Understanding nmcli

- nmcli is a command-line tool for controlling NetworkManager and reporting network status.
- It can be utilised as a replacement for nm-applet or other graphical clients. nmcli is used to create, display, edit, delete, activate, and deactivate network connections, as well as control and display network device status.
- Connections are stored in configuration files
- The NetworkManager service must be running to manage these files

ALSO READ:

16 ip command examples to configure network interfaces (cheatsheet)

Compare nm-settings with ifcfg-* directives (IPv4)

nmcli con mod	ifcfg-* file	Effect
`ipv4.method manual`	`BOOTPROTO=none`	IPv4 address configured statically
`ipv4.method auto`	`BOOTPROTO=dhcp`	Will look for configuration settings from a DHCPv4 server
`ipv4.address "192.168.0.10/24"`	`IPADDR=192.168.0.10` `PREFIX=24`	Set static IPv4 address, network prefix
`ipv4.gateway 192.168.0.1`	`GATEWAY=192.168.0.1`	Set IPv4 Gateway
`ipv4.dns 8.8.8.8`	`DNS1=8.8.8.8`	Modify **/etc/resolv.conf** to use this **nameserver**
`ipv4.dns-search example.com`	`DOMAIN=example.com`	Modify /etc/resolv.conf to use this domain in the search directive
`ipv4.ignore-auto-dns true`	`PEERDNS=no`	Ignore DNS Server information from the DHCP Server
`connection.autoconnect yes`	`ONBOOT=yes`	Automatically activate this connection on boot
`connection.id eth0`	`NAME=eth0`	The name of this connection
`connection.interface-name eth0`	`DEVICE=eth0`	The connection is bound to the network interface with this name

nmcli con mod	ifcfg-* file	Effect
`802-3-ethernet.mac-address 08:00:27:4b:7a:80`	`HWADDR=08:00:27:4b:7a:80`	The connection is bound to the network interface with this MAC Address
`ipv4.never-default no`	`DEFROUTE=yes`	Never use provided interface's gateway as default gateway

Compare nm-settings with ifcfg-* directives (IPv6)

nmcli con mod	ifcfg-* file	Effect
`ipv6.method manual`	`IPV6_AUTOCONF=no`	IPv6 is configured statically
`ipv6.method auto`	`IPV6_AUTOCONF=yes`	Will configure network settings using SLAAC from router advertisements.
`ipv6.method dhcp`	`IPV6_AUTOCONF=no` `DHCPV6C=yes`	Will configure network settings by using DHCPv6, but not SLAAC
`ipv6 . addresses "2001:db8::a/64 2001:db8::1"`	`IPV6ADDR=2001:db8::a/64` `IPV6_DEFAULTGW=2001:db8::1`	Sets static IPv6 Address and Gateway
`ipv6.dns . . .`	`DNS0=. . .`	Modify **/etc/resolv.conf** to use this **nameserver**
`ipv6.dns-search example.com`	`DOMAIN=example.com`	Modify **/etc/resolv.conf** to use to use this domain in the **search** directive
`ipv6.ignore-auto-dns true`	`IPV6_PEERDNS=no`	Ignore DNS server information from the DHCP server
`connection.autoconnect yes`	`ONBOOT=YES`	Automatically activates the connection at boot
`connection.id eth0`	`NAME=eth0`	The name of this connection
`connection.interface-name eth0`	`DEVICE=eth0`	The connection is bound to this network interface with this name
`802-3-ethernet.mac-address . . .`	`HWADDR=. . .`	The connection is bound to the network interface with this MAC Address

Brief list of nmcli commands syntax

Command	Purpose
`nmcli dev status`	Show the Network Manager status of all network interfaces
`nmcli con show`	List all connections
`nmcli con show name`	List the current settings for the connection name
`nmcli con add con-name name ..`	Add a new connection named name
`nmcli con mod name ..`	Modify the connection name
`nmcli con reload`	Tell networkManager to reread the configuration files (useful after they have been edited by hand)
`nmcli con up name`	Activate the connection name
`nmcli dev dis dev`	Deactivate and disconnect the current connection on the network interface dev
`nmcli con del name`	Delete the connection name and its configuration file

nmcli command examples (cheatsheet)

Below are some of the chosen nmcli command examples

1. Check if NetworkManager is running

You can use below command to check if NetworkManager is running or not

```
# nmcli -t -f RUNNING general

running
```

To get a general status

```
# nmcli general
```

```
STATE      CONNECTIVITY  WIFI-HW  WIFI     WWAN-HW  WWAN

connected  full          enabled  enabled  enabled  enabled
```

2. List all the available device

To view and list all the available devices on your Linux system

```
# nmcli dev status

DEVICE      TYPE      STATE         CONNECTION

eth0        ethernet  connected     eth0

virbr0      bridge    disconnected  --

eth1        ethernet  disconnected  --

eth2        ethernet  disconnected  --

lo          loopback  unmanaged     --

virbr0-nic  tun       unmanaged     --
```

ALSO READ: How to create or configure NIC Teaming using nmcli (CentOS / RHEL 7/8)

3. List all the available connections

To list all the available connections

```
# nmcli con show

NAME  UUID                                  TYPE      DEVICE

eth1  01fa0bf4-b6bd-484f-a9a3-2b10ff701dcd  ethernet  eth1

eth0  2e9f0cdd-ea2f-4b63-b146-3b9a897c9e45  ethernet  eth0
```

```
eth2  186053d4-9369-4a4e-87b8-d1f9a419f985  ethernet  eth2
```

4. List all the configuration of interface

To view all the configured values (default and custom) of an interface

```
# nmcli con show eth2

connection.id:                          eth2

connection.uuid:                        186053d4-9369-4a4e-87b8-d1f9a419f985

connection.stable-id:                   --

connection.type:                        802-3-ethernet

connection.interface-name:              eth2

connection.autoconnect:                 yes



<Output trimmed>
```

5. Check physical network device status

Now the status of all the connection network devices

```
# nmcli dev status

DEVICE  TYPE      STATE         CONNECTION

eth1    ethernet  connected     eth1

eth0    ethernet  connected     eth0
```

```
eth2    ethernet  disconnected  --

lo      loopback  unmanaged     --
```

6. Change hostname using nmcli

You can ideally change hostname using `hostnamectl` command, but you can also update hostname using `nmcli`

To get the current hostname

```
# nmcli general hostname

centos-8.example.com
```

Next to update the hostname

```
# nmcli general hostname  centos-8.golinuxcloud.com
```

Verify the same

```
# nmcli general hostname

centos-8.golinuxcloud.com



# hostname

centos-8.golinuxcloud.com
```

7. Create a new ethernet connection and assign static IP Address

In this example `nmcli` configures the eth2 interface statically, using the IPv4 address and network prefix `10.10.10.4/24` and default gateway `10.10.10.1`, but still auto connects at startup and saves its configuration into `/etc/sysconfig/network-scripts/ifcfg-eth2` file.

```
# nmcli con add con-name eth2 type ethernet ifname eth2 ipv4.method manual
ipv4.address 10.10.10.4/24 ipv4.gateway 10.10.10.1

Connection 'eth2' (460b16aa-e755-403e-b0ec-5e1560dcc441) successfully added.
```

ALSO READ: Understanding nova compute architecture basics in Openstack (flow chart)

8. Create a new ethernet connection and assign DHCP IP Address

The following command will add a new connection for the interface eth2, which will get IPv4 networking information using DHCP and will autoconnect on startup. The configuration will be saved in `/etc/sysconfig/network-scripts/ifcfg-eth2` because the `con-name` is eth2

```
# nmcli con add con-name eth2 type ethernet ifname eth2 ipv4.method auto

Connection 'eth2' (d75cb87f-cd15-40a2-9c33-138e69a06a1f) successfully added.
```

We can verify the same in the mapped interface configuration file

```
# egrep BOOTPROTO /etc/sysconfig/network-scripts/ifcfg-eth2

BOOTPROTO=dhcp
```

9. Create and configure bond connection (active-backup) with two slave interface

You can create bond connection with multiple slave interface using nmcli.

HINT:

There are 6 types of bonding mode 802.3ad/balance-alb/balance-tlb/broadcast/active-backup/balance-rr/balance-xor

Delete any configuration file which exists for slave interface

```
# nmcli con del "eth1"

# nmcli con del "Wired connection 1"
```

Add bond interface using nmcli. This command adds a master bond connection, naming the bonding interface `mybond0` and using active-backup mode. I have given some dummy values for **MII**, **UPDELAY** and **DOWNDELAY**. If you wish to add primary interface using "`primary=<ifname>`"

```
# nmcli con add type bond ifname mybond0 bond.options "mode=active-
backup,downdelay=5,miimon=100,updelay=10"

Connection 'bond-mybond0' (a5c76dbe-550b-4abf-8dc0-88184ade369e) successfully
added.
```

Similarly for **round-robin bonding** you can use bond.options as "`downdelay=5,miimon=100,mode=balance-rr,updelay=10`"

Next add the slaves for `mybond0` using nmcli. This command binds first slave to eth1 interface

```
# nmcli con add type ethernet ifname eth1 master mybond0

Connection 'bond-slave-eth1' (54dc4282-b90b-4469-9cbf-82bce042de85) successfully
added.
```

This command binds slave 2 to eth2 interface

```
# nmcli con add type ethernet ifname eth2 master mybond0

Connection 'bond-slave-eth2' (41a5b4a6-8e6b-4df9-bff2-b67c5328311a) successfully
added.
```

List the active connections. So we have our bond and slave interface with us.

```
# nmcli con show

NAME             UUID                                  TYPE      DEVICE

bond-mybond0     25ce17b2-fffb-4bf1-a5a3-e7593299f303  bond      mybond0

bond-slave-eth1  54dc4282-b90b-4469-9cbf-82bce042de85  ethernet  eth1

bond-slave-eth2  41a5b4a6-8e6b-4df9-bff2-b67c5328311a  ethernet  eth2

eth0             d05aee6a-a069-4e55-9fe4-771ca3336db6  ethernet  eth0
```

Here I am setting static IP Address, NetMask, Gateway, DNS and DNS Search to mybond0 using nmcli

```
# nmcli con mod bond-mybond0 ipv4.method manual ipv4.address 10.10.10.8/24
ipv4.gateway 10.10.10.1 ipv4.dns 8.8.8.8 ipv4.dns-search example.com
```

NOTE:

To use DHCP IP, use `ipv4.method` auto and do not provide any IP Address related details in the above command

Verify your `mybond0` configuration file

```
# egrep 'BOOTPROTO|IPADDR|PREFIX|GATEWAY|DNS' /etc/sysconfig/network-
scripts/ifcfg-bond-mybond0

BOOTPROTO=none

IPADDR=10.10.10.8

PREFIX=24

GATEWAY=10.10.10.1

DNS1=8.8.8.8
```

refresh/reload the network configuration change for mybond0

```
# nmcli con up bond-mybond0

Connection successfully activated (master waiting for slaves) (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/11)
```

Verify the bond IP Address

```
# ip addr show mybond0

7: mybond0: <BROADCAST,MULTICAST,MASTER,UP,LOWER_UP> mtu 1500 qdisc noqueue state
UP group default qlen 1000

    link/ether 08:00:27:0d:ca:0c brd ff:ff:ff:ff:ff:ff
```

```
    inet 10.10.10.8/24 brd 10.10.10.255 scope global noprefixroute mybond0

       valid_lft forever preferred_lft forever
```

Verify the list of available connection

```
# nmcli con show --active

NAME  UUID                                  TYPE      DEVICE

eth1  01fa0bf4-b6bd-484f-a9a3-2b10ff701dcd  ethernet  eth1

eth0  2e9f0cdd-ea2f-4b63-b146-3b9a897c9e45  ethernet  eth0
```

ALSO READ: Configure LUKS Network Bound Disk Encryption with clevis & tang server

10. Create and configure Network Bridge

I have written another article with detailed steps to create and configure network bridge using nmcli and nmtui separately on RHEL/CentOS 7 and 8 Linux.

11. Create and configure Network Teaming

I have written another article with detailed steps to create and configure NIC teaming with two slaves using nmcli validated on RHEL/CentOS 7/8 Linux

12. Reload connection using nmcli (restart)

Reload all connection files from disk. NetworkManager does not monitor changes to connection files by default. So you need to use this command in order to tell NetworkManager to re-read the connection profiles from disk when a change was made to them.

```
# nmcli con reload
```

13. Interactively add/edit a connection

You can use `nmcli` con edit to Edit an existing connection or add a new one, using an interactive editor. In the below example we will edit eth1's IP Address

```
# nmcli con edit  eth1



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



Editing existing '802-3-ethernet' connection: 'eth1'



Type 'help' or '?' for available commands.

Type 'print' to show all the connection properties.

Type 'describe [.]' 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> help

------------------------------------------------------------------------------

---[ Main menu ]---

goto     [ | ]        :: go to a setting or property

remove   [.] |  :: remove setting or reset property value

set      [. ]  :: set property value
```

```
describe [.]          :: describe property

print    [all | [.]]  :: print the connection

verify   [all | fix]                :: verify the connection

save     [persistent|temporary]     :: save the connection

activate [] [/|]    :: activate the connection

back                                :: go one level up (back)

help/?   []                 :: print this help

nmcli            :: nmcli configuration

quit                                :: exit nmcli

------------------------------------------------------------------------------

nmcli> print ipv4.address

ipv4.addresses: 10.10.10.4/24

nmcli> remove ipv4.address "10.10.10.4/24"

nmcli> print ipv4.address

ipv4.addresses:

nmcli> set ipv4.address 10.10.10.5/24

Do you also want to set 'ipv4.method' to 'manual'? [yes]: yes

nmcli> print ipv4.address
```

```
ipv4.addresses: 10.10.10.5

nmcli> verify

Verify connection: OK

nmcli> save

Connection 'eth1' (7e3a1246-1743-4bb8-9eab-09664ab996b8) successfully updated.

nmcli> quit
```

Now verify your changes in eth1's configuration file

```
# egrep IPADDR /etc/sysconfig/network-scripts/ifcfg-eth1

IPADDR=10.10.10.5
```

ALSO READ:15+ SSH command examples in Linux [Cheat Sheet]

14. Change ethernet connection BOOTPROTO from DHCP to Static

Now to change ethernet connection BOOTPROTO from DHCP to static using nmcli, we must modify `ipv4.method` directive to use manual

```
# nmcli con mod eth2 ipv4.method manual ipv4.address 10.10.10.4/24 ipv4.gateway
10.10.10.1
```

Now verify the network configuration file for `eth2`

```
# egrep 'BOOTPROTO|IPADDR|PREFIX|GATEWAY' /etc/sysconfig/network-scripts/ifcfg-
eth2

BOOTPROTO=none

IPADDR=10.10.10.4

PREFIX=24
```

```
GATEWAY=10.10.10.1
```

15. Change ethernet connection BOOTPROTO from Static to DHCP

Similarly to change ethernet connection BOOTPROTO from static to DHCP using nmcli, we must modify `ipv4.method` directive to use auto

```
# nmcli con mod eth2 ipv4.method auto
```

Now verify the eth2 network configuration file

```
# egrep 'BOOTPROTO|IPADDR|PREFIX|GATEWAY' /etc/sysconfig/network-scripts/ifcfg-eth2

BOOTPROTO=dhcp

IPADDR=10.10.10.4

PREFIX=24

GATEWAY=10.10.10.1
```

As you see we still have IPADDR and other variables from previous command but they are considered null, because you can see my DHCP has assigned `10.10.10.5` to eth2

```
# ip addr show dev eth2

4: eth2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000

    link/ether 08:00:27:a8:19:0a brd ff:ff:ff:ff:ff:ff

    inet 10.10.10.5/24 brd 10.10.10.255 scope global noprefixroute dynamic eth2

       valid_lft 1068sec preferred_lft 1068sec

    inet6 fe80::b81f:a58b:43f1:b8d3/64 scope link noprefixroute
```

```
       valid_lft forever preferred_lft forever
```

16. Change ONBOOT directive using nmcli

By default ONBOOT is yes in the interface configuration file. So to disable ONBOOT we must modify `connection.autoconnect` directive using `nmcli`

Verify the ONBOOT value before changing this directive

```
# egrep 'ONBOOT' /etc/sysconfig/network-scripts/ifcfg-eth2


ONBOOT=yes
```

Change ONBOOT directive, disable ONBOOT using nmcli

```
# nmcli con mod eth2 connection.autoconnect no
```

Re-verify the ONBOOT directive for eth2

```
# egrep 'ONBOOT' /etc/sysconfig/network-scripts/ifcfg-eth2


ONBOOT=no
```

ALSO READ: 9 commands to check if connected to internet with shell script examples

17. Change DEFROUTE directive (Never use this network for default route)

By default any gateway we add for any ethernet connection will also be considered as default gateway, to turn off this directive use `ipv4.never-default` with `nmcli`

Before we make any change verify `DEFROUTE` directive in the eth2 configuration file

```
# egrep '^DEFROUTE' /etc/sysconfig/network-scripts/ifcfg-eth2


DEFROUTE=yes
```

So by default this directive is ON, we will disable the default gateway option for eth2. To turn off this directive we must select `ipv4.never-default` as "`yes`"

```
# nmcli con mod eth2 ipv4.never-default yes
```

Next verify the DEFROUTE directive for eth2

```
# egrep '^DEFROUTE' /etc/sysconfig/network-scripts/ifcfg-eth2


DEFROUTE=no
```

18. Disable IPv6 Address for ethernet connection (IPV6INIT)

By default both IPv4 and IPv6 connection type (`IPV6INIT`) is enabled for any ethernet connection type. To only use IPv4 and disable IPv6 using `nmcli`

Verify the existing status of IPv6 connection type for eth2

```
# egrep  'IPV6INIT' /etc/sysconfig/network-scripts/ifcfg-eth2


IPV6INIT=yes
```

So this is enabled, we will disable IPv6 connection type using `ipv6.method` directive with nmcli

```
# nmcli con mod eth2 ipv6.method ignore
```

HINT:
Supported input arguments for `ipv6.method` are ignore, auto, dhcp, link-local, manual, shared. You can use the same options to enable/disable IPv4 using `ipv4.method`

Now re-verify the IPV6INIT directive from eth2 config file

```
# egrep  'IPV6INIT' /etc/sysconfig/network-scripts/ifcfg-eth2


IPV6INIT=no
```

ALSO READ:10+ cut command examples in Linux [Cheat Sheet]

19. Change “Automatically Connect” Directive

By default any ethernet connection will be allowed to automatically connect, you can modify this using

```
# nmcli con mod eth2 connection.autoconnect no
```

20. Add or Modify DNS to existing connection

You can use `ipv4.dns` to add DNS server to an new connection or modify any existing connection using `nmcli`.
Currently there are no DNS server IP provided for eth1

```
# egrep DNS /etc/sysconfig/network-scripts/ifcfg-eth1
```

Next modify connection to add DNS Server IP Address

```
# nmcli con mod eth1 ipv4.dns 8.8.8.8
```

Verify the `eth1` config file

```
[root@rhel-8 ~]# egrep DNS /etc/sysconfig/network-scripts/ifcfg-eth1


DNS1=8.8.8.8
```

21. Append single/multiple DNS Server to connection

Use `+` prefix with `ipv4.dns` to append new DNS IP Addresses to an existing connection using nmcli. In the previous example we added 8.8.8.8 as my DNS server for eth1. Now we will append 8.2.2.2 to the same connection

```
# nmcli con mod eth1 +ipv4.dns 8.2.2.2
```

Verify the eth1 configuration file

```
# egrep DNS /etc/sysconfig/network-scripts/ifcfg-eth1


DNS1=8.8.8.8


DNS2=8.2.2.2
```

HINT:
You can also append single or multiple values from other multi-value (container) properties like ipv4.dns, ipv4.addresses, bond.options, etc

22. Remove single/multiple DNS Server from connection

As it is understood, with `+` we append so with - we remove single/multiple entries of DNS Server from the interface connection using `nmcli`.

```
# nmcli con mod eth1 -ipv4.dns 8.2.2.2,8.8.8.8
```

Verify the eth1 configuration file

```
# egrep DNS /etc/sysconfig/network-scripts/ifcfg-eth1
```

HINT:
You can also remove single or multiple values from other multi-value (container) properties like ipv4.dns, ipv4.addresses, bond.options, etc

ALSO READ:10+ cut command examples in Linux [Cheat Sheet]

23. Display selected fields with values of connection

You can list all the configured values of a connection using "`nmcli con show <ifname>`" but that gives you a long list of details, you can actually also get selected value of the provided directive of an individual connection

To get the IPv4 Address of eth1

```
# nmcli -g ip4.address connection show eth1

10.10.10.4/24
```

You can use `-g` to print values from specific fields using `nmcli`

```
# nmcli -g ip4.address,ipv4.dns connection show eth1

8.8.8.8,8.2.2.2

10.10.10.4/24
```

But here as you see we do not get a field to value mapping. You can use `-f` to specify what fields (column names) should be printed using nmcli. Valid field names differ for specific commands. List available fields by providing an invalid value to the `--fields` option.

```
# nmcli -f ipv4.dns,ipv4.addresses,ipv4.gateway con show eth1

ipv4.dns:                               8.8.8.8,8.2.2.2
```

```
ipv4.addresses:                         10.10.10.4/24

ipv4.gateway:                           10.10.10.1
```

HINT:
You can choose these fields: ipv4.method,ipv4.dns,ipv4.dns-search,ipv4.dns-options,ipv4.dns-priority,ipv4.addresses,ipv4.gateway,ipv4.routes,ipv4.route-metric,ipv4.route-table,ipv4.ignore-auto-routes,ipv4.ignore-auto-dns,ipv4.dhcp-client-id,ipv4.dhcp-timeout,ipv4.dhcp-send-hostname,ipv4.dhcp-hostname,ipv4.dhcp-fqdn,ipv4.never-default,ipv4.may-fail,ipv4.dad-timeout

24. Monitor connection and device activity

Using nmcli monitor you can observe NetworkManager activity. Watches for changes in connectivity state, devices or connection profiles. Here in this example we will execute `nmcli monitor` for eth1 in one terminal, and on the other terminal we will make some modification to eth1 connection

```
# nmcli con mod eth1 ipv4.method manual ipv4.address 10.10.10.4/24
```

As you see, after the modification, the monitor command gives below output

```
# nmcli con monitor eth1

eth1: connection profile changed
```

ALSO READ: How to mount filesystem without fstab using systemd (CentOS/RHEL 7/8)

25. Activate a connection

Just opposite to what we used above, we will use `nmcli con up`

```
# nmcli con up eth2

Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/23)
```

Verify the list of available connection

```
# nmcli con show --active

NAME  UUID                                  TYPE      DEVICE
```

```
eth1  01fa0bf4-b6bd-484f-a9a3-2b10ff701dcd  ethernet  eth1

eth0  2e9f0cdd-ea2f-4b63-b146-3b9a897c9e45  ethernet  eth0

eth2  186053d4-9369-4a4e-87b8-d1f9a419f985  ethernet  eth2
```

26. De-activate a connection

Deactivate a connection from a device without preventing the device from further auto-activation using `nmcli con down <ifname>`. Multiple connections can be passed to the command.

```
# nmcli con down eth1

Connection 'eth1' successfully deactivated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/32)
```

Verify the list of active connections

```
# nmcli con show --active

NAME  UUID                                  TYPE      DEVICE

eth0  d05aee6a-a069-4e55-9fe4-771ca3336db6  ethernet  eth0
```

NOTE:
If you are connected to your server using this interface then your connection would close once the connection is de-activated

27. Delete connection

lastly in nmcli command examples, you can delete all type of available connection using "`nmcli con del <ifname>`"

```
# nmcli con del bond-mybond0 bond-slave-eth1 bond-slave-eth2

Connection 'bond-mybond0' (25ce17b2-fffb-4bf1-a5a3-e7593299f303) successfully
deleted.
```

```
Connection 'bond-slave-eth1' (54dc4282-b90b-4469-9cbf-82bce042de85) successfully
deleted.

Connection 'bond-slave-eth2' (41a5b4a6-8e6b-4df9-bff2-b67c5328311a) successfully
deleted.
```

Lastly I hope the steps from the article with nmcli command examples (cheatsheet) on Linux was helpful. So, let me know your suggestions and feedback using the comment section.

References:
man page nmcli
man page nmcli-examples