Date - 27/03/2021

Experiment No. 1

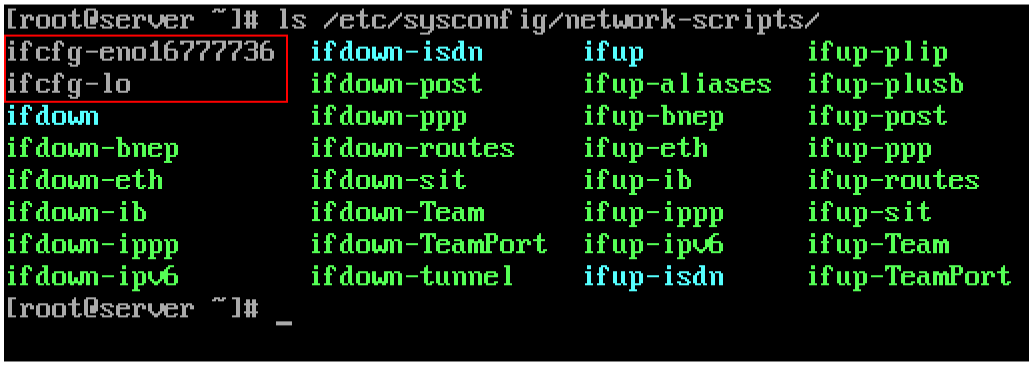
Familiarization of Network configuration files and Networking Commands in Linux.

Aim: To familiarize network configuration files and networking commands in Linux .

Theory: To store IP addresses and other related settings, Linux uses a separate configuration file for each network interface. All these Configuration files are stored in the *`/etc/sysconfig/network-scripts`* directory.

Name of configuration files starts with the **ifcfg-**. After the **ifcfg-**, to keep each file separate, the name of the interface is used. For example, if the interface name is **eno16777736** then the name of its configuration file will be the **ifcfg-eno16777736**.

The following image shows a sample output of the /etc/sysconfig/network-scripts directory.



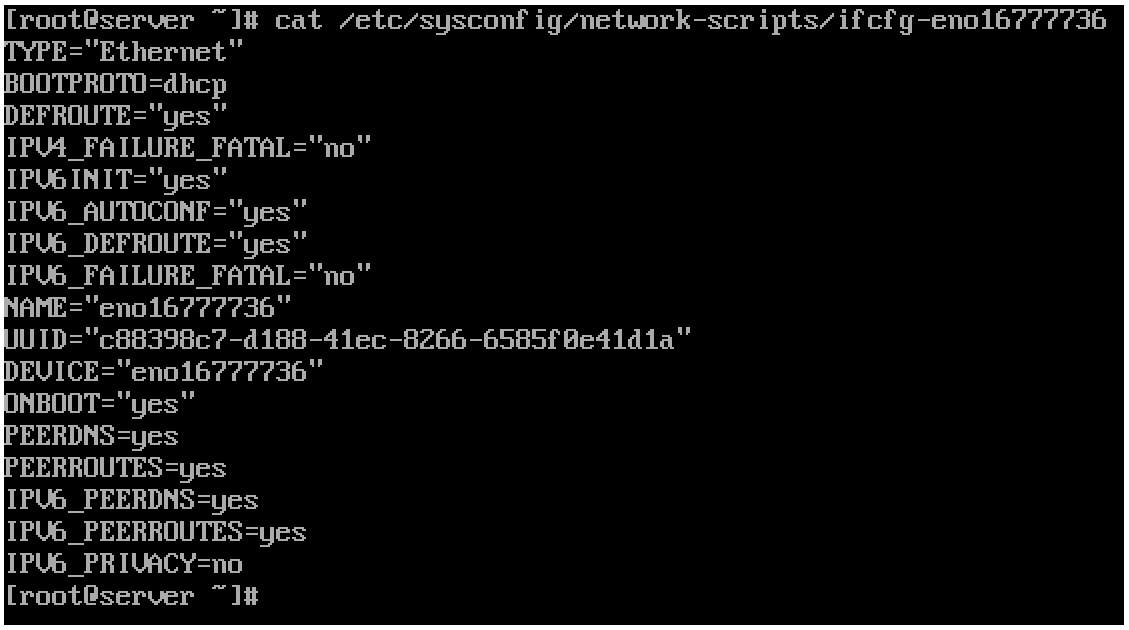
```
[root@server ~]# ls /etc/sysconfig/network-scripts/
ifcfg-eno16777736  ifdown-isdn      ifup            ifup-plip
ifcfg-lo           ifdown-post      ifup-aliases    ifup-plusb
ifdown             ifdown-ppp       ifup-bnep       ifup-post
ifdown-bnep        ifdown-routes    ifup-eth        ifup-ppp
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
[root@server ~]# _
```

The following table briefly describes the files stored in this directory.

Files	Description
ifcfg-lo	Stores the configuration of loopback device. Loopback device is a virtual network interface. It is used to test the TCP/IP protocol stocks in local host.
ifcfg-*	Stores the configuration of the network interface. Each file only stores the configuration of that interface, which it represents.
ifup-* and ifdown-*	Stores the scripts which activate and deactivate their associated protocols. For example, the **ifup-ppp** and **ifdown** files activate and deactivate the **ppp** protocol respectively.

Settings stored in the interface's configuration file are applied on the interface when we start the system or activate the interface. To view these settings, we can use the **cat** command. For example, the following command displays the settings of the **eno16777736** interface.

The following image shows how a configuration file looks like.



```
[root@server ~]# cat /etc/sysconfig/network-scripts/ifcfg-eno16777736
TYPE="Ethernet"
BOOTPROTO=dhcp
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
NAME="eno16777736"
UUID="c88398c7-d188-41ec-8266-6585f0e41d1a"
DEVICE="eno16777736"
ONBOOT="yes"
PEERDNS=yes
PEERROUTES=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_PRIVACY=no
[root@server ~]#
```

The following table lists important directives of this file with description.

Directive	Description
BOOTPROTO	Defines how the IP address should be obtained. Four values can be used here; dhcp, bootp, none and static. Use the value "**dhcp**", to obtain IP from the dhcp server. To boot from a network boot server and get IP, use the value "**bootp**". To assign IP manually, use the value "**static**". Use the value "**none**" if we don't want to assign the IP address.
BRIDGE	Specifies the name of the network bridge.
BROADCAST0	Broadcast address of the first IP configuration.
GATEWAY0	Gateway address of the first IP configuration.
DEFROUTE	Specifies whether to use this interface as the default route or not.
DEVICE	Device name of the network interface.

DNS1	IP address of the first DNS server.
HWADDR	Hardware address of the network interface.
IPADDR0	Specifies the first IP address of the interface.
IPV6INIT	Specifies whether to enable IPv6 or not.
NAME	Name of the interface. If not assigned manually, the device name will be used.
NETMASK0	Netmask or subnet mask address of the first IP configuration.
NM_CONTROLLED	Specifies whether the Network Manager service is allowed to modify the settings stored in this file or not.
ONBOOT	Whether to activate or not this interface on boot.
USERCTL	Specifies whether the non-root users are allowed to activate this interface or not.
UUID	Unique ID of this interface.
TYPE	Type of this interface.

Although we can edit configuration files directly, it is recommended that we should use available configuration tools for editing and updating these files. Using a configuration tool reduces the chances of incorrect editing in configuration files. The next parts of this tutorial explain network configuration tools in detail with practical examples.

The /etc/sysconfig/network file

This file, based on how the network interfaces are configured, may contain none, one or two configuration directives. If a system is configured to get IP configuration from the DHCP server, there will be no configuration in this file.

Two directives that may appear in this file are; GATEWAY and NETWORKING.

GATEWAY directive shows the IP address of the default gateway. This directive appears only if the same IP address is used for the default gateway in all network interfaces.

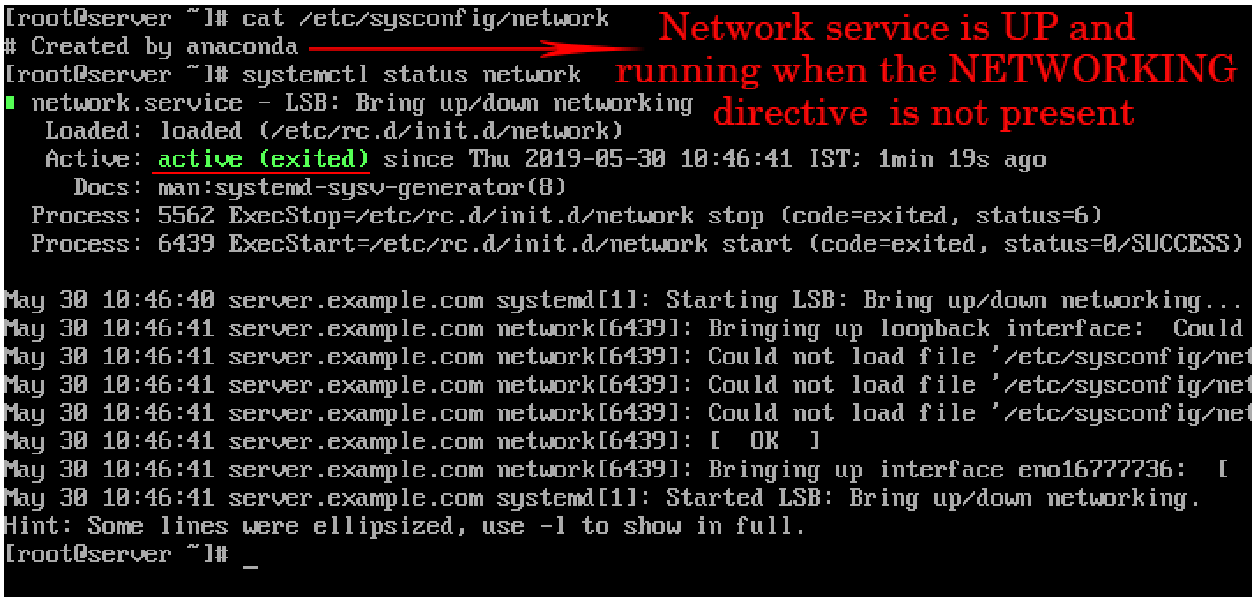
NETWORKING directive is used to control the network service. It can be configured with two values; yes and no. If the value "**no**" is used, the network service does not start.

If we run the "**ip addr show**" command and see no output or we start network service and it does not start then it means all network devices are currently inactive. In this situation, the first thing that we should check is the contents of the */etc/sysconfig/network* configuration file. This file either should not contain the **NETWORKING** directive or if it contains, its value must be set to "**yes**".

To understand how the "**NETWORKING=no**" directive affects the network service practically, follow these steps in our test system.

- Take the backup of the */etc/sysconfig/network* file.
- Delete the **NETWORKING** directive or set its value to "**yes**".
- Check the status of network service.
- Add the "**NETWORKING=no**" directive in the */etc/sysconfig/network* file
- Restart the network service.
- Remove the "**NETWORKING=no**" directive or restore the original file back
- Check the status of network service again.

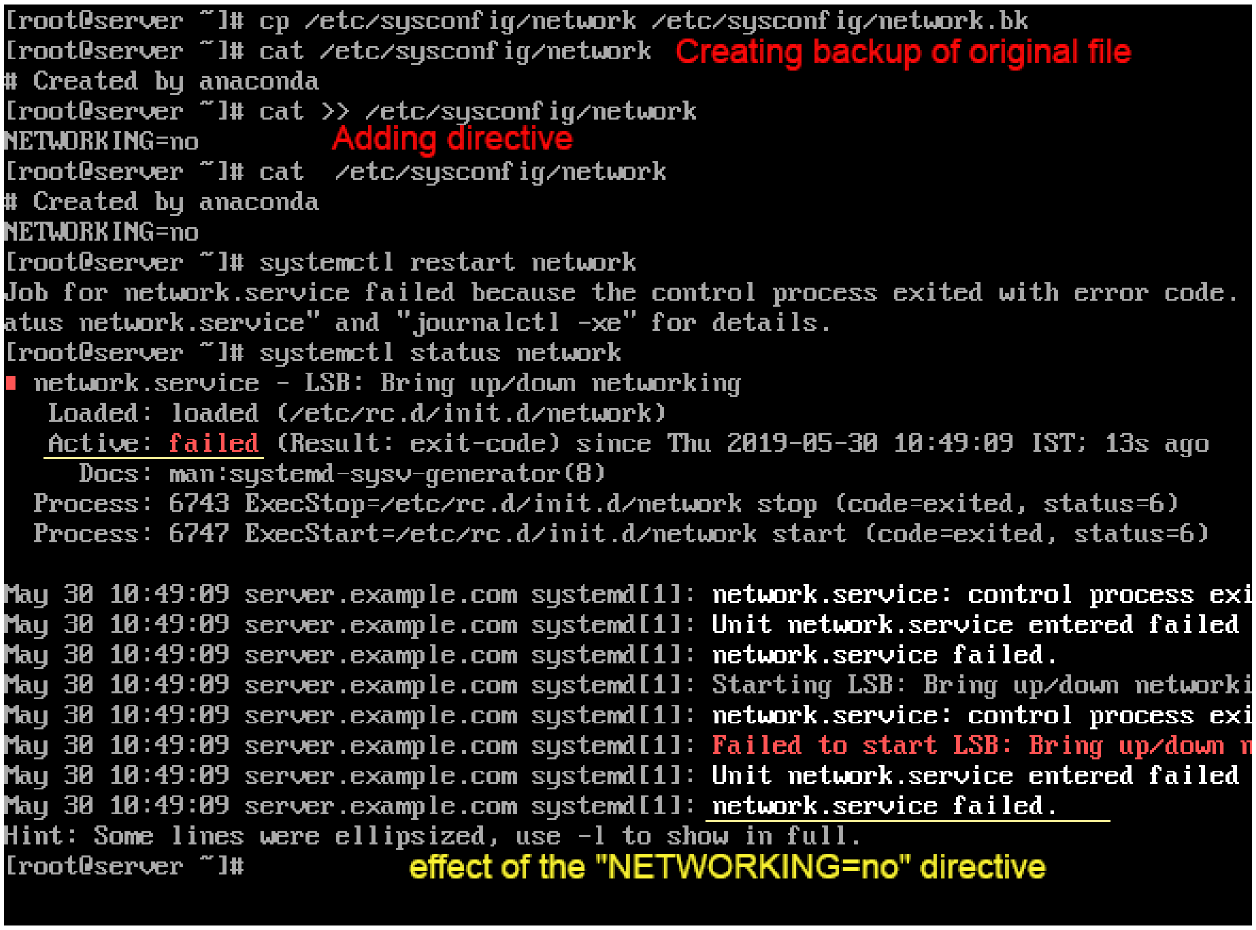
The following image shows the status of network service when the */etc/sysconfig/network* file does not contain the **NETWORKING** directive.


Network service is UP and running when the NETWORKING directive is not present

```
[root@server ~]# cat /etc/sysconfig/network
# Created by anaconda
[root@server ~]# systemctl status network
● network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network)
   Active: active (exited) since Thu 2019-05-30 10:46:41 IST; 1min 19s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 5562 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=6)
  Process: 6439 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0/SUCCESS)

May 30 10:46:40 server.example.com systemd[1]: Starting LSB: Bring up/down networking...
May 30 10:46:41 server.example.com network[6439]: Bringing up loopback interface:  Could
May 30 10:46:41 server.example.com network[6439]: Could not load file '/etc/sysconfig/net
May 30 10:46:41 server.example.com network[6439]: Could not load file '/etc/sysconfig/net
May 30 10:46:41 server.example.com network[6439]: Could not load file '/etc/sysconfig/net
May 30 10:46:41 server.example.com network[6439]: [  OK  ]
May 30 10:46:41 server.example.com network[6439]: Bringing up interface eno16777736:  [
May 30 10:46:41 server.example.com systemd[1]: Started LSB: Bring up/down networking.
Hint: Some lines were ellipsized, use -l to show in full.
[root@server ~]# _
```

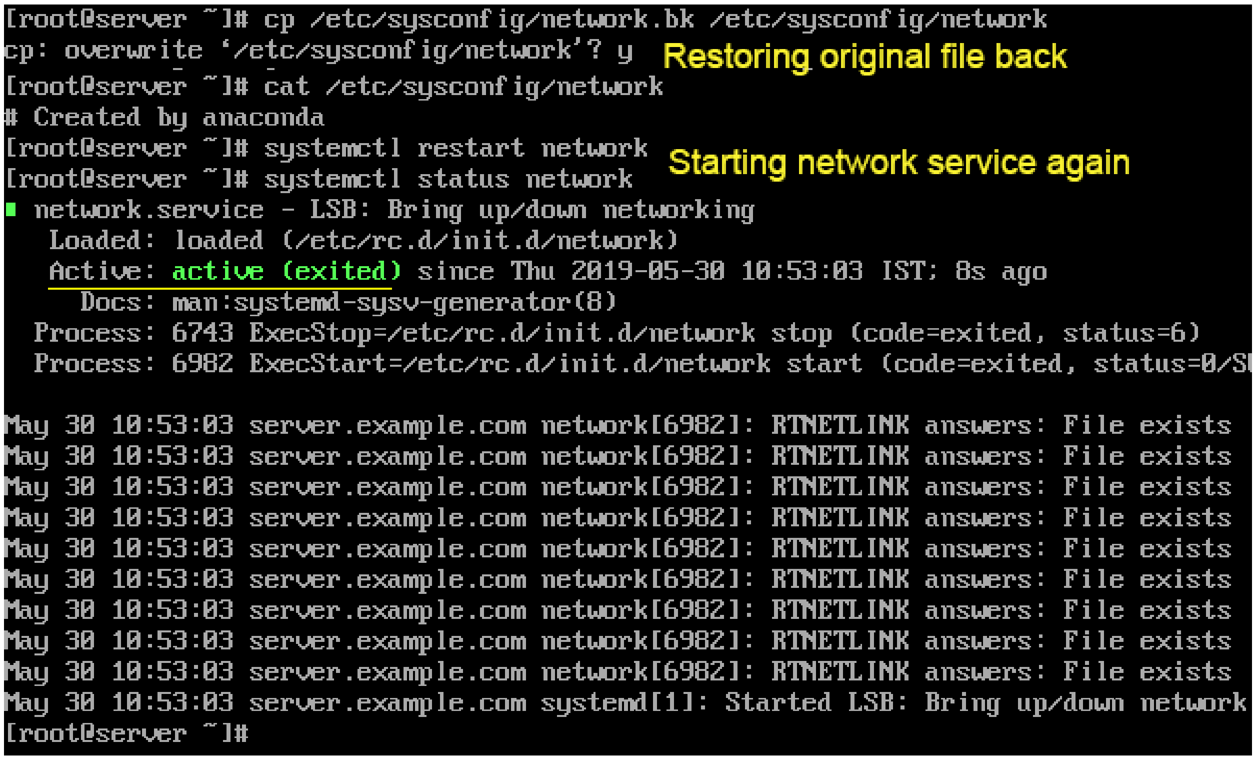
Next image shows how the "**NETWORKING=no**" directive affects the network service.



```
[root@server ~]# cp /etc/sysconfig/network /etc/sysconfig/network.bk
[root@server ~]# cat /etc/sysconfig/network  Creating backup of original file
# Created by anaconda
[root@server ~]# cat >> /etc/sysconfig/network
NETWORKING=no          Adding directive
[root@server ~]# cat  /etc/sysconfig/network
# Created by anaconda
NETWORKING=no
[root@server ~]# systemctl restart network
Job for network.service failed because the control process exited with error code.
atus network.service" and "journalctl -xe" for details.
[root@server ~]# systemctl status network
■ network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network)
   Active: failed (Result: exit-code) since Thu 2019-05-30 10:49:09 IST; 13s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 6743 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=6)
  Process: 6747 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=6)

May 30 10:49:09 server.example.com systemd[1]: network.service: control process exi
May 30 10:49:09 server.example.com systemd[1]: Unit network.service entered failed
May 30 10:49:09 server.example.com systemd[1]: network.service failed.
May 30 10:49:09 server.example.com systemd[1]: Starting LSB: Bring up/down networki
May 30 10:49:09 server.example.com systemd[1]: network.service: control process exi
May 30 10:49:09 server.example.com systemd[1]: Failed to start LSB: Bring up/down n
May 30 10:49:09 server.example.com systemd[1]: Unit network.service entered failed
May 30 10:49:09 server.example.com systemd[1]: network.service failed.
Hint: Some lines were ellipsized, use -l to show in full.
[root@server ~]#          effect of the "NETWORKING=no" directive
```

As we can see in the above image, the "**NETWORKING=no**" directive does not let the network service start. To start network service again, either we have to remove this directive or set its value to "**yes**". After this, we can start network service again. The following image shows these steps.



```
[root@server ~]# cp /etc/sysconfig/network.bk /etc/sysconfig/network
cp: overwrite '/etc/sysconfig/network'? y  Restoring original file back
[root@server ~]# cat /etc/sysconfig/network
# Created by anaconda
[root@server ~]# systemctl restart network  Starting network service again
[root@server ~]# systemctl status network
■ network.service - LSB: Bring up/down networking
   Loaded: loaded (/etc/rc.d/init.d/network)
   Active: active (exited) since Thu 2019-05-30 10:53:03 IST; 8s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 6743 ExecStop=/etc/rc.d/init.d/network stop (code=exited, status=6)
  Process: 6982 ExecStart=/etc/rc.d/init.d/network start (code=exited, status=0/SU

May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com network[6982]: RTNETLINK answers: File exists
May 30 10:53:03 server.example.com systemd[1]: Started LSB: Bring up/down network
[root@server ~]#
```

The /etc/hosts file

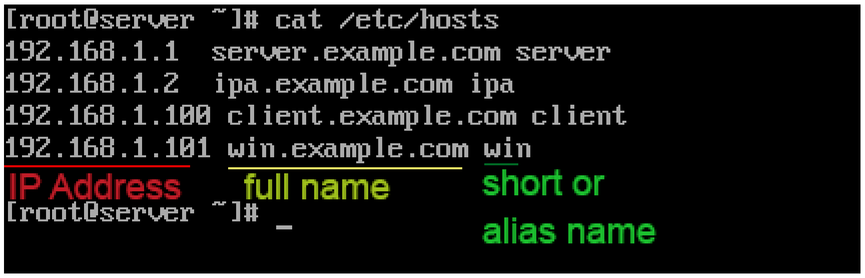
This file is used to map the hostname with IP address. Once hostname and IP address are mapped, hostname can be used to access the services available on the destination IP address. A hostname can be mapped with an IP address in two ways; through the DNS server and through the */etc/hosts* file.

DNS server provides this facility in a more dynamic and flexible way but also requires a lot of complex configuration. Due to complexity, usually DNS servers are not used in small networks.

The */etc/hosts* file also provides this functionality in the local system but requires manual mapping of all hostnames with their IP addresses.

Each row in this file represents a *unique entry*. It contains an IP address in the first column, full or official hostname in the second column and short or alias name in third column.

The following image shows an example of this file.



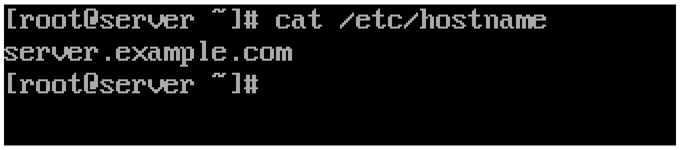
```
[root@server ~]# cat /etc/hosts
192.168.1.1   server.example.com server
192.168.1.2   ipa.example.com ipa
192.168.1.100 client.example.com client
192.168.1.101 win.example.com win
[root@server ~]# _
```

If a DNS server is configured in the network, do not update the ***/etc/hosts*** file. System uses the ***/etc/hosts*** file in the first place to resolve the IP address. If it finds no entry for a hostname in this file only then it uses the configured DNS server.

The /etc/hostname file

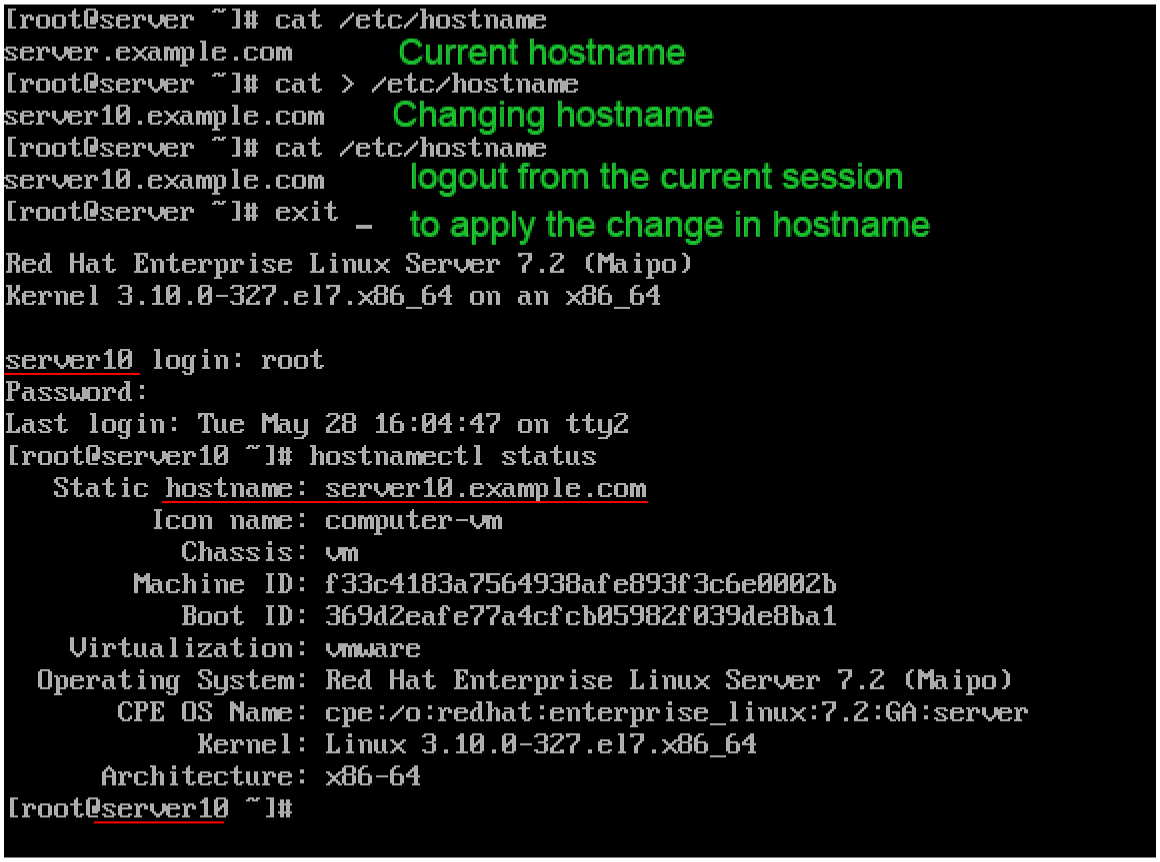
This file sets the hostname of the system. A hostname typically consists of two different parts; hostname and DNS domain name in which the host resides. These two parts together make a FQDN (Fully Qualified Domain Name) which looks like **server.example.com**

The following image shows an example of this file.



```
[root@server ~]# cat /etc/hostname
server.example.com
[root@server ~]#
```

To apply the change in hostname either restart the system or logout from the current session.



```
[root@server ~]# cat /etc/hostname
server.example.com
[root@server ~]# cat > /etc/hostname
server10.example.com
[root@server ~]# cat /etc/hostname
server10.example.com
[root@server ~]# exit _

Red Hat Enterprise Linux Server 7.2 (Maipo)
Kernel 3.10.0-327.el7.x86_64 on an x86_64

server10 login: root
Password:
Last login: Tue May 28 16:04:47 on tty2
[root@server10 ~]# hostnamectl status
   Static hostname: server10.example.com
         Icon name: computer-vm
           Chassis: vm
        Machine ID: f33c4183a7564938afe893f3c6e0002b
           Boot ID: 369d2eafe77a4cfcb05982f039de8ba1
    Virtualization: vmware
  Operating System: Red Hat Enterprise Linux Server 7.2 (Maipo)
       CPE OS Name: cpe:/o:redhat:enterprise_linux:7.2:GA:server
            Kernel: Linux 3.10.0-327.el7.x86_64
      Architecture: x86-64
[root@server10 ~]#
```

Current hostname
Changing hostname
logout from the current session
to apply the change in hostname

Networking Commands in Linux

1.	aria2	– downloading just about everything. Torrents included.
2.	arp	– view or add contents of the kernel's ARP table.
3.	arpwatch	– Ethernet Activity Monitor.
4.	bmon	– bandwidth monitor and rate estimator.
5.	bwm-ng	– live network bandwidth monitor.
6.	Curl / httpie	– transferring data with URLs.
7.	darkstat	– captures network traffic, usage statistics.
8.	dhclient	– Dynamic Host Configuration Protocol Client
9.	dig	– query DNS servers for information.
10.	dstat	– replacement for vmstat, iostat, mpstat, netstat and ifstat.
11.	ethtool	– utility for controlling network drivers and hardware.
12.	gated	– gateway routing daemon.
13.	host	– DNS lookup utility.
14.	hping	– TCP/IP packet assembler/analyzer.
15.	ibmonitor	– shows bandwidth and total data transferred.
16.	ifconfig	– display and manipulate route and network interfaces.
17.	ifplugstatus	– tells whether a cable is plugged in or not.
18.	ifstat	– report network interfaces bandwidth.
19.	iftop	– display bandwidth usage.
20.	ip (PDF file)	– a command with more features than ifconfig (net-tools).
21.	iperf3	– network bandwidth measurement tool. (above screenshot Stacklinux VPS)
22.	iproute2	– collection of utilities for controlling TCP/IP.
23.	iptables	– take control of network traffic.
24.	IPTraf	– An IP Network Monitor.
25.	iputils	– set of small useful utilities for Linux networking.
26.	iw	– a new nl80211 based CLI configuration utility for wireless devices.
27.	jwhois (whois)	– client for the whois service.
28.	“lsof -i”	– reveal information about our network sockets.
29.	mtr	– network diagnostic tool.
30.	net-tools	– utilities include: arp, hostname, ifconfig, netstat, rarp, route, plipconfig, slattach, mii-tool, iptunnel and ipmaddr.
31.	ncat	– improved re-implementation of the venerable netcat.
32.	netcat	– networking utility for reading/writing network connections.
33.	nethogs	– a small ‘net top’ tool.
34.	Netperf	– Network bandwidth Testing.
35.	netplan	– Netplan is a utility for easily configuring networking on a linux system.
36.	netsniff-ng	– Swiss army knife for daily Linux network plumbing.
37.	netwatch	– monitoring Network Connections.
38.	ngrep	– grep applied to the network layer.
39.	nload	– display network usage.
40.	nmap	– network discovery and security auditing.
41.	nmcli	– a command-line tool for controlling NetworkManager and reporting network status.
42.	nmtui	– provides a text interface to configure networking by controlling NetworkManager.
43.	nslookup	– query Internet name servers interactively.
44.	ping	– send icmp echo_request to network hosts.
45.	route	– show / manipulate the IP routing table.
46.	slurm	– network load monitor.

47. snort	– Network Intrusion Detection and Prevention System.
48. smokeping	– keeps track of our network latency.
49. socat	– establishes two bidir. byte streams and transfers data between them.
50. speedometer	– Measure and display the rate of data across a network.
51. speedtest-cli	– test internet bandwidth using speedtest.net
52. ss	– utility to investigate sockets.
53. ssh	– secure system admin. and file transfers over insecure networks.
54. tcpdump	– command-line packet analyzer.
55. tcptrack	– Displays information about tcp connections on a network interface.
56. telnet	– user interface to the TELNET protocol.
57. tracepath	– very similar function to traceroute.
58. traceroute	– print the route packets trace to the network host.
59. vnStat	– network traffic monitor.
60. websocat	– Connection forwarder from/to web sockets to/from usual sockets, in style of socat.
61. wget	– retrieving files using HTTP, HTTPS, FTP and FTPS.
62. Wireless Tools	– includes iwconfig, iwlist, iwspy, iwpriv and ifrename.
63. Wireshark	– network protocol analyzer.

Result:

The network configuration files and network commands in linux have been familiarised successfully.