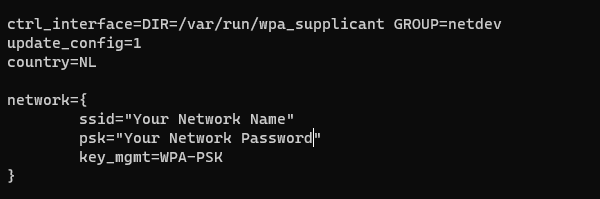
## Setting up Wi-Fi

When it’s on you can select the Wi-Fi network you want to connect to via the monitor.

Or if you’re using an SSH connection, you can type this command:

sudo nano /etc/wpa\_supplicant/wpa\_supplicant.conf

then add this to the file:



Afterwards press “ctrl + x 🡪 y 🡪 Enter”

Now reboot again.

If everything went ok your raspberry pi should be connected to your network. If it didn’t work yet, you may need to load another kernel module. So double check if you’ve installed the right one.

# Installation Access Point

## Getting started

In order for the raspberry pi to work as an access point (AP), we will need to install the AP software, along with DHCP server software to provide connecting devices with a network address.

To create an access point, we’ll need DNSMasq and HostAPD. Install all the required software in one go with this command:

sudo apt install dnsmasq hostapd

Since the configuration files are not ready yet, turn the new software off as follows:

sudo systemctl stop dnsmasq

sudo systemctl stop hostapd

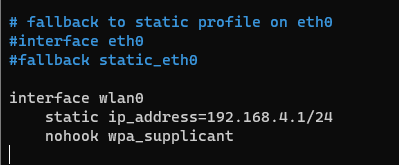
## Configuring a static IP

We are configuring a standalone network to act as a server, so the Raspberry Pi needs to have a static IP address assigned to the wireless port. This documentation assumes that we are using the standard 192.168.x.x IP addresses for our wireless network, so we will assign the server the IP address 192.168.4.1. It is also assumed that the wireless device being used is wlan0.

To configure the static IP address, edit the dhcpcd configuration file with:

sudo nano /etc/dhcpcd.conf

Go to the end of the file and edit it so that it looks like the following:



interface wlan0

static ip\_address=192.168.4.1/24

nohook wpa\_supplicant

Now restart the dhcpcd daemon and set up the new wlan0 configuration:

sudo service dhcpcd restart

## Configuring the DHCP server (DNSMasq)

The DHCP service is provided by dnsmasq. By default, the configuration file contains a lot of information that is not needed, and it is easier to start from scratch. Rename this configuration file, and edit a new one:

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

sudo nano /etc/dnsmasq.conf

Type or copy the following information into the dnsmasq configuration file and save it:

interface=wlan0 # Use the require wireless interface - usually wlan0

dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h

So, for wlan0, we are going to provide IP addresses between 192.168.4.2 and 192.168.4.20, with a lease time of 24 hours. If you are providing DHCP services for other network devices (e.g. eth0), you could add more sections with the appropriate interface header, with the range of addresses you intend to provide to that interface.

Start dnsmasq (it was stopped), it will now use the updated configuration:

sudo systemctl start dnsmasq

## Configuring the access point host software (HostAPD)

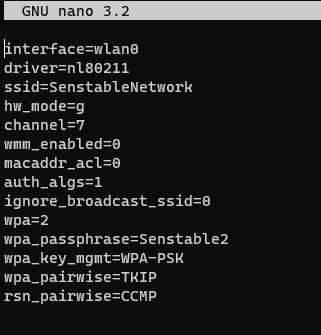
You need to edit the hostapd configuration file, located at /etc/hostapd/hostapd.conf, to add the various parameters for your wireless network. After initial install, this will be a new/empty file.

sudo nano /etc/hostapd/hostapd.conf

Add the information below to the configuration file. This configuration assumes we are using channel 7, with a network name of SenstableNetwork, and a password Senstable. Note that the name and password should not have quotes around them. The passphrase should be between 8 and 64 characters in length.

To use the 5 GHz band, you can change the operations mode from hw\_mode=g to hw\_mode=a. Possible values for hw\_mode are:

* a = IEEE 802.11a (5 GHz)
* b = IEEE 802.11b (2.4 GHz)
* g = IEEE 802.11g (2.4 GHz)
* ad = IEEE 802.11ad (60 GHz) (Not available on the Raspberry Pi)



interface=wlan0

driver=nl80211

ssid=NameOfYourNetwork

hw\_mode=g

channel=7

wmm\_enabled=0

macaddr\_acl=0

auth\_algs=1

ignore\_broadcast\_ssid=0

wpa=2

wpa\_passphrase=YourPassword

wpa\_key\_mgmt=WPA-PSK

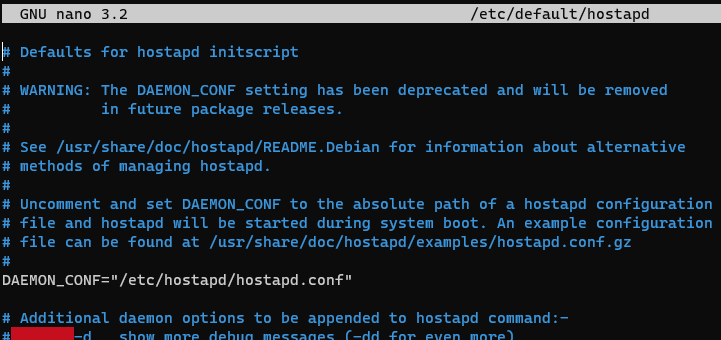
wpa\_pairwise=TKIP

rsn\_pairwise=CCMP

We now need to tell the system where to find this configuration file.

sudo nano /etc/default/hostapd

Find the line with #DAEMON\_CONF, and replace it with this:

DAEMON\_CONF="/etc/hostapd/hostapd.conf" 

## Start it up

Now enable and start hostapd:

sudo systemctl unmask hostapd

sudo systemctl enable hostapd

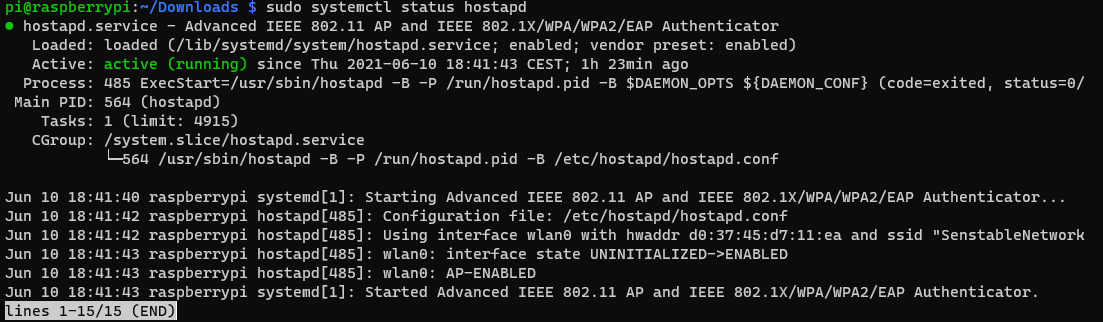
sudo systemctl start hostapd

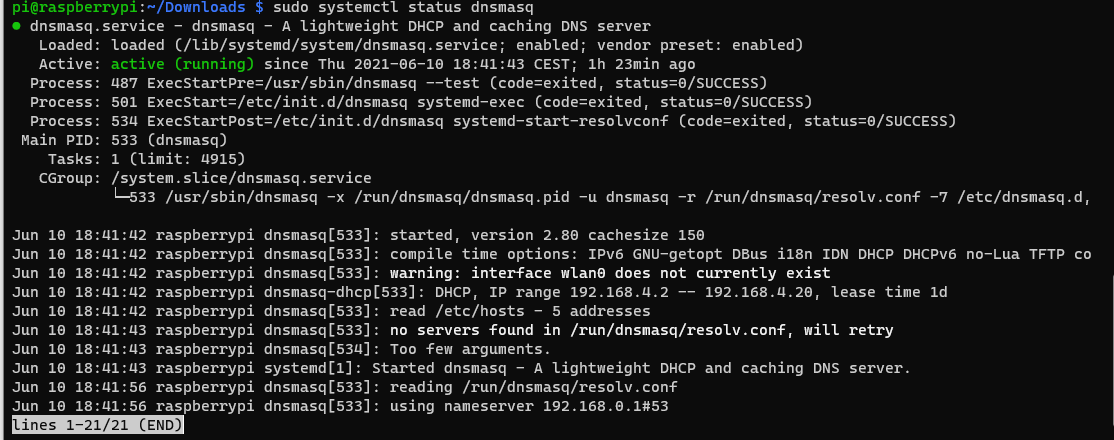
Do a quick check of their status to ensure they are active and running:

sudo systemctl status hostapd

sudo systemctl status dnsmasq

This, or something similar to this, should show up.

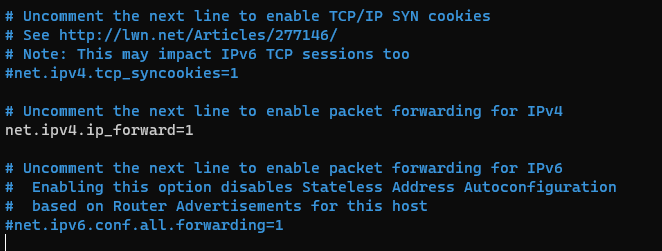




## Add routing and masquerade

Edit /etc/sysctl.conf and uncomment this line:

net.ipv4.ip\_forward=1



Add a masquerade for outbound traffic on eth0:

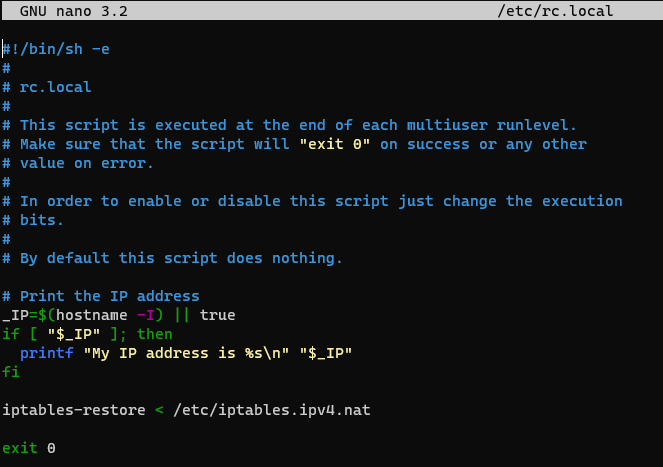
sudo iptables -t nat -A POSTROUTING -o wlan1 -j MASQUERADE

Save the iptables rule.

sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"

Edit /etc/rc.local and add this just above “exit 0” to install these rules on boot.

iptables-restore < /etc/iptables.ipv4.nat



Now reboot the raspberry pi.

Once rebooted, if you go to another device that has wireless and search for wireless networks you should be able to see your wireless access point and be able to connect to it using the credentials that you configured.



# Setup Webpage

To setup a webpage we’ll need to install some software:

sudo apt install apache2

sudo apt install php

sudo apt install libapache2-mod-php

When it asks if you want to continue just type ‘y’ and press enter.

Afterwards we need to install some libraries to use web sockets with PHP:

sudo apt install php-cli unzip

Make sure you’re in your home directory, then retrieve the installer using curl:

cd ~

curl -sS https://getcomposer.org/installer -o composer-setup.php

Next, we’ll verify that the downloaded installer matches the SHA-384 hash for the latest installer found on the Composer Public Keys / Signatures page. To facilitate the verification step, you can use the following command to programmatically obtain the latest hash from the Composer page and store it in a shell variable:

HASH=`curl -sS https://composer.github.io/installer.sig`

If you want to verify the obtained value, you can run:

echo $HASH

output is something like this:



Now execute the following PHP code, as provided in the Composer download page, to verify that the installation script is safe to run:

php -r "if (hash\_file('SHA384', 'composer-setup.php') === '$HASH') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP\_EOL;"

You’ll see the following output:

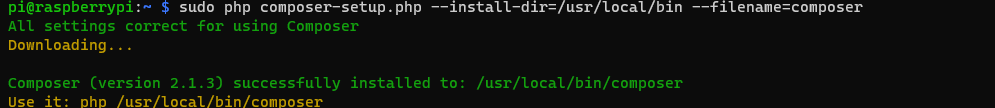


If the output says Installer corrupt, you’ll need to download the installation script again and double check that you’re using the correct hash. Then, repeat the verification process. When you have a verified installer, you can continue.

To install composer globally, use the following command which will download and install Composer as a system-wide command named composer, under /usr/local/bin:

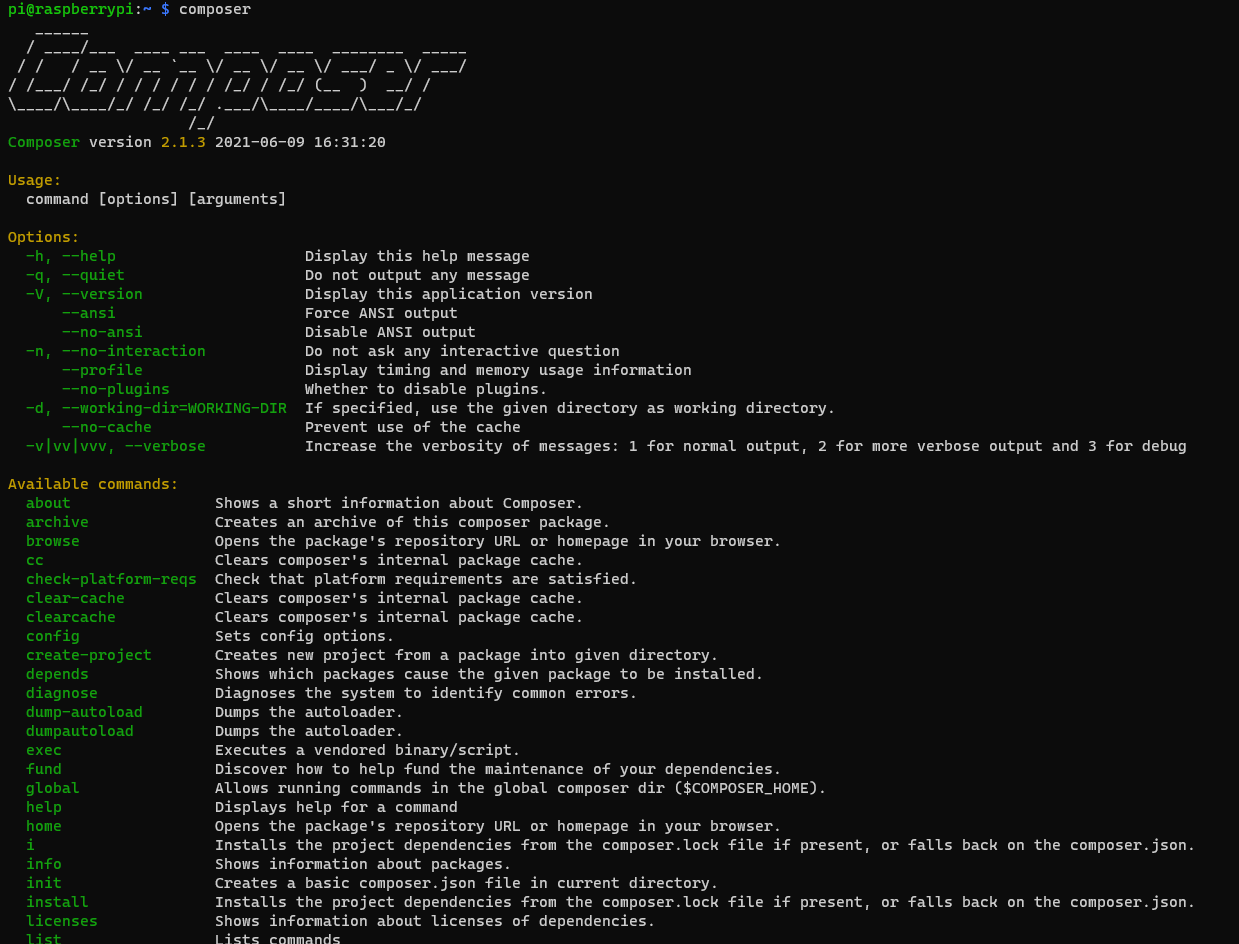
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer

You’ll see something like this:



To test your installation, run:

Composer



The last step is to go to the directory ‘/var/www/html’:

cd /var/www/html

Remove ‘index.html’

sudo rm index.html

Then copy the given files to this directory using something like FileZilla.



# Setup NodeJS server

Go to the directory ‘/var/www’

cd /var/www

s

create a directory called ‘nodejs’:

sudo mkdir nodejs

Make sure you’re in the directory you just made before you install the packages:

cd /var/www/nodejs

Install NodeJS:

sudo apt-get install nodejs

Install npm for the packages we need to install:

sudo apt install npm

Install the packages we need to make the web sockets work:

sudo npm install body-parser

sudo npm install express

sudo npm install ws

Copy and paste the given file named ‘server.js’ in the nodejs directory, it should look like this:



Next, we’re going to install PM2. This is going to automatically start the NodeJS server when we boot up the Raspberry Pi and if it crashes it will restart the server.

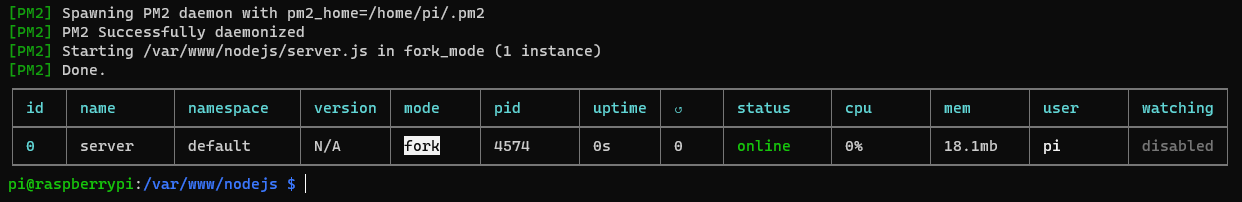
We will use npm to install PM2 globally -g.

sudo npm install -g pm2

To start ‘server.js’ with PM2:

pm2 start server.js

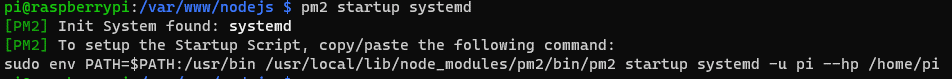
You’ll see something like this:



The pm2 startup command will generate a script that will lunch PM2 on boot together with the applications that you configure to start.

pm2 startup systemd

This will generate the following:



Copy the generated command and run it.

sudo env PATH=$PATH:/usr/bin /usr/local/lib/node\_modules/pm2/bin/pm2 startup systemd -u pi --hp /home/pi

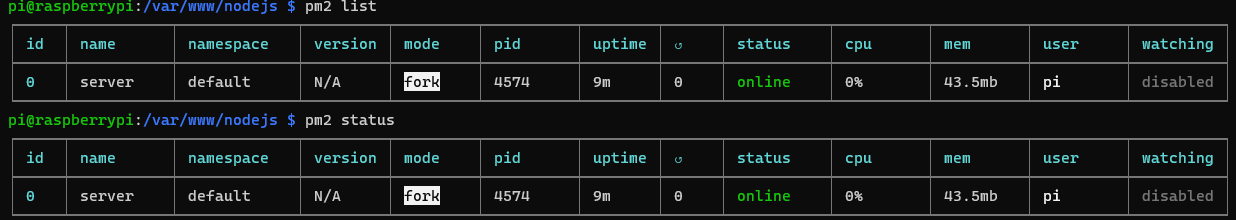
This created a system unit that will start PM2 on boot. When the system will boot PM2 will resurrect from a dump file that is not created yet. To create it run the following:

pm2 save

This will save the current state of PM2 (with server.js running) in a dump file that will be used when resurrecting PM2.

That's it! This will keep the server running and will start the server when you boot up the Raspberry.

Using pm2 list or pm2 status you’ll be able to see the status of the server.



[How to Setup a Wireless Access Point on the Raspberry Pi • Pi Supply Maker Zone (pi-supply.com)](https://learn.pi-supply.com/make/how-to-setup-a-wireless-access-point-on-the-raspberry-pi/)

[TP-Link WN725N on Raspberry PI B – JuanLu's blog (wordpress.com)](https://sirjuanlu.wordpress.com/2015/04/05/tp-link-wn725n-on-raspberry-pi-b/)

[(UPDATE) Drivers for TL-WN725N V2 - 3.6.11+ -> 4.xx.xx+ - Raspberry Pi Forums](https://www.raspberrypi.org/forums/viewtopic.php?p=462982#p462982)

[Run your Node.js application on a headless Raspberry Pi - DEV Community](https://dev.to/bogdaaamn/run-your-nodejs-application-on-a-headless-raspberry-pi-4jnn)

[How To Install and Use Composer on Ubuntu 20.04 | DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-install-and-use-composer-on-ubuntu-20-04)