Manage network connections from the Linux command line with nmcli

Image by:
Image by Mapbox Uncharted ERG, CC-BY 3.0 US

The nmcli command lets you tap into the power of the NetworkManager tool directly from the Linux command line. It's an integral part of the NetworkManager package that makes use of an application programmer's interface (API) to access NetworkManager's functionality.

nmcli was released in 2010 and replaces other modes of configuring network interfaces and connections, such as ifconfig. Because it is a command-line interface (CLI) tool designed to be used in terminal windows and scripts, it is ideal for system administrators working on systems without a graphical user interface (GUI).

ncmli syntax

The nmcli command accepts *options* that modify nmcli's behavior, *sections* that tell nmcli which of its capabilities you want to use, and *actions* that tell it what you want it to do:

```
$ nmcli <options> <section> <action>
```

There are eight sections, each related to a specific set of network actions:

- **Help** provides help about ncmcli's commands and usage.
- **General** retrieves NetworkManager's status and global configuration.
- **Networking** provides commands to query a network connection's status and enable or disable connections.
- **Radio** provides commands to query a WiFi network connection's status and enable or disable connections.
- **Monitor** provides commands to monitor NetworkManager activity and observe network connections' status changes.
- **Connection** provides commands to bring network interfaces up and down, to add new connections, and to delete existing connections.
- **Device** is mainly used to modify parameters associated with a device (e.g., the interface name) or to connect a device using an existing connection.
- **Secret** registers nmcli as a NetworkManager secret agent listening for secret messages. This is very rarely required because nmcli does this automatically when connecting to networks.

Simple examples

As a first check, verify NetworkManager is running and nmcli can communicate with it:

```
$ nmcli general
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN
connected  full          enabled  enabled  enabled  enabled
```

Reconnaissance is often the first part of administering a system. To list all in-memory and on-disk network connection profiles:

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ac3241e4-b424-35d6-aaa7-07498561688d  ethernet  enp0s3
Wired connection 2  2279d917-fa02-390c-8603-3083ec5a1d3e  ethernet  enp0s8
Wired connection 3  52d89737-de92-35ec-b082-8cf2e5ac36e6  ethernet  enp0s9
```

This command uses the `show` action from the `connection` section.

The test machine used for this example is running Ubuntu 20.04. It has three network adaptors installed: `enp0s3`, `enp0s8`, and `enp0s9`.

Connection management

It's important to understand nmcli's nomenclature. A network **connection** is something that holds all the information about a connection. You can think of it as a network **configuration**. A connection encapsulates all the information related to a connection, including the data-link layer and the IP-addressing information. That's layer 2 and layer 3 in the OSI networking model.

When you are configuring networking on Linux, you're usually configuring connections that will eventually bind to networking devices, which are the network interfaces installed in a computer. When a connection is used by a device, the connection is said to be **active** or **up**. The opposite of active is **inactive** or **down**.

Adding network connections

The ncmli command allows you to quickly create network connections and specify elements of their configuration at the same time. To add a new connection using wired connection 2, `enp0s8`, you need to use `sudo`:

```
$ sudo nmcli connection add type ethernet ifname enp0s8
Connection 'ethernet-enp0s8' (09d26960-25a0-440f-8b20-c684d7adc2f5) successfully
added.
```

The `type` option requests an Ethernet connection, and the `ifname` (interface name) option specifies the network interface device you want the connection to use.

Check what happened:

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ac3241e4-b424-35d6-aaa7-07498561688d  ethernet  enp0s3
Wired connection 2  2279d917-fa02-390c-8603-3083ec5a1d3e  ethernet  enp0s8
Wired connection 3  52d89737-de92-35ec-b082-8cf2e5ac36e6  ethernet  enp0s9
ethernet-enp0s8     09d26960-25a0-440f-8b20-c684d7adc2f5  ethernet  --
```

Your new connection, `ethernet-enp0s8`, was created. Its universally unique identifier (UUID) was assigned, and the connection type is Ethernet. Make it active with the `up` command followed by the connection name (or the UUID):

```
$ nmcli connection up ethernet-enp0s8
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
```

Check your active connections once more:

```
$ nmcli connection show --active
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  ac3241e4-b424-35d6-aaa7-07498561688d  ethernet  enp0s3
ethernet-enp0s8     09d26960-25a0-440f-8b20-c684d7adc2f5  ethernet  enp0s8
Wired connection 3  52d89737-de92-35ec-b082-8cf2e5ac36e6  ethernet  enp0s9
```

Your new connection, `ethernet-enp0s8`, is now active and bound to the `enp0s8` network interface device.

Adjusting connections

The ncmli command makes it easy to adjust existing connections' parameters. Perhaps you want to switch one network interface from Dynamic Host Configuration Protocol (DHCP) to a static IP address.

Suppose you need a fixed IP address of `192.168.4.26` for your new connection. To achieve that, you need to issue two commands. One to set the IP address, and one to set the connection's method of obtaining an IP address to `manual`:

```
$ nmcli connection modify ethernet-enp0s8 ipv4.address 192.168.4.26/24
$ nmcli connection modify ethernet-enp0s8 ipv4.method manual
```

Remember to specify the subnet mask. On this test network, it is `255.255.255.0`, or `/24` in Classless Inter-Domain Routing (CIDR).

For your changes to take effect, you need to *bounce* the connection by stopping it and bringing it back up again. The first command takes the connection down and the second brings it back up:

```
$ nmcli connection down ethernet-enp0s8
Connection 'ethernet-enp0s8' successfully deactivated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/4)
$ nmcli connection up ethernet-enp0s8
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/6)
```

If you want to set the connection to use DHCP, use `auto` instead of `manual`:

```
$ nmcli connection modify ethernet-enp0s8 ipv4.method auto
```

Device management

The commands in the `device` section of the nmcli command allow you to manage the network interfaces installed on your computer.

Checking device status

To quickly check the status of all the network interfaces:

```
$ nmcli device status
DEVICE  TYPE      STATE      CONNECTION
enp0s3  ethernet  connected  Wired connection 1
enp0s8  ethernet  connected  ethernet-enp0s8
enp0s9  ethernet  connected  Wired connection 3
lo      loopback  unmanaged  --
```

Showing device details

To examine the details of a network interface, use the `show` action from the `device` section. If you do not provide a device name, the details of all devices are retrieved and displayed. You can scroll and page up and down to review them.

Take a look at `enp0s8`, the device your new connection is using. Verify that the IP address in use is the address that you previously requested:

```
$ nmcli device show enp0s8
GENERAL.DEVICE:                         enp0s8
GENERAL.TYPE:                           ethernet
GENERAL.HWADDR:                         08:00:27:81:16:20
GENERAL.MTU:                            1500
GENERAL.STATE:                          100 (connected)
GENERAL.CONNECTION:                     ethernet-enp0s8
GENERAL.CON-PATH:
/org/freedesktop/NetworkManager/ActiveConnection/6
WIRED-PROPERTIES.CARRIER:               on
IP4.ADDRESS[1]:                         192.168.4.26/24
IP4.GATEWAY:                            --
IP4.ROUTE[1]:                           dst = 192.168.4.0/24, nh = 0.0.0.0, mt =
103
IP6.ADDRESS[1]:                         fe80::6d70:90de:cb83:4491/64
IP6.GATEWAY:                            --
IP6.ROUTE[1]:                           dst = fe80::/64, nh = ::, mt = 103
IP6.ROUTE[2]:                           dst = ff00::/8, nh = ::, mt = 256,
table=255
```

The response is quite detailed. Among other things, it shows:

- The **network interface name**, which in this case is `enp0s8`, which is assigned to it by udev.

- The **network connection type**, which in this case is a physical Ethernet connection.
- The device's **media access control (MAC) address**, which identifies the device on the network.
- The maximum transmission unit, which is the size of the largest protocol data unit that can be transmitted in a single transaction. Anything larger than this is split into several packets.
- This device is **currently connected**.
- The **name of the connection** using this device is `ethernet-enp0s8`.
- The **IP address of the connection** using this device. As requested, it is set to `192.168.4.26/24`.

The other information relates to the default routing and gateway settings that were applied to this connection, according to the network it is connected to.

nmcli's interactive editor

Although it is a command-line tool, nmcli includes an elementary interactive editor. The `edit` action will open the interactive editor on the connection you specify:

```
$ nmcli connection edit ethernet-enp0s8
```

It displays a small amount of help text, then the nmcli command prompt:

```
===| nmcli interactive connection editor |===

Editing existing '802-3-ethernet' connection: 'ethernet-enp0s8'

Type 'help' or '?' for available commands.
Type 'print' to show all the connection properties.
Type 'describe [<setting>.<prop>]' for detailed property description.

You may edit the following settings: connection, 802-3-ethernet (ethernet), 802-
1x, dcb, sriov, ethtool, match, ipv4, ipv6, tc, proxy
nmcli>
```

If you type `print` and hit **Enter**, nmcli will list all the properties associated with the connection. There are many properties. You can scroll up and down through the list:

```
===============================================================================
                Connection profile details (ethernet-enp0s8)
===============================================================================
connection.id:                          ethernet-enp0s8
connection.uuid:                        09d26960-25a0-440f-8b20-c684d7adc2f5
connection.stable-id:                   --
connection.type:                        802-3-ethernet
connection.interface-name:              enp0s8
connection.autoconnect:                 yes
connection.autoconnect-priority:        0
connection.autoconnect-retries:         -1 (default)
connection.multi-connect:               0 (default)
connection.auth-retries:                -1
connection.timestamp:                   1593967212
connection.read-only:                   no
connection.permissions:                 --
connection.zone:                        --
connection.master:                      --
connection.slave-type:                  --
connection.autoconnect-slaves:          -1 (default)
connection.secondaries:                 --
```

Change your connection back to use DHCP. Type `goto ipv4` and hit **Enter**:

```
nmcli> goto ipv4
You may edit the following properties: method, dns, dns-search, dns-options, dns-
priority, addresses, gateway, routes, route-metric, route-table, routing-rules,
ignore-auto-routes, ignore-auto-dns, dhcp-client-id, dhcp-iaid, dhcp-timeout,
dhcp-send-hostname, dhcp-hostname, dhcp-fqdn, dhcp-hostname-flags, never-default,
may-fail, dad-timeout
nmcli ipv4>
```

The property you want to change is `method`. Type `set method auto` and hit **Enter**:

```
nmcli ipv4> set method auto
Do you also want to clear 'ipv4.addresses'? [yes]:
```

If you want the connection to purge the static IP address, press **Enter**. To keep it, type `no` and hit **Enter**. You can keep it if you think you might use it again in the future. Even with a stored static IP address, if `method` is set to `auto`, it will use DHCP.

Type `save` to save your changes:

```
nmcli ipv4> save
Connection 'ethernet-enp0s8' (09d26960-25a0-440f-8b20-c684d7adc2f5) successfully
updated.
nmcli ipv4>
```

Type `quit` to exit the nmcli interactive editor. If you don't want to quit, type `back` to go back to the main level, and carry on using the editor.

There's much more to nmcli

Browse around the interactive editor and see just how many settings there are and how many properties each setting has. The interactive editor is a neat tool, but for nifty one-liners or to use nmcli in scripts, you'll need the regular command-line version.

Now that you have the basics in hand, check out the nmcli man page to see what else it can offer.

What to read next
