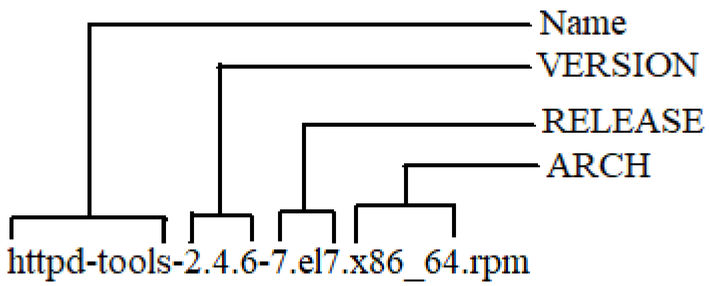
INSTALLING AND UPDATING SOFTWARE PACKAGE

RPM Software Packages and Yum

RPM package files are named using a combination of the package *name-version-release.architecture:*



- **NAME** is one or more words describing the contents (httpd-tools).
- **VERSION** is the version number of the original software (2.4.6).
- **RELEASE** is the release number of the package based on that version, and is set by the packager, who might not be the original software developer (7.el7).
- **ARCH** is the processor architecture the package was compiled to run on. "**noarch**" indicates that this package's contents are not architecture-specific (x86_64).

When installing packages from repositories, only the package name is required. The package with the higher version will be installed. If there are multiple files with the same version, the package with the higher release number will be installed.

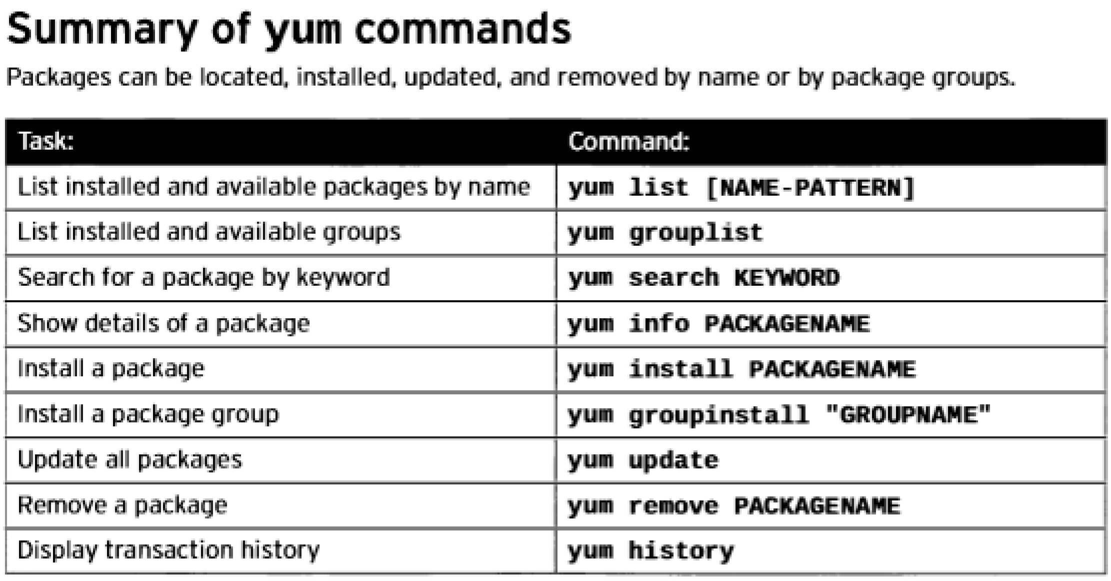
Updates and Patches

When the upstream source code for a software package is patched by Red Hat, a complete RPM package is generated. If a package is newly added to a system, only the latest version of that package is needed, not every version of the package since the first release. For systems that need updating, the old version of the package is actually removed and the new version is installed. Configuration files are usually retained during an upgrade, but the exact behavior for a particular package is defined when the new version of the package is created.

In most cases, only one version or release of a package may be installed at a time. Typically, the RPM installation process will not allow files to be overwritten. If a package is built so that there are no conflicting filenames, then multiple versions may be installed. This is the case for the **kernel** package. Since a new kernel can only be tested by booting to that kernel, the package is specifically designed so that multiple versions may be installed at once. If the new kernel fails to boot, the old kernel is still available.

The YUM package manager

The **yum** command searches numerous repositories for packages and their dependencies so they may be installed together in an effort to alleviate dependency issues. The main configuration file for **yum** is **/etc/yum.conf** with additional repository configuration files located in the **/etc/yum.repos.d** directory. Repository configuration files include, at a minimum, a repo id (in square brackets), a name and the URL location of the package repository. The URL can point to a local directory (file) or remote network share (http, ftp, etc.). If the URL is pasted in a browser, the contents should display the RPM packages, possibly in one or more subdirectories, and a **repodata** directory with information about available packages.



Summary of yum commands

Packages can be located, installed, updated, and removed by name or by package groups.

Task:	Command:
List installed and available packages by name	**yum list [NAME-PATTERN]**
List installed and available groups	**yum grouplist**
Search for a package by keyword	**yum search KEYWORD**
Show details of a package	**yum info PACKAGENAME**
Install a package	**yum install PACKAGENAME**
Install a package group	**yum groupinstall "GROUPNAME"**
Update all packages	**yum update**
Remove a package	**yum remove PACKAGENAME**
Display transaction history	**yum history**

#yum-config-manager

- manage yum configuration options and yum repositories

Syntax:

#yum-config-manager [options] [section]

Options:

--enable Enable the specified repos (automatically saves). To enable all repositories run

```
#yum-config-manager   --enable   [repo-id]
```

--disable Disable the specified repos (automatically saves). To disable all repositories run

```
#yum-config-manager    --disable    [repo-id]
```

--add-repo=ADDREPO

Add (and enable) the repo from the specified file or url.

```
#yum-config-manager    --add-repo=file:///path/to/file
#yum-config-manager    --add-repo=http://site.com/path/to/url
#yum-config-manager    --add-repo=ftp://file/transfer/path
```

Sample yum configuration file:

```
#vim   /etc/yum.repos.d/custom.repo
```

Entries in custom.repo file:

```
[custom]
name=custom repo
baseurl=http://url                    #url of the repository
baseurl=file:///path/to/file          #path of the repository
gpgcheck=0                            #for verification of package with secure key
gpgkey=file:///path/to/file           #add this line if gpgcheck=1
enabled=1                             #for enabling the repo file
                                      #enabled=0 will disable the repo
```

```
#rpm    -ivh    PACKAGEFILE.rpm
```

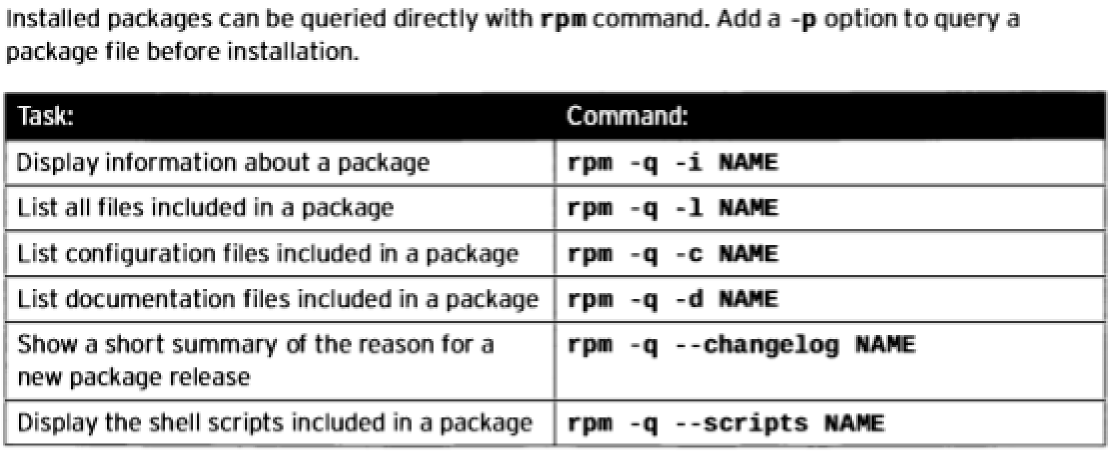
can also be used to install package files. However, using **yum** helps maintain a transaction history kept by yum (see **yum history**).

```
#rpm  --import   gpgkey-file
```

This command will import **gpgkey** which is required for verifying packages at installation time.

Examining downloaded packages with rpm

The rpm utility is a low-level tool that can get information about the contents of package files and installed packages. It gets its information from a local database or the package files themselves.



Installed packages can be queried directly with **rpm** command. Add a **-p** option to query a package file before installation.

Task:	Command:
Display information about a package	`rpm -q -i NAME`
List all files included in a package	`rpm -q -l NAME`
List configuration files included in a package	`rpm -q -c NAME`
List documentation files included in a package	`rpm -q -d NAME`
Show a short summary of the reason for a new package release	`rpm -q --changelog NAME`
Display the shell scripts included in a package	`rpm -q --scripts NAME`

A small RPM can be downloaded by this command by giving URL in its argument:

```
#wget    http://URL
```

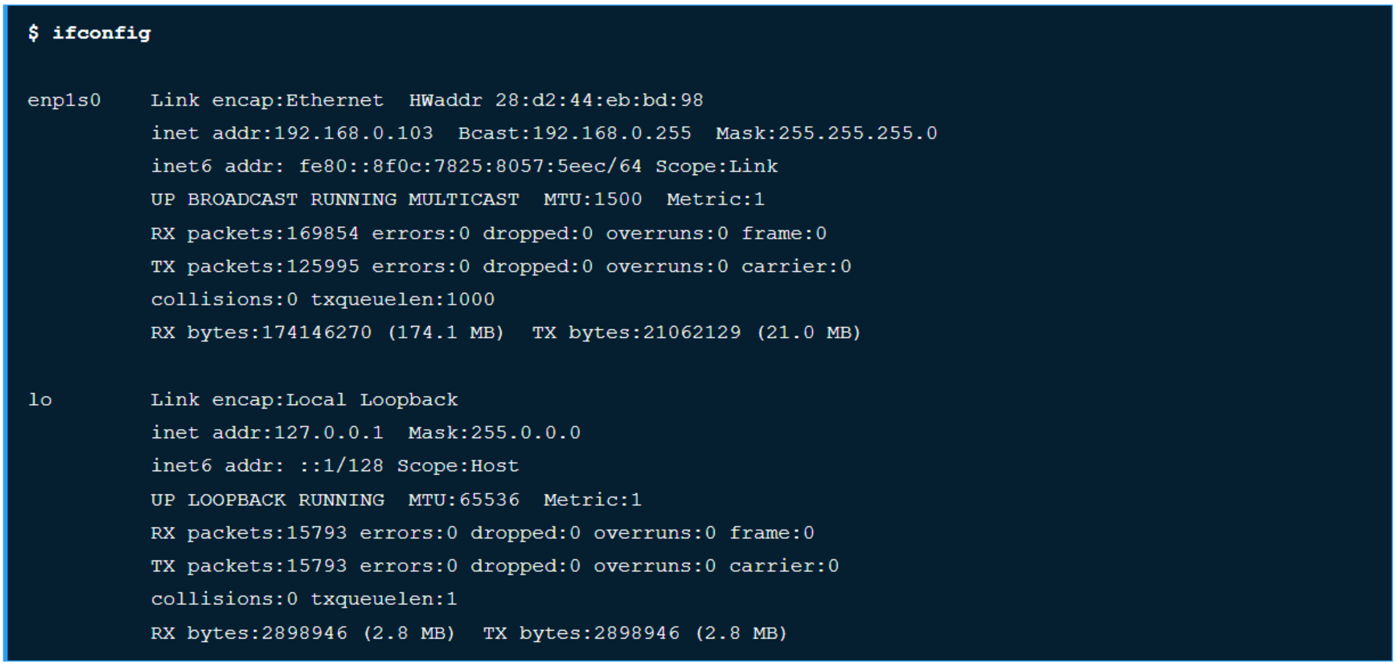
Networking:

1. **ifconfig** Command

ifconfig is a command line interface tool for network interface configuration and also used to initialize an interfaces at system boot time. Once a server is up and running, it can be used to assign an IP Address to an interface and enable or disable the interface on demand.

It is also used to view the status IP Address, Hardware / MAC address, as well as MTU (Maximum Transmission Unit) size of the currently active interfaces. **ifconfig** is thus useful for debugging or performing system tuning.

Here is an example to display status of all active network interfaces.

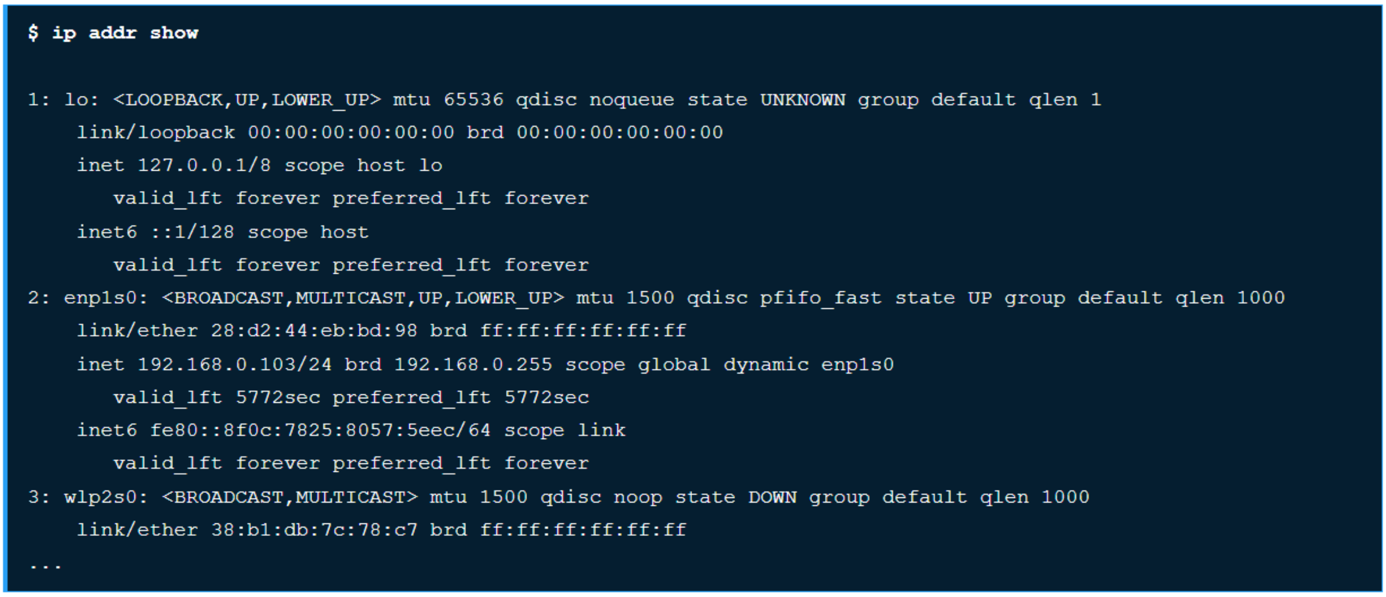


```
$ ifconfig

enp1s0    Link encap:Ethernet  HWaddr 28:d2:44:eb:bd:98
          inet addr:192.168.0.103  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::8f0c:7825:8057:5eec/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:169854 errors:0 dropped:0 overruns:0 frame:0
          TX packets:125995 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:174146270 (174.1 MB)  TX bytes:21062129 (21.0 MB)

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:65536  Metric:1
          RX packets:15793 errors:0 dropped:0 overruns:0 frame:0
          TX packets:15793 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1
          RX bytes:2898946 (2.8 MB)  TX bytes:2898946 (2.8 MB)
```

2. IP Command

ip command is another useful command line utility for displaying and manipulating routing, network devices, interfaces. It is a replacement for **ifconfig** and many other networking commands.

The following command will show the IP address and other information about a network interface.

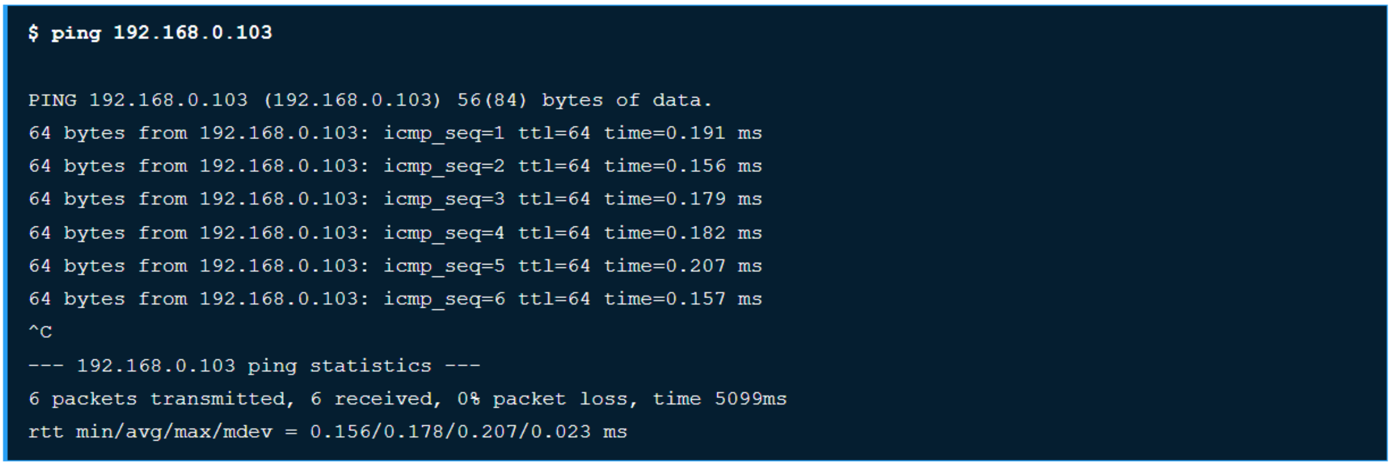


```
$ ip addr show

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default 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: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 28:d2:44:eb:bd:98 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.103/24 brd 192.168.0.255 scope global dynamic enp1s0
       valid_lft 5772sec preferred_lft 5772sec
    inet6 fe80::8f0c:7825:8057:5eec/64 scope link
       valid_lft forever preferred_lft forever
3: wlp2s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 38:b1:db:7c:78:c7 brd ff:ff:ff:ff:ff:ff
...
```

3. Ping Command

ping (Packet INternet Groper) is a utility normally used for testing connectivity between two systems on a network (Local Area Network (LAN) or Wide Area Network (WAN)). It use ICMP (Internet Control Message Protocol) to communicate to nodes on a network.

To test connectivity to another node, simply provide its IP or host name, for example.

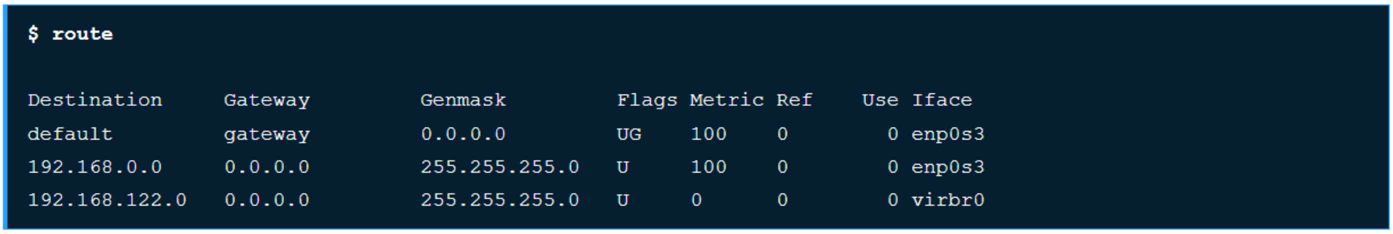


```
$ ping 192.168.0.103

PING 192.168.0.103 (192.168.0.103) 56(84) bytes of data.
64 bytes from 192.168.0.103: icmp_seq=1 ttl=64 time=0.191 ms
64 bytes from 192.168.0.103: icmp_seq=2 ttl=64 time=0.156 ms
64 bytes from 192.168.0.103: icmp_seq=3 ttl=64 time=0.179 ms
64 bytes from 192.168.0.103: icmp_seq=4 ttl=64 time=0.182 ms
64 bytes from 192.168.0.103: icmp_seq=5 ttl=64 time=0.207 ms
64 bytes from 192.168.0.103: icmp_seq=6 ttl=64 time=0.157 ms
^C
--- 192.168.0.103 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5099ms
rtt min/avg/max/mdev = 0.156/0.178/0.207/0.023 ms
```

4. Route Command

route is a command line utility for displaying or manipulating the IP routing table of a Linux system. It is mainly used to configure static routes to specific hosts or networks via an interface.

You can view Kernel IP routing table by typing.



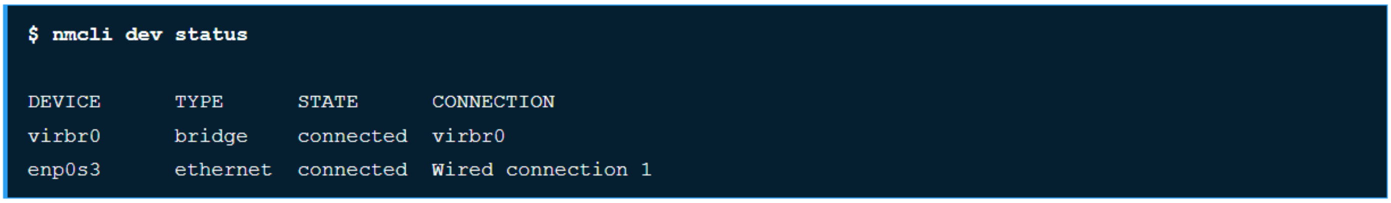
```
$ route

Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         gateway         0.0.0.0         UG    100    0        0 enp0s3
192.168.0.0     0.0.0.0         255.255.255.0   U     100    0        0 enp0s3
192.168.122.0   0.0.0.0         255.255.255.0   U     0      0        0 virbr0
```

You’ll get GATEWAY by typing: **#route -rn**

5. Nmcli Command

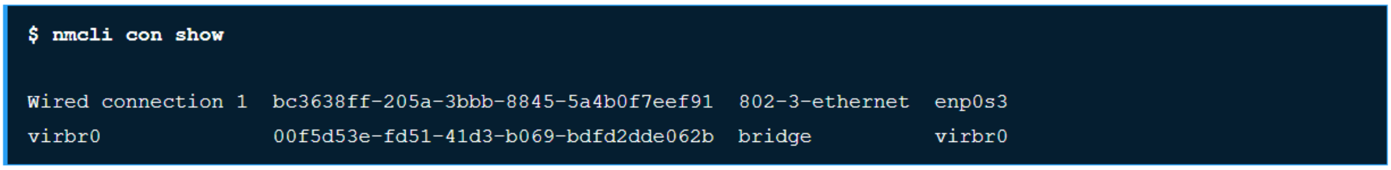
nmcli is an easy-to-use, scriptable command-line tool to report network status, manage network connections, and control the **NetworkManager**.

To view all your network devices, type.



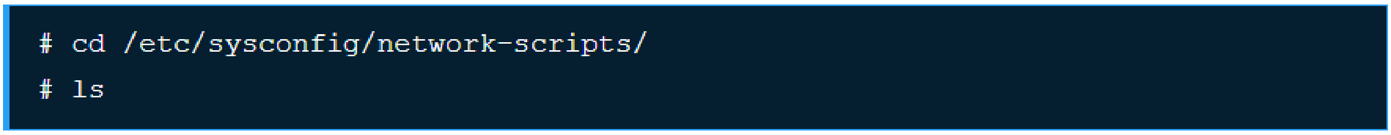
```
$ nmcli dev status

DEVICE      TYPE      STATE      CONNECTION
virbr0      bridge    connected  virbr0
enp0s3      ethernet  connected  Wired connection 1
```

To check network connections on your system, type.



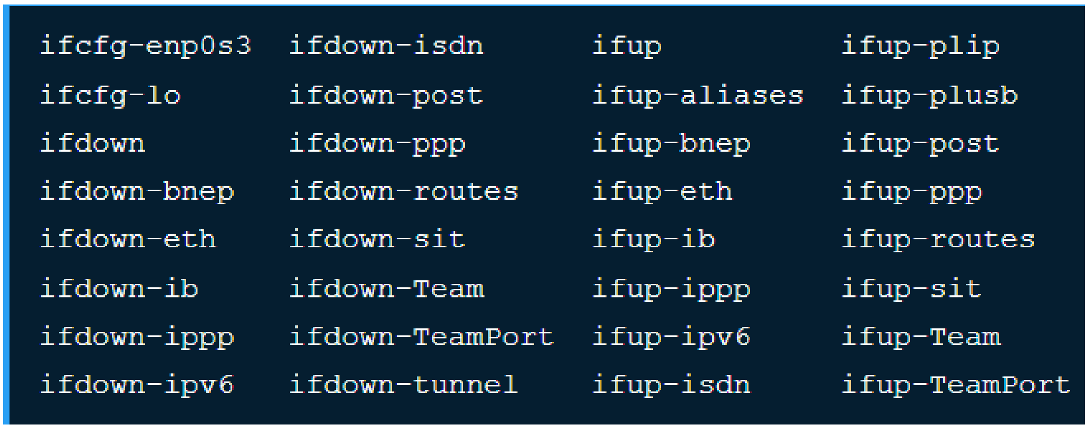
```
$ nmcli con show

Wired connection 1  bc3638ff-205a-3bbb-8845-5a4b0f7eef91  802-3-ethernet  enp0s3
virbr0              00f5d53e-fd51-41d3-b069-bdfd2dde062b  bridge          virbr0
```

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.



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

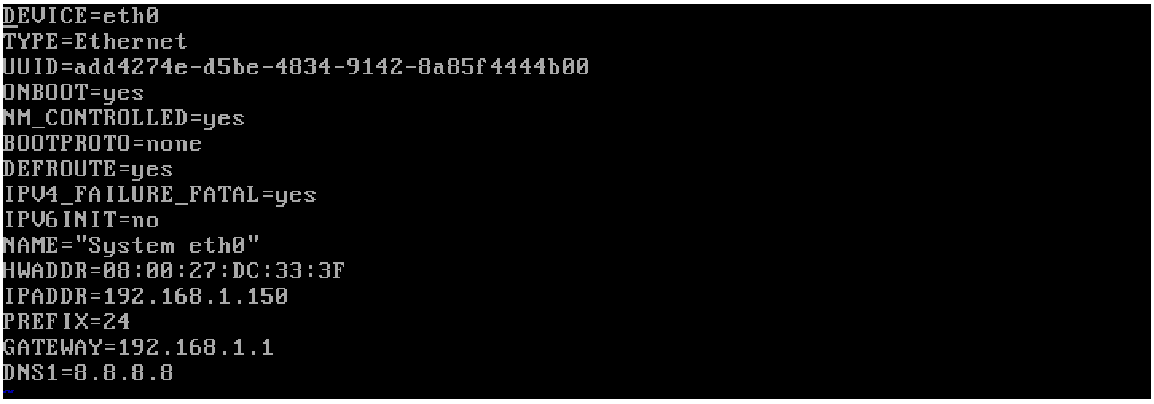
Sample Output:



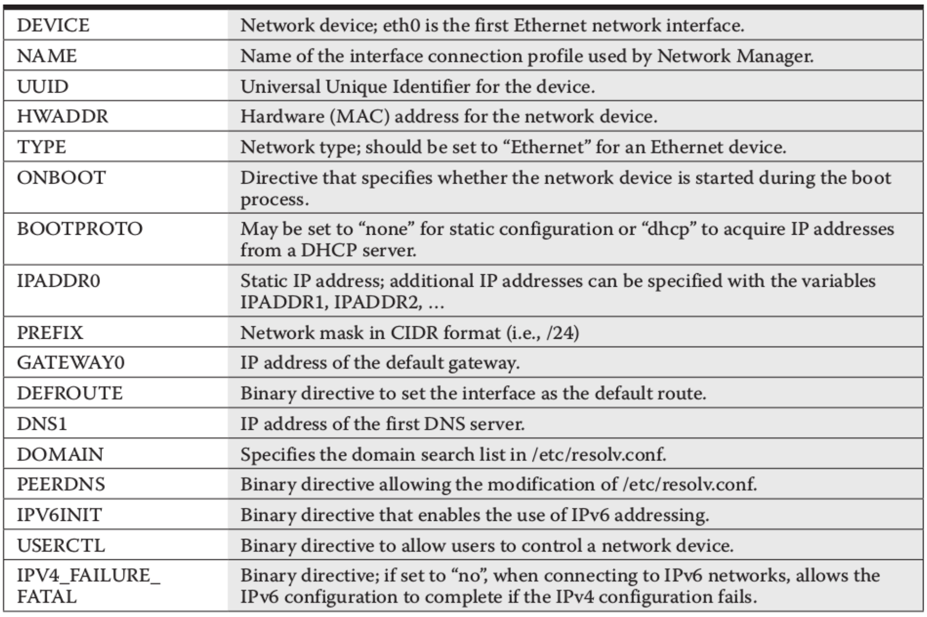
```
ifcfg-enp0s3   ifdown-isdn       ifup           ifup-plip
ifcfg-lo       ifdown-post       ifup-aliases   ifup-plusb
ifdown         ifdown-ppp        ifup-bnep      ifup-post
ifdown-bnep    ifdown-routes     ifup-eth       ifup-ppp
ifdown-eth     ifdown-sit        ifup-ib        ifup-routes
ifdown-ib      ifdown-Team       ifup-ippp      ifup-sit
ifdown-ippp    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.

A sample **ifcfg-eth0** file:



```
DEVICE=eth0
TYPE=Ethernet
UUID=add4274e-d5be-4834-9142-8a85f4444b00
ONBOOT=yes
NM_CONTROLLED=yes
BOOTPROTO=none
DEFROUTE=yes
IPV4_FAILURE_FATAL=yes
IPV6INIT=no
NAME="System eth0"
HWADDR=08:00:27:DC:33:3F
IPADDR=192.168.1.150
PREFIX=24
GATEWAY=192.168.1.1
DNS1=8.8.8.8
```



DEVICE	Network device; eth0 is the first Ethernet network interface.
NAME	Name of the interface connection profile used by Network Manager.
UUID	Universal Unique Identifier for the device.
HWADDR	Hardware (MAC) address for the network device.
TYPE	Network type; should be set to “Ethernet” for an Ethernet device.
ONBOOT	Directive that specifies whether the network device is started during the boot process.
BOOTPROTO	May be set to “none” for static configuration or “dhcp” to acquire IP addresses from a DHCP server.
IPADDR0	Static IP address; additional IP addresses can be specified with the variables IPADDR1, IPADDR2, …
PREFIX	Network mask in CIDR format (i.e., /24)
GATEWAY0	IP address of the default gateway.
DEFROUTE	Binary directive to set the interface as the default route.
DNS1	IP address of the first DNS server.
DOMAIN	Specifies the domain search list in /etc/resolv.conf.
PEERDNS	Binary directive allowing the modification of /etc/resolv.conf.
IPV6INIT	Binary directive that enables the use of IPv6 addressing.
USERCTL	Binary directive to allow users to control a network device.
IPV4_FAILURE_FATAL	Binary directive; if set to “no”, when connecting to IPv6 networks, allows the IPv6 configuration to complete if the IPv4 configuration fails.

DNS (Domain Name System or Service)

DNS is a hierarchical decentralized naming system/service that translates domain names into IP addresses on the Internet or a private network and a server that provides such a service is called a DNS server.

The **/etc/hosts** is an operating system file that translate hostnames or domain names to IP addresses. This is useful for testing websites changes or the SSL setup before taking a website publicly live.

This method will only work if the hosts have a static IP address. Therefore ensure that you have set static IP addresses *for your Linux hosts or nodes running other operating systems.*

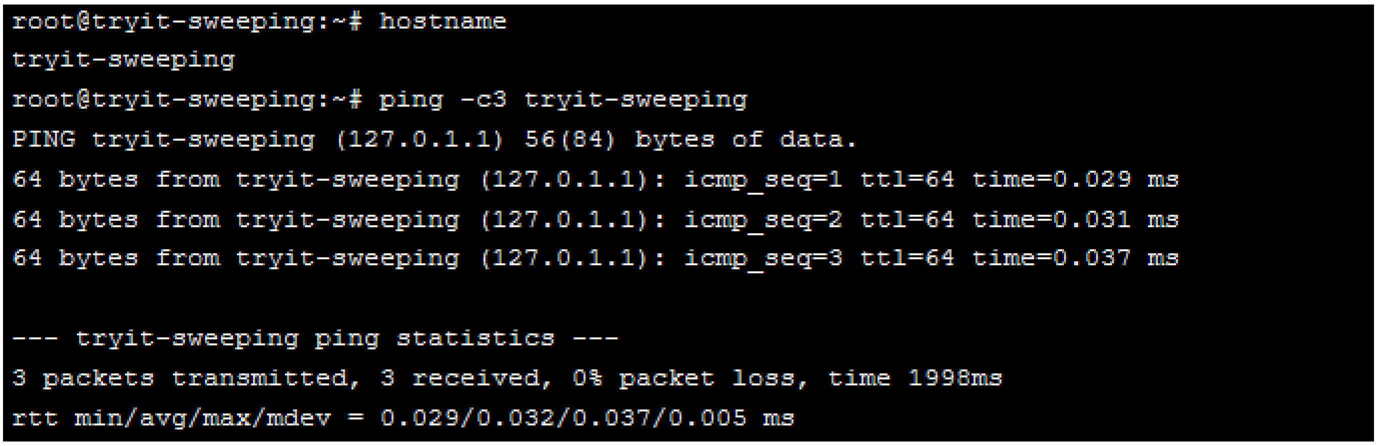
Configure DNS Locally Using /etc/hosts File in Linux

#vim /etc/hosts

Add lines (sample syntax)

```
ip-address      host-name
192.168.0.10    sample.example.com
192.168.0.1     another.example.com
```

Here is the example of ping to the localhost who’s hostname is “tryit-sweeping”, ip=127.0.0.1



```
root@tryit-sweeping:~# hostname
tryit-sweeping
root@tryit-sweeping:~# ping -c3 tryit-sweeping
PING tryit-sweeping (127.0.1.1) 56(84) bytes of data.
64 bytes from tryit-sweeping (127.0.1.1): icmp_seq=1 ttl=64 time=0.029 ms
64 bytes from tryit-sweeping (127.0.1.1): icmp_seq=2 ttl=64 time=0.031 ms
64 bytes from tryit-sweeping (127.0.1.1): icmp_seq=3 ttl=64 time=0.037 ms

--- tryit-sweeping ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 1998ms
rtt min/avg/max/mdev = 0.029/0.032/0.037/0.005 ms
```

DNS entries will be found or edited in **/etc/resolv.conf**

#cat /etc/resolv.conf

Sample output:

```
search   example.com
nameserver     192.168.0.1              #DNS
```