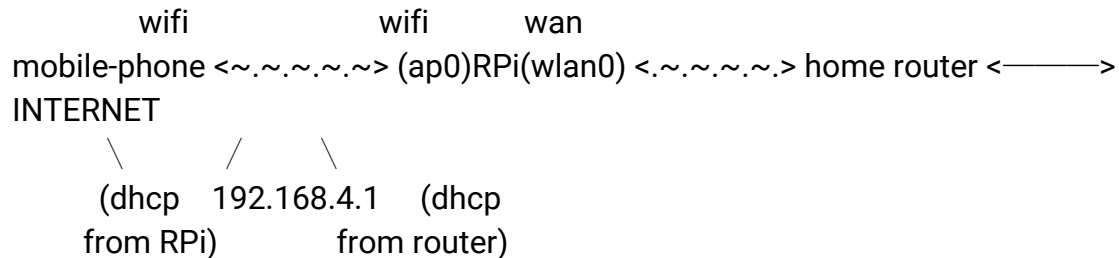


Set up Wi-Fi Access Point

The structure of the network looks like this:



Step 1: Setup systemd-networkd

Part 1: Enable systemd-networkd

To simplify commands we will work as root:

```
pi@raspberrypi:~ $ sudo -Es
```

Deinstall classic Debian networking that is managed with file `/etc/network/interfaces` and deinstall default Raspbian `dhcpcd` network management.

```
root@raspberrypi:~ # apt --autoremove purge ifupdown
root@raspberrypi:~ # rm -r /etc/network
root@raspberrypi:~ # apt --autoremove purge dhcpcd5
root@raspberrypi:~ # apt --autoremove purge isc-dhcp-client isc-dhcp-common
root@raspberrypi:~ # rm -r /etc/dhcp
root@raspberrypi:~ # apt --autoremove purge rsyslog
```

And enable systemd-networkd:

```
root@raspberrypi:~ # systemctl enable systemd-networkd.service
```

Part 2: Enable systemd-resolved

First we will enable **systemd-resolved** and then configure its three interfaces:

```
root@raspberrypi:~ # systemctl enable systemd-resolved.service
```

Check that the D-Bus should be installed by default and running:

```
rpi:~ # systemctl status dbus.service
```

Then configure the NSS software interface by deinstalling the **avahi** service and the **mdns** service and clean up `/etc/nsswitch.conf`:

```
root@raspberrypi:~ # apt --autoremove purge avahi-daemon
```

Now install the **systemd-resolved** software interface:

```
root@raspberrypi:~ # apt install libnss-resolve
```

Next configure the DNS stub listener interface by symlinking `/etc/resolv.conf` to the stub listener:

```
root@raspberrypi:~ # ln -sf /run/systemd/resolve/stub-resolv.conf  
/etc/resolv.conf
```

Part 3: Configure Network Interfaces

Create an interface file for a WiFi connection.

```
root@raspberrypi:~ # mv /etc/wpa_supplicant/wpa_supplicant.conf  
/etc/wpa_supplicant/wpa_supplicant-wlan0.conf  
root@raspberrypi:~ # systemctl disable wpa_supplicant.service  
root@raspberrypi:~ # systemctl enable wpa_supplicant@wlan0.service  
root@raspberrypi:~ # cat > /etc/systemd/network/08-wifi.network <<EOF  
[Match]  
Name=wl*  
[Network]  
DHCP=yes  
EOF
```

Then reboot.

In addition install **hostapd**:

```
rpi ~$ sudo -Es  
rpi ~# apt install hostapd  
rpi ~# systemctl unmask hostapd  
rpi ~# systemctl enable hostapd
```

Step 2: Install hostapd for the Access Point

Create this file with your settings for `ssid=`, `country_code=` and `wpa_passphrase=`. As `channel=` select the same channel `wpa_supplicant` with **wlan0** will connect to your internet router.

```
rpi ~# cat > /etc/hostapd/hostapd.conf <<EOF
interface=ap0
driver=nl80211
ssid=Your_Network
country_code=US
hw_mode=g
channel=1 //same channel as wpa_supplicant with wlan0
auth_algs=1
wpa=2
wpa_passphrase=Your_Network_Password
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
EOF
rpi ~# chmod 600 /etc/hostapd/hostapd.conf
```

Set `DAEMON_CONF="/etc/hostapd/hostapd.conf"` in `/etc/default/hostapd`

```
rpi ~# sed -i
's/^#DAEMON_CONF=.*$/DAEMON_CONF="/etc/hostapd/hostapd.conf"/'
/etc/default/hostapd
```

Edit the `hostapd.service`:

```
rpi ~# systemctl --full edit hostapd.service
```

and comment the line `After=network.target` with `#` so it looks like:

```
#After=network.target
```

Save it and quit the editor.

Next add interface **ap0** to the `hostapd.service` with:

```
rpi ~# systemctl edit hostapd.service
```

In the empty editor insert these statements, save it and quit the editor:

```
[Unit]
Wants=wpa_supplicant@wlan0.service
[Service]
```

```
ExecStartPre=/sbin/iw dev wlan0 interface add ap0 type __ap
ExecStopPost=-/sbin/iw dev ap0 del
```

Step 3: Setup wpa_supplicant for Client Connection

Create this file with your settings for `country=`, `ssid=` and `psk=` and enable it:

```
rpi ~# cat >/etc/wpa_supplicant/wpa_supplicant-wlan0.conf <<EOF
country=US
ctrl_interface=DIR=/var/run/wpa_supplicant
GROUP=netdev
update_config=1
network={
    ssid="TestNet"
    psk="reallyNotMyPassword"
    key_mgmt=WPA-PSK # see ref (4)
}
EOF
rpi ~# chmod 600 /etc/wpa_supplicant/wpa_supplicant-wlan0.conf
rpi ~# systemctl disable wpa_supplicant.service
rpi ~# systemctl enable wpa_supplicant@wlan0.service
rpi ~# rfkill unblock 0
```

Extend wpa_supplicant with:

```
rpi ~# systemctl edit wpa_supplicant@wlan0.service
```

In the empty editor insert these statements. Have attention to the minus sign after equal `=-` on some statements. Save it and quit the editor:

```
[Unit]
BindsTo=hostapd.service
After=hostapd.service
[Service]
ExecStartPost=/sbin/iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
ExecStopPost=-/sbin/iptables -t nat -D POSTROUTING -o wlan0 -j MASQUERADE
```

Step 4: Setup Static Interfaces

Create these files:

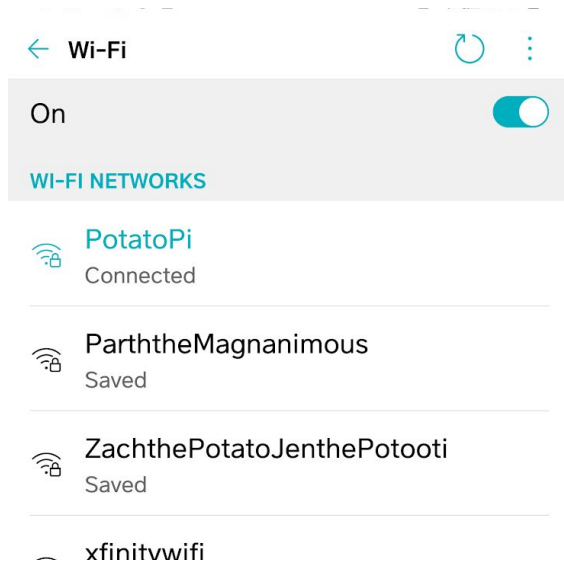
```
rpi ~# cat > /etc/systemd/network/08-wlan0.network <<EOF
[Match]
Name=wlan0
```

```
[Network]
IPForward=yes
DHCP=yes
EOF

rpi ~# cat > /etc/systemd/network/12-ap0.network <<EOF
[Match]
Name=ap0
[Network]
Address=192.168.4.1/24
DHCPServer=yes
[DHCPServer]
DNS=84.200.69.80 1.1.1.1
EOF
```

Then reboot.

You should be able to connect to your RPi access point now (my SSID is PotatoPi in this case):



Set up Apache Server

Part 1: Install Apache

First, update the available packages by typing the following command into the Terminal:

```
sudo apt update
```

Then, install the apache2 package with this command:


```
sudo apt install apache2 -y
```

Part 2: Test the web server


By default, Apache puts a test HTML file in the web folder. This default web page is served when you browse to <http://localhost/>.

To find the Pi's IP address, type `hostname -I` at the command line.

You should see the following on the default page:



Apache2 Debian Default Page



It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Debian systems. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Debian's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Debian tools. The configuration system is **fully documented in `/usr/share/doc/apache2/README.Debian.gz`**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the `apache2-doc` package was installed on this server.

The configuration layout for an Apache2 web server installation on Debian systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|   |-- ports.conf
|-- mods-enabled
|   |-- *.load
|   |-- *.conf
|-- conf-enabled
|   |-- *.conf
|-- sites-enabled
|   |-- *.conf
```



This means you have Apache working!

Part 3: Changing the default web page

This default web page is just an HTML file on the Pi's filesystem. It is located at `/var/www/html/index.html`.

Navigate to this directory in a terminal window and have a look at what's inside:

```
cd /var/www/html
ls -al
```

This will show you:

```
total 12
drwxr-xr-x  2 root root 4096 Jan  8 01:29 .
drwxr-xr-x 12 root root 4096 Jan  8 01:28 ..
-rw-r--r--  1 root root  177 Jan  8 01:29 index.html
```

This shows that by default there is one file in `/var/www/html` called `index.html` and it is owned by the root user (as is the enclosing folder). In order to edit the file, you need to change its ownership to your own username. Change the owner of the file (the default pi user is assumed here) using `sudo chown pi: index.html`.

You can now try editing this file and then refreshing the browser to see the web page change.

Part 4: Setting up a virtual host (Custom Domain Name)

Step 1: Create a New Directory

First, it is necessary to create a directory where we will keep the new website's information. This location will be your Document Root in the Apache virtual configuration file. By adding a `-p` to the line of code, the command automatically generates all the parents for the new directory.

You will need to designate an actual DNS approved domain (or an IP address) to test that a virtual host is working. In this tutorial, we will use `example.com` as a placeholder for a correct domain name.

```
sudo mkdir -p /var/www/example.com/public_html
```

*If you want to use an unapproved domain name to test the process, you will find information on how to make it work on your local computer in Step Seven.

Step 2: Grant Permissions

Now you must grant ownership of the directory to the user, as opposed to just keeping it on the root system.

```
sudo chown -R $USER:$USER /var/www/example.com/public_html
```

Additionally, it is important to make sure that everyone will be able to read your new files.

```
sudo chmod -R 755 /var/www
```

Now you are all done with permissions.

Step 3: Create the Page

Within your configurations directory, create a new file called index.html

```
sudo nano /var/www/example.com/public_html/index.html
```

It's also useful to add some text to the file, in order to have something to look at when the IP redirects to the virtual host.

```
<html>
  <head>
    <title>www.example.com</title>
  </head>
  <body>
    <h1>Success: You Have Set Up a Virtual Host</h1>
  </body>
</html>
```

Save & Exit.

Step 4: Create the New Virtual Host File

The next step is to set up the apache configuration. We're going to work off a duplicate—go ahead and make a copy of the file (naming it after your domain name) in the same directory:

```
sudo cp /etc/apache2/sites-available/default
/etc/apache2/sites-available/example.com
```

Step 5: Turn on Virtual Hosts

Open up the new config file:

```
sudo nano /etc/apache2/sites-available/example.com
```

We are going to set up a virtual host in this file. To begin, insert a line for the ServerName under the ServerAdmin line.


```
ServerName example.com
```

The ServerName specifies the domain name that the virtual host uses.

If you want to make your site accessible from more than one name (ie with www in the URL), you can include the alternate names in your virtual host file by adding a ServerAlias Line. The beginning of your virtual host file would then look like this:

```
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    ServerName example.com
    ServerAlias www.example.com
    [...]
```

The next step is to fill in the correct Document Root. For this section, write in the extension of the new directory created in Step One. If the document root is incorrect or absent you will not be able to set up the virtual host. The section should look like this:

```
DocumentRoot /var/www/example.com/public_html
```

You do not need to make any other changes to this file. Save and Exit.

The last step is to activate the host with the built-in apache shortcut:

```
sudo a2ensite example.com
```

Step 6: Restart Apache

Although there have been a lot of changes to the configuration and the virtual host is set up, none of the changes will take effect until Apache is restarted:

```
sudo service apache2 restart
```

Step 7: Setting Up the Local Hosts

If you have pointed your domain name to your virtual private server's IP address you can skip this step. However, if you want to try out your new virtual hosts without having to connect to an actual domain name, you can set up local hosts on your computer alone.

For this step, make sure you are on the computer itself and not your droplet. To proceed with this step, you need to know your computer's administrative password; otherwise, you will be required to use an actual domain name to test the virtual hosts.

If you are on a Mac or Linux, access the root user (**su**) on the computer and open up your hosts file:

```
nano /etc/hosts
```

If you are on a Windows Computer, you can find the directions to alter the host file on the Microsoft site

You can add the local hosts details to this file, as seen in the example below. As long as that line is there, directing your browser toward, say, example.com will give you all the virtual host details for the corresponding IP address.

```
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting. Do not change this entry.
##
127.0.0.1        localhost

#Virtual Hosts
12.34.56.789     example.com
```

However, it may be a good idea to delete these made up addresses out of the local hosts folder when you are done in order to avoid any future confusion. Change 127.0.1.1 from raspberrypi to example.com.

Sync files from Google Drive

Step 1: Installing rclone

To install rclone on Linux/macOS/BSD systems, run:

```
curl https://rclone.org/install.sh | sudo bash
```

For beta installation, run:

```
curl https://rclone.org/install.sh | sudo bash -s beta
```

Note that this script checks the version of rclone installed first and won't re-download if not needed

Step 2: Using rclone

Issue the command *man rclone* to find out how the tool is used. The gist of the command line is:

To see the top level directory of your Drive, issue the command *rclone lsd drive*.

To list all of the files of your Drive, issue the command *rclone ls drive*.

To copy a local file to Drive, issue the command `rclone copy /path/to/local/file drive:/path/to/remote/folder`.

Sources

<https://raspberrypi.stackexchange.com/questions/89803/access-point-as-wifi-router-repeater-optional-with-bridge>

<https://www.digitalocean.com/community/tutorials/how-to-set-up-apache-virtual-hosts-on-debian-7#optional-step-seven%E2%80%944setting-up-the-local-hosts>

<https://www.techrepublic.com/article/how-to-sync-from-linux-to-google-drive-with-rclone/>