Configuring Raspberry Pi as Hotspot

Terminal commands are in blue

1. Update and Install Packages:

```
sudo apt update
sudo apt install hostapd dnsmasq
sudo systemctl stop hostapd
sudo systemctl stop dnsmasq
```

2. Verify installation

```
which hostapd
which dnsmasq
```

   Should return valid paths.

3. Disable NetworkManager for wlan0. Create /etc/NetworkManager/conf.d/10-ignore-wlan0.conf to say:

```
[keyfile]
unmanaged-devices=interface-name:wlan0
```

   Run: `sudo systemctl restart NetworkManager`

   Run: `sudo pkill dhcpcd && sudo dhcpcd`

4. Configure static IP for wlan0 in */etc/dhcpcd.conf to say*

```
interface wlan0
static ip_address=192.168.4.1/24
nohook wpa_supplicant
```

   Run: `sudo service dhcpcd restart`

   Check: `ip addr show wlan0`

   Should show "inet 192.168.4.1/24"

   Configure dnsmasq in */etc/dnsmasq.conf*

   Copy: `sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig`

   Sudo nano */etc/dnsmasq.conf:*

```
interface=wlan0
dhcp-range=192.168.4.10,192.168.4.50,255.255.255.0,24h
```

5. Configure hostapd in */etc/hostapd/hostapd.conf to say*

```
interface=wlan0
driver=nl80211
ssid=MyPiHotspot #CHOOSE HOTSPOT NAME
hw_mode=g
channel=7
wmm_enabled=0
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=yourpassword #CHOOSE HOTSPOT PASSWORD
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

   Also: *sudo nano /etc/default/hostapd* to include:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf”
```

6. Enable everything on boot

```
sudo systemctl unmask hostapd
sudo systemctl enable hostapd
sudo systemctl enable dnsmasq
```

7. Test manually before reboot

```
sudo systemctl start hostapd
sudo systemctl start dnsmasq
```

8. Reboot and test. At this point, your Raspberry Pi hotspot should be visible in your phone’s WiFi settings! The password is what you set in step 6.

2. Install Necessary Packages:

You'll need hostapd (to create the access point) and dnsmasq (to handle DHCP requests).

sudo apt install hostapd dnsmasq -y

hostapd: Creates the wireless access point.

dnsmasq: Provides DHCP service for devices connecting to your Wi-Fi.

3. Disable NetworkManager (If Interfering):

If you're using NetworkManager (which is common in desktop setups), it can interfere with the access point configuration. You'll need to disable it:

```
sudo systemctl stop NetworkManager
sudo systemctl disable NetworkManager
```

You may also want to disable wpa_supplicant to prevent it from managing the wireless interface:

```
sudo systemctl stop wpa_supplicant
sudo systemctl disable wpa_supplicant
```

4. Enable hostapd and dnsmasq to Start on Boot:

```
sudo systemctl enable hostapd
sudo systemctl enable dnsmasq
```

5. Configure hostapd:

Edit the hostapd.conf file:

```
sudo nano /etc/hostapd/hostapd.conf
```

Add the following configuration:

```
interface=wlan0
driver=nl80211
ssid=YourSSID
hw_mode=g
channel=7
auth_algs=1
wmm_enabled=1
macaddr_acl=0
ignore_broadcast_ssid=0
```

```
wpa=2
wpa_passphrase=YourPassword
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
```

ssid=YourSSID: Set your desired Wi-Fi network name.

wpa_passphrase=YourPassword: Set your password for the Wi-Fi network.

Point hostapd to this file by editing the default configuration:

```
sudo nano /etc/default/hostapd
```

Find the line:

```
#DAEMON_CONF=""
```

Change it to:

```
   DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

6. Configure dnsmasq:

Edit the dnsmasq.conf file:

```
sudo nano /etc/dnsmasq.conf
```

Add the following configuration at the end of the file:

```
interface=wlan0       # Use the wlan0 interface
dhcp-range=192.168.4.50,192.168.4.150,12h  # Set the DHCP range (optional)
```

Ensure that dnsmasq runs only on wlan0 by disabling it on other interfaces (e.g., Ethernet):

```
sudo nano /etc/dhcpcd.conf
```

Add the following lines at the end of the file to disable DHCP on Ethernet:

```
  interface eth0
  denyinterfaces wlan0
```

7. Configure Static IP for wlan0:

To ensure your Pi's wlan0 interface has a fixed IP, edit the dhcpcd.conf file:

```
sudo nano /etc/dhcpcd.conf
```

At the end of the file, add the following:

```
interface wlan0
static ip_address=192.168.4.1/24
```

This sets your Raspberry Pi's IP address to 192.168.4.1 for the access point.

8. Restart dhcpcd and Reboot:

Restart the DHCP client service and reboot the Pi to apply the changes.

```
sudo systemctl restart dhcpcd
sudo reboot
```

9. Verify Wi-Fi Interface (wlan0) is Up:

After the Pi reboots, check if the wlan0 interface is up:

```
ip link show wlan0
```

If the state is UP, then it's ready.

10. Start hostapd:

Start the hostapd service to enable the Wi-Fi access point:

```
sudo systemctl start hostapd
```

11. Verify hostapd Status:

Check the status of hostapd to confirm that it’s running correctly:

```
sudo systemctl status hostapd
```

If everything is set up correctly, you should now have an access point running on your Raspberry Pi!
12. Troubleshooting:

Check logs if something isn’t working:

```
sudo journalctl -xeu hostapd.service | tail -20
```

Check for conflicting services:
If hostapd fails to start, ensure no other service (like NetworkManager) is managing wlan0.

Ensure that wlan0 supports AP mode:
Run the following to check if your Wi-Fi adapter supports Access Point mode:

```
iw list | grep -A 10 "Supported interface modes"
```

Summary:

Install packages: hostapd, dnsmasq.

Disable NetworkManager (if interfering).

Configure hostapd with a suitable config (hostapd.conf).

Configure dnsmasq for DHCP.

Set static IP for wlan0 in dhcpcd.conf.

Restart services and reboot.

Start hostapd and ensure it's running.

Also:

Configure /etc/network/interfaces to include:

```
auto wlan0
iface wlan0 inet static
   address 192.168.4.1
   netmask 255.255.255.0
   gateway 192.168.4.1
```

run ip addr show to verify wlan0 has an ip

Step 8: Set Up Flask App to Run on Boot Using systemd

Create a systemd service file:

Open a new file for your service definition:

```
sudo nano /etc/systemd/system/flaskapp.service
```

Add the following configuration to the file:

```
[Unit]
Description=Flask App
After=network.target

[Service]
User=pi
```

```
WorkingDirectory=/home/pi/flask_app   # Adjust if your app is in a different directory
ExecStart=/usr/bin/python3 /home/pi/flask_app/app.py  # Adjust path to app.py
Restart=always

[Install]
WantedBy=multi-user.target
```

Description: Describes the service.

After: Ensures the service starts after the network is up.

User: Specifies which user the app will run as (pi is typical).

WorkingDirectory: The directory where your Flask app is located.

ExecStart: Command to start the Flask app using Python (python3 with path to app.py).

Restart: Ensures the app restarts if it crashes.

Save and close the service file:

Press Ctrl + X to exit, then press Y to save, and hit Enter to confirm.

Enable the service:

This command ensures the Flask app starts on boot:

```
sudo systemctl enable flaskapp.service
```

Start the service immediately:

You can start your Flask app manually by running:

```
sudo systemctl start flaskapp.service
```

Check the service status:

To verify that your app is running, use:

```
sudo systemctl status flaskapp.service
```

You should see output indicating that the service is active and running. If it’s not, check the logs (step 6).

Reboot your Raspberry Pi:

To ensure that your Flask app starts automatically on reboot, reboot your Pi:

```
sudo reboot
```

Verify Flask app is running:

After rebooting, open a browser and go to:

```
http://<your-pi-static-ip>:5000
```

You should see your Flask app’s webpage.

Optional: Troubleshooting

If the app isn’t running after a reboot, check the service logs:

```
journalctl -u flaskapp.service
```

This will show the log output of the service and any error messages that may help with troubleshooting.