

## How to configure Network

You may have set up a CentOS server and, in the process, accidentally set it up with DHCP. If your CentOS server uses a GUI, changing that IP address from dynamic to static is very simple. But what if your server is a text-only machine? What do you do then? Fortunately, it's not all that hard to configure that GUI-less server with a static IP address—you just have to know where it's configured and know the syntax of the configuration. Of course, by nature of what we're working on this is all done manually, so be prepared to type.

I'll be working on [CentOS 7](#). I'll assume you already have the operating system installed and working properly, have access to the machine, and have an administrative account. With that out of the way, let's set up that static IP address.

What is a loopback interface used for?

The **loopback** device is a special, virtual network **interface** that your computer **uses** to communicate with itself. It is **used** mainly for diagnostics and troubleshooting, and to connect to servers running on the local machine.

The [loopback device](#) is a special, [virtual network interface](#) that your computer uses to communicate with itself. It is used mainly for diagnostics and troubleshooting, and to connect to servers running on the local machine.

### The Purpose of Loopback

When a network interface is disconnected—for example, when an [Ethernet](#) port is unplugged or [Wi-Fi](#) is turned off or not associated with an [access point](#)--no communication on that interface is possible, not even communication between your computer and itself. The loopback interface does not represent any actual hardware, but exists so applications running on your computer can always connect to servers on the same machine.

This is important for troubleshooting (it can be compared to looking in a mirror). The loopback device is sometimes explained as purely a diagnostic tool. But it is also helpful when a server offering a resource you need *is running on your own machine*.

For example, if you run a web server, you have all your web documents and could examine them file by file. You may be able to load the files in your browser too, though with server-side active content, it won't work the way it does when someone accesses it normally.

So if you want to experience the same site others do, the best course is usually to connect to your own server. The loopback interface facilitates that.

### Addresses on Loopback

For [IPv4](#), the loopback interface is assigned all the [IPs](#) in the `127.0.0.0/8` [address block](#). That is, `127.0.0.1` through `127.255.255.254` *all* represent your computer. For most

purposes, though, it is only necessary to use one IP address, and that is 127.0.0.1. This IP has the [hostname](#) of [localhost](#) mapped to it.

Thus, to log in as bob via [SSH](#) to the SSH server running on your own machine, you would use:

```
ssh bob@localhost
```

Like other network adapters, the loopback device shows up in the output of [ifconfig](#). Its name is [lo](#).

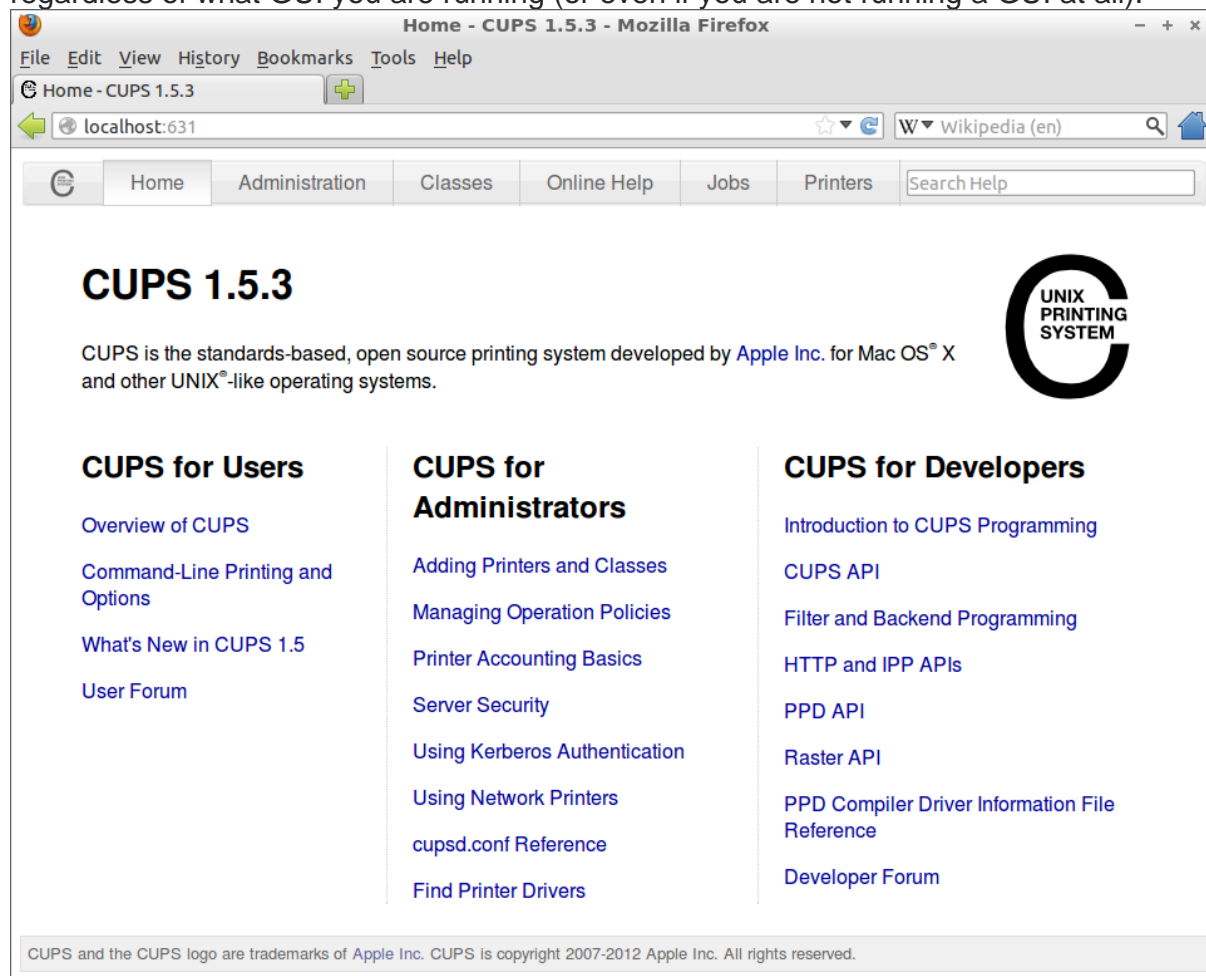
```
ek@Del:~$ ifconfig lo
lo          Link encap:Local Loopback
            inet addr:127.0.0.1  Mask:255.0.0.0
            inet6 addr: ::1/128 Scope:Host
            UP LOOPBACK RUNNING  MTU:16436  Metric:1
            RX packets:50121 errors:0 dropped:0 overruns:0 frame:0
            TX packets:50121 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:4381349 (4.3 MB)  TX bytes:4381349 (4.3 MB)
```

## An Example: CUPS

One common, production (i.e., not just diagnostic) use of [localhost](#) on Ubuntu is to perform advanced printer configuration. In a web browser, go to:

```
http://localhost:631
```

[CUPS](#) runs a web server on port 631, and this can be used to configure printing, regardless of what GUI you are running (or even if you are not running a GUI at all).



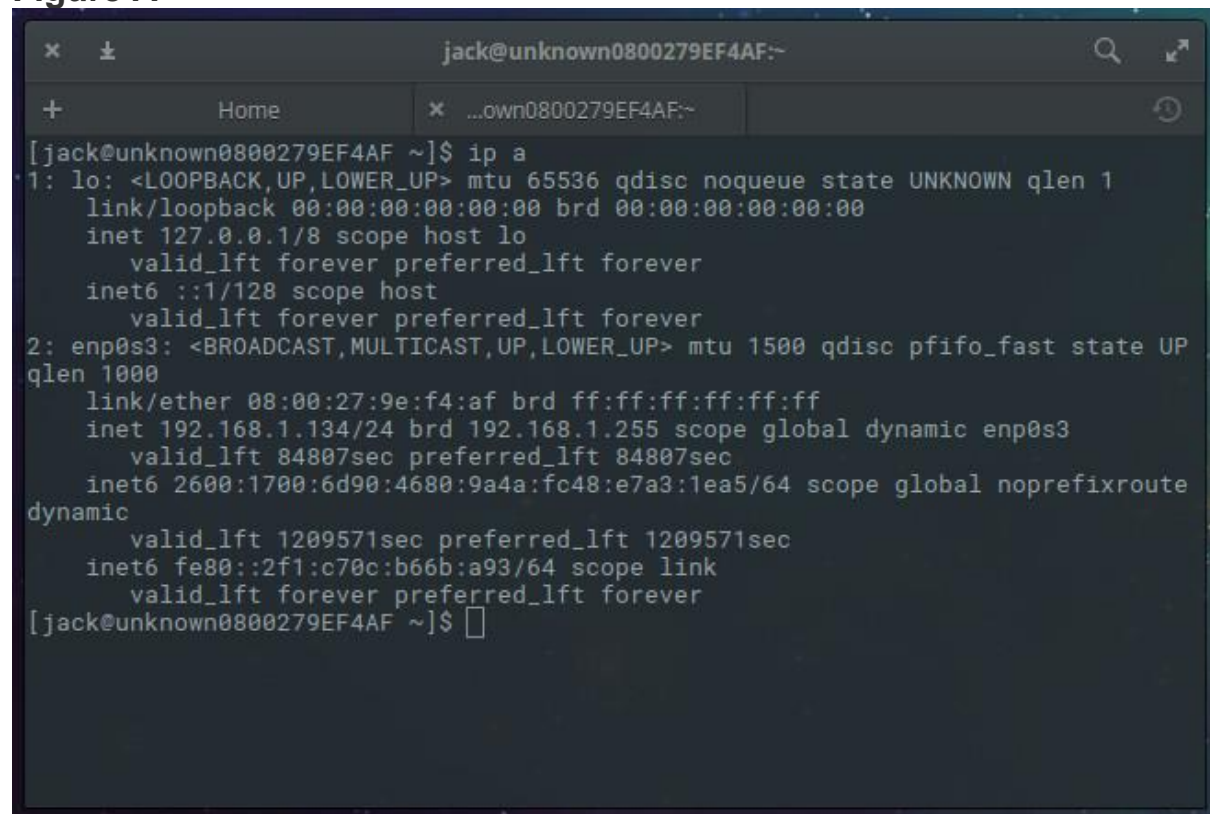
If you try connecting to [http://127.0.0.1:631](#), this will work too. However, if you try to connect to [http://127.0.0.2](#), it will not. All the [127.\\*.\\*.\\*](#) addresses identify your

computer on the loopback interface, but a server program can decide to bind just to a specific IP address.

## Find your interface

The first thing we must do is find out the name of our ethernet interface. A static IP address cannot be configured without this name. To do this, log into your server and issue the command `ip a`. The output of this command (**Figure A**) will include the name of the interface.

**Figure A**



```
jack@unknown0800279EF4AF:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
   qlen 1000
    link/ether 08:00:27:9e:f4:af brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.134/24 brd 192.168.1.255 scope global dynamic enp0s3
        valid_lft 84807sec preferred_lft 84807sec
    inet6 2600:1700:6d90:4680:9a4a:fc48:e7a3:1ea5/64 scope global noprefixroute
        dynamic
        valid_lft 1209571sec preferred_lft 1209571sec
    inet6 fe80::2f1:c70c:b66b:a93/64 scope link
        valid_lft forever preferred_lft forever
[jack@unknown0800279EF4AF ~]$
```

Our network information listed with the `ip a` command.

As you can see, from my output, the name of my interface is `enp0s3`. Now that we know the name of our interface, we can configure the static address.

## Configuring the address

Within the directory `/etc/sysconfig/network-scripts/` you should find the file `ifcfg-INTERFACENAME` (Where `INTERFACENAME` is the name of your interface). In my instance, the file is `ifcfg-enp0s3`. It is important that you configure that file, and not the `ifcfg-eth` file. Open the correct file for editing with the command `sudo nano /etc/sysconfig/network-scripts/ifcfg-enp0s3`. We need to modify that file in order to not only change the protocol from

dhcp to static, but to add the specific IP address. So when you open up that file, you'll want to change:

**BOOTPROTO=dhcp**

To:

**BOOTPROTO=static**

Now you'll need to add the entries to set not only the IP address, but the netmask, gateway, and DNS addresses. At the bottom of that file, add the following:

**IPADDR=192.168.1.200**

**NETMASK=255.255.255.0**

**GATEWAY=192.168.1.1**

**DNS1=1.0.0.1**

**DNS2=1.1.1.1**

**DNS3=8.8.4.4**

NOTE: All fields in bold, you will edit to reflect your networking needs. If you have fewer or more DNS entries, add or remove them as needed.

Save and close that file. In order to make the changes take effect, issue the command `sudo systemctl restart network`. Once the networking system has restarted, issue the command `ip a` to see that your IP address has changed to reflect your configuration.

And that's all there is to setting a static IP address on CentOS. That wasn't so hard, now was it? Don't think this technique is limited only to GUI-less CentOS servers. You can use the same method to set a static IP address on a CentOS server with a GUI as well.

As a Linux administrator you've got various tools to use in order to configure your network connections, such as: `nmtui`, your NetworkManager with GNOME graphical user interface and of course `nmcli` (network manager command line tool).

## Configure Network Ethernet Connection Using nmcli Tool

## Configure Network Ethernet Connection Using nmcli Tool

I have seen many administrators using nmtui for simplicity. However using nmcli saves your time, gives you confidence, can use it in scripts and it's the first tool to use in order to troubleshoot your Linux server networking and bring back rapidly its functionality.

Always read carefully man pages (its the No1 help for you).

The syntax of nmcli is:

```
# nmcli [OPTIONS] OBJECT {COMMAND | help}
```

Where OBJECT is one of: general, networking, radio, connection, device, agent.

A good starting point would be to check our devices:

```
# nmcli dev status
```

DEVICE	TYPE	STATE	CONNECTION
docker0	bridge	connected	docker0
virbr0	bridge	connected	virbr0
enp0s3	ethernet	connected	enp0s3
virbr0-nic	ethernet	disconnected	--
lo	loopback	unmanaged	--

As we can see in the first column is a list of our network devices. We have one network cards with name enp0s3. In your machine you could see other names.

Naming depends on the type of the network card (if it is onboard, pci card , etc). In the last column we see our configuration files which is used by our devices in order to connect to the network.

It is simple to understand that our devices by themselves can do nothing. They need us to make a configuration file to tell them how to achieve network connectivity. We call these files also as “connection profiles”. We find them in `/etc/sysconfig/network-scripts` directory.

```
# ls /etc/sysconfig/network-scripts/
```

Sample Output

```
ifcfg-enp0s3 ifdown-isdn  ifup      ifup-plip  ifup-tunnel
ifcfg-lo     ifdown-post  ifup-aliases ifup-plusb ifup-wireless
ifdown       ifdown-ppp   ifup-bnep   ifup-post  init.ipv6-global
ifdown-bnep  ifdown-routes ifup-eth    ifup-ppp   network-functions
ifdown-eth   ifdown-sit   ifup-ib     ifup-routes network-functions-ipv6
ifdown-ib    ifdown-Team  ifup-ippv   ifup-sit
ifdown-ippv  ifdown-TeamPort ifup-ipv6   ifup-Team
ifdown-ipv6  ifdown-tunnel ifup-isdn   ifup-TeamPort
```

As you can see here the files with name starting with `ifcfg-` (interface configuration) are connection profiles. When we create a new connection or modify an existing one with `nmcli` or `nmtui`, the results are saved here as connection profiles.

Check these files on your machine

from my machine, one with a dhcp configuration and one with static ip.

```
# cat ifcfg-static1
```

```
# cat ifcfg-Myoffice1
```

Check Network Configuration

Check Network Configuration

We realize that some properties have different values and some others don't exist if it isn't necessary. Let's have a quick look to most important of them.

**TYPE**, we have ethernet type here. We could have wifi, team, bond and others.

**DEVICE**, the name of the network device which is associated with this profile.

BOOTPROTO, if it has value "dhcp" then our connection profile takes dynamic IP from dhcp server, if it has value "none" then it takes no dynamic IP and probably we assign a static IP.

IPADDR, is the static IP we assign to our profile.

PREFIX, the subnet mask. A value of 24 means 255.255.255.0. You can understand better the subnet mask if you write down its binary format. For example values of 16, 24, 26 means that the first 16, 24 or 26 bits respectively are 1 and the rest 0, defining exactly what the network address is and what is the range of ip which can be assigned.

GATEWAY, the gateway IP.

DNS1, DNS2, two dns servers we want to use.

ONBOOT, if it has value "yes" it means, that on boot our computer will read this profile and try to assign it to its device.

Now, let's move on and check our connections:

**# nmcli con show**

**Show Active Network Connections**

**Show Active Network Connections**

The last column of devices helps us understand which connection is "UP" and running and which is not. In the above image you can see the two connections which are active: Myoffice1 and enp0s8.

Hint: If you want to see only the active connections, type:

**# nmcli con show -a**

**Hint: You can use the auto-complete hitting Tab when you use nmcli, but is better to use minimal format of the command. Thus, the following commands are equal:**

**# nmcli connection show**

**# nmcli con show**

**# nmcli c s**

If I check the ip addresses of my devices:

**# ip a**

Check Server IP Address

## Check Server IP Address

I see that my device enp0s3 took the 192.168.1.6 IP from dhcp server, because the connection profile Myoffice1 which is up has a dhcp configuration. If I bring “up” my connection profile with name static1 then my device will take the static IP 192.168.1.40 as it is defined in the connection profile.

```
# nmcli con down Myoffice1 ; nmcli con up static1
```

```
# nmcli con show
```

Let's see the IP address again:

```
# ip a
```

## Check Network Static IP Address

## Check Network Static IP Address

We can make our first connection profile. The minimum properties we must define are type, ifname and con-name:

type – for the type of connection.

ifname – for the device name which is assigned our connection.

con-name – for the connection name.

Let's make a new ethernet connection with name Myhome1, assigned to device enp0s3:

```
# nmcli con add type ethernet con-name Myhome1 ifname enp0s3
```

Check its configuration:

```
# cat ifcfg-Myhome1
```

## Create New Network Connection

## Create New Network Connection

As you can see it has BOOTPROTO=dhcp, because we didn't give any static ip address.



Hint: We can modify any connection with the “nmcli con mod” command. However if you modify a dhcp connection and change it to static don’t forget to change its “ipv4.method” from “auto” to “manual”. Otherwise you will end up with two IP addresses: one from dhcp server and the static one.

Let’s make a new Ethernet connection profile with name static2, which will be assigned to device enp0s3, with static IP 192.168.1.50, subnet mask 255.255.255.0=24 and gateway 192.168.1.1.

```
# nmcli con add type ethernet con-name static2 ifname enp0s3 ip4 192.168.1.50/24 gw4 192.168.1.1
```

Check its configuration:

```
# cat ifcfg-static2
```

Create New Ethernet Connection

Create New Ethernet Connection

Let’s modify the last connection profile and add two dns servers.

```
# nmcli con mod static2 ipv4.dns “8.8.8.8 8.8.4.4”
```

Hint: There is something here you must pay attention: the properties for IP address and gateway have different names when you add and when you modify a connection. When you add connections you use “ip4” and “gw4”, while when you modify them you use “ipv4” and “gwv4”.

Now let’s bring up this connection profile:

```
# nmcli con down static1 ; nmcli con up static2
```

As you can see, the device enp0s3 has now IP address 192.168.1.50.

```
# ip a
```

Verify IP Address of New Network Connection

Verify IP Address of New Network Connection

Hint: There are a lot of properties you can modify. If you don't remember them by heart you can help yourself by typing "nmcli con show" and after that the connection name:

```
# nmcli con show static2
```

Verify IP Address of New Network Connection

Verify IP Address of New Network Connection

You can modify all these properties written in lowercase.

For example: when you bring down a connection profile, the NetworkManager searches for another connection profile and brings it up automatically. (I leave it as exercise to check it). If you don't want your connection profile to autoconnect:

```
# nmcli con mod static2 connection.autoconnect no
```

The last exercise is very useful: you made a connection profile but you want it to be used by specific users. It's good to classify your users!

We let only user stella to use this profile:

```
# nmcli con mod static2 connection.permissions stella
```

Hint: If you want to give permissions to more than one users, you must type user:user1,user2 without blank space between them:

```
# nmcli con mod static2 connection.permissions user:stella,john
```

Allow Network Connections to Users

Allow Network Connections to Users

If you login as another user you can't bring "up" this connection profile:

```
# nmcli con show
```

```
# nmcli con up static2
```

```
# ls /etc/sysconfig/network-scripts
```

Enable Network Connection

Enable Network Connection

An error message says that connection 'static2' does not exist, even if we see that it exists. That's because current user has no permissions to bring up this connection.

## Troubleshooting

- 1. netstat or ss ----- to show socket statistics**
  - 2. ip addr ----- to see ipaddress**
  - 3. traceroute <hostname> or tracepath----- to check the route for package transfer**
  - 4. ping ip address----- to check whether m/c is reachable**
  - 5. ip -s link show eth0 ----- to check drop packages, received packages**
- “\*” Indicates listening to ip4 and :: - listening to IPV6**