

بسم الله الرحمن الرحيم

# Linux Administration For DevOps

By Motaz Adel Azeem  
code.sd

2024



```
Activities Terminal Jan 26 4:38 PM en
motaz@motaz-lenovo-t460: ~
top - 16:38:53 up 11:07, 1 user, load average: 0.58, 0.68, 0.71
Tasks: 282 total, 1 running, 281 sleeping, 0 stopped, 0 zombie
%Cpu(s): 1.4 us, 0.7 sy, 0.0 ni, 97.9 id, 0.0 wa, 0.0 hi, 0.0 st, 0.0 sr
MiB Mem : 7343.7 total, 975.7 free, 3754.5 used, 2613.5 buff/cache
MiB Swap: 12288.0 total, 11746.4 free, 541.6 used, 2559.2 avail Mem

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  %MEM    TIME+  COMMAND
 1972 couchdb  20   0 3736264 38236 6400  S   3.6   0.5  10:52.90 beam.smp
 2897 motaz    9  -11 4103232 12928 9524  S   1.7   0.2  10:07.75 pulseaudio
 1971 asterisk -11   0 2221492 41060 11488  S   1.3   0.5   5:06.36 asterisk
 1422 mysql    20   0 2314724 305504 4928  S   1.0  4.1   4:09.26 mysql
 3051 motaz    20   0 5355152 271580 64884  S   0.7   3.6  33:52.49 gnome-shell
 1311 tomcat   20   0 5381712 707176 8792  S   0.3   9.4   1:10.09 java
 24987 motaz   20   0 10.9g 375632 117576 S   0.3   5.0   2:11.30 Isolated Web Co
27880 motaz   20   0 12080 3932 3168  R   0.3   0.1   0:00.65 top
   1 root     20   0 168872 8208 4448  S   0.0   0.1   0:09.44 systemd
   2 root     20   0 0 0 0  S   0.0   0.0   0:00.04 kthreadd
   3 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 rcu_gp
   4 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 rcu_par_gp
   5 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 slub_flushwq
   6 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 netns
   8 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 kworker/0:0H-events_highpri
  10 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 mm_percpu_wq
  11 root     20   0 0 0 0  S   0.0   0.0   0:00.00 rcu_tasks_rude_
  12 root     20   0 0 0 0  S   0.0   0.0   0:00.00 rcu_tasks_trace
  13 root     20   0 0 0 0  S   0.0   0.0   0:01.81 ksoftirqd/0
  14 root     20   0 0 0 0  I   0.0   0.0   0:34.21 rcu_sched
  15 root     rt    0 0 0 0  S   0.0   0.0   0:00.19 migration/0
  16 root    -51   0 0 0 0  S   0.0   0.0   0:00.00 idle_inject/0
  18 root     20   0 0 0 0  S   0.0   0.0   0:00.00 cpuhp/0
  19 root     20   0 0 0 0  S   0.0   0.0   0:00.01 cpuhp/1
  20 root    -51   0 0 0 0  S   0.0   0.0   0:00.00 idle_inject/1
  21 root     rt    0 0 0 0  S   0.0   0.0   0:00.41 migration/1
  22 root     20   0 0 0 0  S   0.0   0.0   1:30.39 ksoftirqd/1
  24 root     0 -20 0 0 0  I   0.0   0.0   0:00.00 kworker/1:0H-events_highpri
```

# Linux administration for DevOps

2 February 2024

## Introduction

Linux is used in most servers for many services, such as web servers, contents hosting, database servers, FTP servers, telecom servers such as PBX, load balancer, firewall and router, yes Linux can turn PCs and servers into routers and firewalls instead of buying hardware appliances.

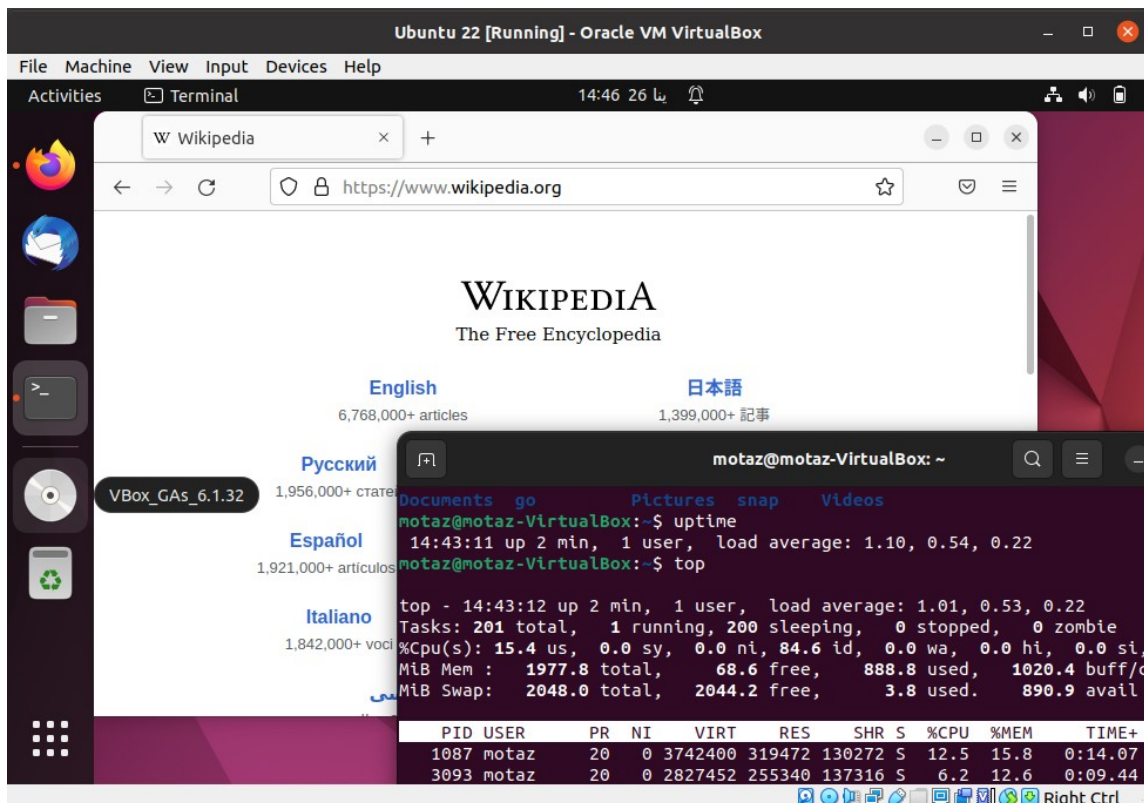
Currently more than 90% of internet servers are Linux according to many statistics, and top sites on the Internet are using Linux.

As a free OS, we can install Linux **as much as we need** in virtual machines, cloud servers, new PCs, old PCs, physical servers, single board computers, for testing, for production, for commercial, non-chimerical, and for academic institutes.

## Ubuntu

We will use [Ubuntu](#) Linux for commands and configurations in this book, most of commands and configurations are working the same in most Linux distributions specially for Debian family which Ubuntu is descending from.

To practice Linux commands using this book examples you have to install [Ubuntu](#) or any [Debian](#) based distribution in your PC or as virtual installation, you can install it as Server which contains command line terminal only, or as Desktop if you want to explore Linux as desktop alternative to your current OS.



Ubuntu running in Oracle [VirtualBox](#)

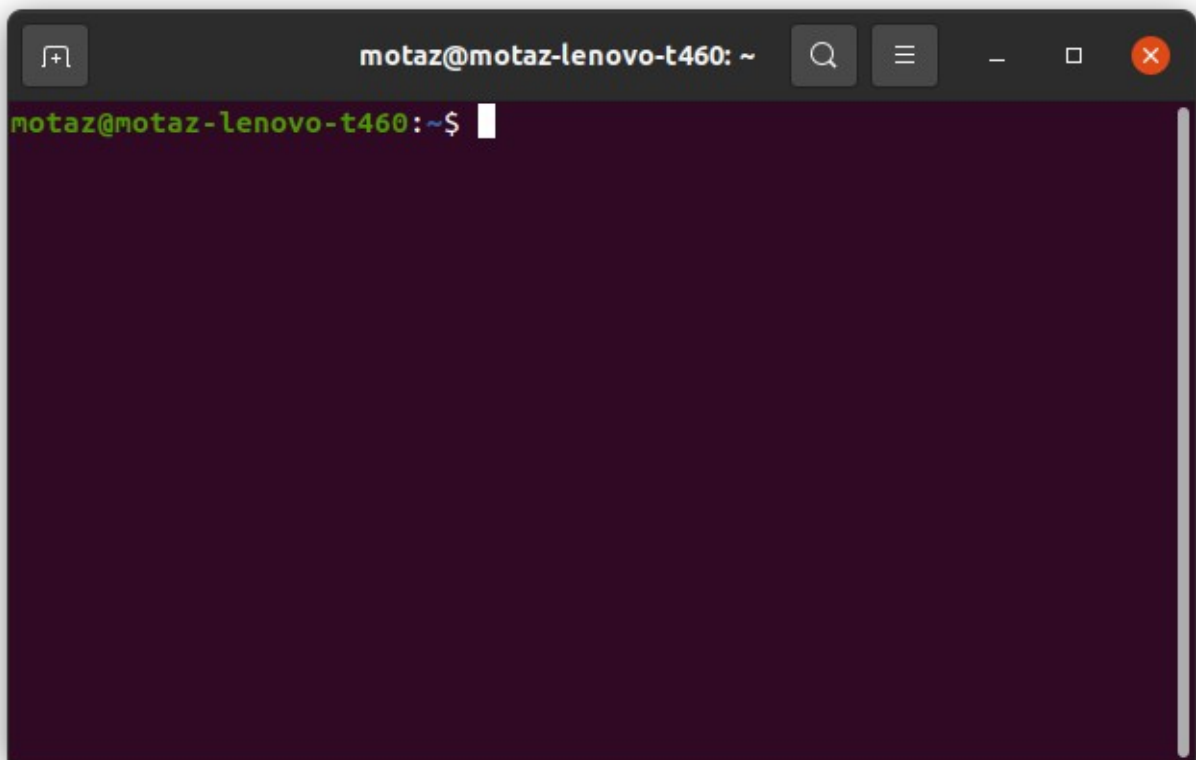
## Why Command line and terminal

Linux has a powerful and rich command line tools that let administrators and developers to control and manage OS resources such as files, applications, processes, and networking. Developers and DevOps could benefit from this commands to write shell scripts and use that command lines and tools as API for to be used in their software. Server versions of Ubuntu does not contain GUI, so that the best way to manage it is to learn how to use command line tools. In other hand GUI interface for Linux as desktop requires no study, the same like any other OS desktops.

## Terminal window

In order to execute Linux commands, you need to run terminal which is a command line tool that enables you to manage Linux, you can use Linux via terminal only specially in servers, no GUI is need to manage Linux servers.

There are many terminal tools, such as **bash** or **sh**, you can access them locally in Ubuntu desktop by opening Terminal, or clicking CTRL+ALT+F3 .. F6, to return to GUI click CTRL + ALT F2. For remote servers **ssh** tool will be used login to a server, in this case you need to install **openssh-server** in the server



## Author

Motaz Abel Azeem

I'm software developer and I have studied Linux on 1997, and another course in 2001. I have start using Linux at home since about 2002, then at work since 2010. Both courses were about Linux commands line, and this subject was Interesting, fun, deserve study and experimentation.

## Book license:

This books is licensed under [Creative Commons](#)



## Table of Contents

Introduction.....	2
Ubuntu.....	2
Why Command line and terminal.....	3
Terminal window.....	3
Author.....	4
Book license:.....	4
1.....	5
1.....	5
Files and directories commands.....	6
Redirection to a file.....	11
Grep command.....	12
Piping.....	15
Processes.....	17
Killing processes.....	19
Switching commands to background.....	20
CPU and Memory usage.....	21
Installing packages.....	23
Files Permissions.....	26
Files information.....	29
Connecting to remote server.....	30
Crontab jobs.....	34
Networking commands.....	36
Static and Automatic IPs.....	38
Routing and gateways.....	40
Networking troubleshooting.....	41
NGINX Web server.....	47
Reverse proxy.....	48
Load balancing.....	53
IPTables.....	54
Display system information.....	56

## Files and directories commands

After running terminal, we can run below simple commands:

1. **pwd** (print working directory) it displays current directory:

```
motaz@motaz-lenovo-t460:~$ pwd
/home/motaz
```

2. **cd** (change directory) it changes current directory to another directory:

```
motaz@motaz-lenovo-t460:~$ cd /
motaz@motaz-lenovo-t460:/$
```

It will change to root directory (/)

3. **ls** (list files and directories):

```
motaz@motaz-lenovo-t460:/$ ls
bin      dev      lib      libx32   mnt      root     snap     sys      var
boot    etc      lib32    lost+found  opt      run      srv      tmp
cdrom   home    lib64    media     proc     sbin     swapfile  usr
```

To display more information, we can add **-l** (long listing format):

```
motaz@motaz-lenovo-t460:/$ ls -l
total 2097248
lrwxrwxrwx  1 root root          7 Dec  5  2020 bin -> usr/bin
drwxr-xr-x  4 root root    4096 Dec 28 05:23 boot
drwxrwxr-x  2 root root    4096 Dec  5  2020 cdrom
drwxr-xr-x 22 root root   4960 Dec 30 05:12 dev
drwxr-xr-x 167 root root  12288 Dec 31 06:25 etc
drwxr-xr-x  6 root root    4096 Nov 13 07:41 home
lrwxrwxrwx  1 root root          7 Dec  5  2020 lib -> usr/lib
lrwxrwxrwx  1 root root          9 Dec  5  2020 lib32 -> usr/lib32
lrwxrwxrwx  1 root root          9 Dec  5  2020 lib64 -> usr/lib64
lrwxrwxrwx  1 root root         10 Dec  5  2020 libx32 -> usr/libx32
drwx----- 2 root root  16384 Dec  5  2020 lost+found
drwxr-xr-x  3 root root    4096 May 25  2022 media
drwxr-xr-x  2 root root    4096 Jul 31  2020 mnt
drwxr-xr-x  8 root root    4096 Aug 19 07:30 opt
dr-xr-xr-x 357 root root      0 Dec 30 05:12 proc
drwx----- 11 root root    4096 Dec  2 08:15 root
drwxr-xr-x 46 root root   1320 Dec 31 07:53 run
lrwxrwxrwx  1 root root          8 Dec  5  2020 sbin -> usr/sbin
drwxr-xr-x 23 root root    4096 Nov  6 05:47 snap
drwxr-xr-x  3 root root    4096 Nov  3  2022 srv
-rw-----  1 root root 2147483648 Dec 29  2020 swapfile
dr-xr-xr-x 13 root root      0 Dec 30 05:12 sys
drwxrwxrwt 28 root root   20480 Dec 31 07:53 tmp
drwxr-xr-x 14 root root    4096 Jul 31  2020 usr
drwxr-xr-x 16 root root    4096 Dec 25  2021 var
```

**cd ~** will return to current user home directory, or just **cd**:

[code.sd](#)

```
motaz@motaz-lenovo-t460:/$ cd
motaz@motaz-lenovo-t460:~$
```

Note that prompt contains information in below format:

`username@computername:(directory name)`, example:

```
motaz@motaz-lenovo-t460:~$
```

In this case user is (motaz), computer name is (motaz-lenovo-t460), and current directory is home (~)

4. **mkdir**: it creates new directory, for example:

```
motaz@motaz-lenovo-t460:~$ mkdir linux
motaz@motaz-lenovo-t460:~$
```

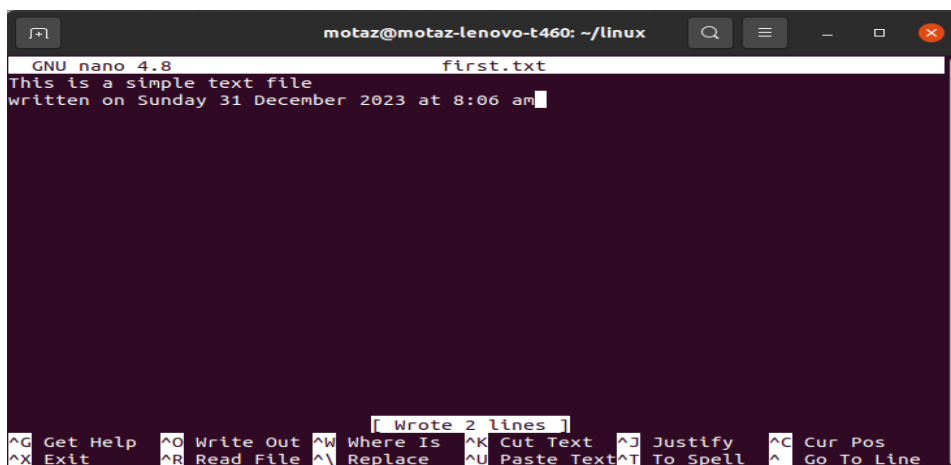
You have to make sure that current user has permission to create a new directory in current folder, in our example we have created it in our home directory `/home/motaz`

5. **touch**: it creates new empty file:

```
motaz@motaz-lenovo-t460:~$ cd linux
motaz@motaz-lenovo-t460:~/linux$ ls
motaz@motaz-lenovo-t460:~/linux$ touch first.txt
motaz@motaz-lenovo-t460:~/linux$ ls -l
total 0
-rw-r--r-- 1 motaz motaz 0 Dec 31 08:04 first.txt
```

6. **nano** (simple command line editor) to edit this file

```
motaz@motaz-lenovo-t460:~/linux$ nano first.txt
```



```
motaz@motaz-lenovo-t460: ~/linux
GNU nano 4.8 first.txt
This is a simple text file
written on Sunday 31 December 2023 at 8:06 am
Wrote 2 lines
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

After that we use `Ctrl+O` to save modifications and then `Ctrl+x` to exist to terminal:

[code.sd](#)

7. **cat** : displays text file contents:

```
motaz@motaz-lenovo-t460:~/linux$ cat first.txt
This is a simple text file
written on Sunday 31 December 2023 at 8:06 am
```

8. **mv**: to rename file or directory:

```
motaz@motaz-lenovo-t460:~/linux$ mv first.txt newfile.txt
```

9. **rm**: remove file :

```
motaz@motaz-lenovo-t460:~/linux$ touch tempfile.txt
motaz@motaz-lenovo-t460:~/linux$ ls
3 newfile.txt tempfile.txt
motaz@motaz-lenovo-t460:~/linux$ rm tempfile.txt
motaz@motaz-lenovo-t460:~/linux$ ls
3 newfile.txt
motaz@motaz-lenovo-t460:~/linux$
```

10. **rmdir**: remove directory, but the directory should be empty:

```
motaz@motaz-lenovo-t460:~/linux$ mkdir dir
motaz@motaz-lenovo-t460:~/linux$ rm 3/three
dir newfile.txt
motaz@motaz-lenovo-t460:~/linux$ touch dir/test.txt
motaz@motaz-lenovo-t460:~/linux$ rmdir dir
rmdir: failed to remove 'dir': Directory not empty
motaz@motaz-lenovo-t460:~/linux$
```

In this case for not empty directory it can be removed by (rm) command using (-r) recursive parameter:

```
motaz@motaz-lenovo-t460:~/linux$ rm -r dir
motaz@motaz-lenovo-t460:~/linux$
```

11. **cp**: copy file or files:

```
motaz@motaz-lenovo-t460:~/linux$ cp newfile.txt secondcopy.txt
```

**Hint:**

You can start the typing command part or filename and press (tab) button to complete the name or suggest name choices

[code.sd](#)



To create hidden file, add dot (.) as prefix to the name of file, for example **.hidden.txt**:

```
motaz@motaz-lenovo-t460:~/linux$ nano .hidden.txt
motaz@motaz-lenovo-t460:~/linux$ ls
first.sh      log.txt      memory.txt  newfile.txt  root.lst     second.sh    subfolder
lastfile.txt  memory.log   nano.save   nohup.out    secondcopy.txt  second.text  test.txt
```

To show all files including hidden files use (-a) parameter:

```
motaz@motaz-lenovo-t460:~/linux$ ls -a
.  first.sh      lastfile.txt  memory.log   nano.save    nohup.out    secondcopy.txt
second.text  test.txt
..  .hidden.txt   log.txt      memory.txt  newfile.txt  root.lst     second.sh
subfolder

motaz@motaz-lenovo-t460:~/linux$ ls -lha
total 68K
drwxr-xr-x  3 motaz www-data 4.0K Jan  5 14:53 .
drwxr-xr-x 102 motaz motaz   4.0K Jan  4 17:47 ..
-rwxrwxr-x  1 root  www-data  42 Jan  2 16:15 first.sh
-rw-rw-r--  1 motaz motaz    12 Jan  5 14:53 .hidden.txt
-rw-rw-r--  1 motaz motaz    10 Jan  4 07:49 lastfile.txt
-r--r--r--  1 motaz www-data   64 Dec 31 10:01 log.txt
-rw-r--r--  1 motaz www-data  200 Jan  1 06:55 memory.log
-rw-r--r--  1 motaz www-data 1.4K Dec 31 09:58 memory.txt
-rw-----  1 motaz www-data    5 Jan  2 13:32 nano.save
-rw-r--r--  1 motaz www-data   73 Dec 31 08:06 newfile.txt
-rw-----  1 motaz www-data  850 Jan  2 13:30 nohup.out
-rw-r--r--  1 motaz www-data 1.3K Dec 31 09:57 root.lst
-rw-r--r--  1 motaz www-data   73 Dec 31 09:22 secondcopy.txt
-rwxrwxr-x  1 motaz motaz    48 Jan  3 16:24 second.sh
-rw-r--r--  1 motaz www-data   12 Jan  2 09:11 second.text
drwxrwxr-x  2 motaz www-data 4.0K Jan  2 15:55 subfolder
--wxr--r--  1 motaz www-data    5 Jan  2 09:15 test.txt
```

For large files there are important useful commands : **head**, **more** and **tail**

1. **head**: display first lines of text file, for example:

```
motaz@motaz-lenovo-t460:~/linux$ head /var/log/syslog
Dec 18 05:24:37 motaz-lenovo-t460 rsyslogd: [origin software="rsyslogd"
swVersion="8.2001.0" x-pid="1041" x-info="https://www.rsyslog.com"] rsyslogd was HUPed
Dec 18 05:24:37 motaz-lenovo-t460 systemd[1]: logrotate.service: Main process exited,
code=exited, status=1/FAILURE
Dec 18 05:24:37 motaz-lenovo-t460 systemd[1]: logrotate.service: Failed with result
'exit-code'.
Dec 18 05:24:37 motaz-lenovo-t460 systemd[1]: Failed to start Rotate log files.
Dec 18 05:24:37 motaz-lenovo-t460 systemd[1]: dmesg.service: Succeeded.
Dec 18 05:24:37 motaz-lenovo-t460 ntpd[1271]: error resolving pool ntp.ubuntu.com:
Temporary failure in name resolution (-3)
Dec 18 05:24:38 motaz-lenovo-t460 /usr/lib/gdm3/gdm-wayland-session[1496]: dbus-
daemon[1496]: [session uid=125 pid=1496] Activating service
name='org.freedesktop.systemd1' requested by ':1.1' (uid=125 pid=1497
comm="/usr/libexec/gnome-session-binary --systemd --auto" label="unconfined")
Dec 18 05:24:38 motaz-lenovo-t460 /usr/lib/gdm3/gdm-wayland-session[1496]: dbus-
daemon[1496]: [session uid=125 pid=1496] Activated service 'org.freedesktop.systemd1'
failed: Process org.freedesktop.systemd1 exited with status 1
Dec 18 05:24:38 motaz-lenovo-t460 /usr/lib/gdm3/gdm-wayland-session[1496]: dbus-
daemon[1496]: [session uid=125 pid=1496] Activating service
name='org.freedesktop.systemd1' requested by ':1.1' (uid=125 pid=1497
comm="/usr/libexec/gnome-session-binary --systemd --auto" label="unconfined")
```

[code.sd](#)

```
Dec 18 05:24:38 motaz-lenovo-t460 /usr/lib/gdm3/gdm-wayland-session[1496]: dbus-daemon[1496]: [session uid=125 pid=1496] Activated service 'org.freedesktop.systemd1' failed: Process org.freedesktop.systemd1 exited with status 1
```

We can add **-n <lines count>** to display specific lines count

2. **more <filename>**: is used to scroll down on files content using Enter button to show more lines, or show next pages using Space bar. Pressing **q** will exist more commands

3. **tail**: it shows last lines of file, and this is often used for log files to display the last events or errors. To monitor new events on log file, such as Apache/Nginx access.log file vists we can add **-f** to display log actively and monitoring it. If you have opened many applications, you can execute below command and keep monitoring:

```
motaz@motaz-lenovo-t460:~/linux$ tail -f /var/log/syslog
Jan  5 15:53:46 motaz-lenovo-t460 tracker-extract[37852]: Setting priority nice level to 19
Jan  5 15:53:46 motaz-lenovo-t460 dbus-daemon[2805]: [session uid=1000 pid=2805] Successfully activated service 'org.freedesktop.Tracker1.Miner.Extract'
Jan  5 15:53:46 motaz-lenovo-t460 systemd[2794]: Started Tracker metadata extractor.
Jan  5 15:53:56 motaz-lenovo-t460 systemd[2794]: tracker-extract.service: Succeeded.
Jan  5 15:54:16 motaz-lenovo-t460 tracker-store[37844]: OK
Jan  5 15:54:16 motaz-lenovo-t460 systemd[2794]: tracker-store.service: Succeeded.
Jan  5 15:54:58 motaz-lenovo-t460 thunderbird.desktop[24404]: console.error: mailnews.pop3.89: "NetworkTimeoutError: a Network error occurred"
Jan  5 15:54:58 motaz-lenovo-t460 thunderbird.desktop[24404]: console.error: mailnews.pop3.89: "SecurityError info: "
Jan  5 15:55:36 motaz-lenovo-t460 ntpd[1142]: 185.125.190.56 local addr 192.168.44.211 -> <null>
Jan  5 15:56:24 motaz-lenovo-t460 ntpd[1142]: 196.192.32.7 local addr 192.168.44.211 -> <null>
```

We can keep monitoring specific events or text by piping **tail -f** output to **grep** command:

```
motaz@motaz-lenovo-t460:~/linux$ tail -f /var/log/syslog | grep error
Jan  5 15:54:58 motaz-lenovo-t460 thunderbird.desktop[24404]: console.error: mailnews.pop3.89: "NetworkTimeoutError: a Network error occurred"
Jan  5 15:54:58 motaz-lenovo-t460 thunderbird.desktop[24404]: console.error: mailnews.pop3.89: "SecurityError info: "
```

## Redirection to a file

OS in general can redirect output from standard output device (monitor) to a file using (>) directive , example: below command write ls command output in a file:

```
motaz@motaz-lenovo-t460:~/linux$ ls / -lh > root.lst
motaz@motaz-lenovo-t460:~/linux$ cat root.lst
total 2.1G
lrwxrwxrwx   1 root root    7 Dec  5  2020 bin -> usr/bin
drwxr-xr-x   4 root root 4.0K Dec 28 05:23 boot
drwxrwxr-x   2 root root 4.0K Dec  5  2020 cdrom
drwxr-xr-x  22 root root 4.9K Dec 30 05:12 dev
drwxr-xr-x 167 root root 12K Dec 31 09:46 etc
drwxr-xr-x   6 root root 4.0K Nov 13 07:41 home
lrwxrwxrwx   1 root root    7 Dec  5  2020 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Dec  5  2020 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Dec  5  2020 lib64 -> usr/lib64
lrwxrwxrwx   1 root root   10 Dec  5  2020 libx32 -> usr/libx32
drwx-----  2 root root 16K Dec  5  2020 lost+found
drwxr-xr-x   3 root root 4.0K May 25  2022 media
drwxr-xr-x   2 root root 4.0K Jul 31  2020 mnt
drwxr-xr-x   8 root root 4.0K Aug 19 07:30 opt
dr-xr-xr-x 355 root root    0 Dec 30 05:12 proc
drwx----- 11 root root 4.0K Dec  2 08:15 root
drwxr-xr-x  46 root root 1.3K Dec 31 08:59 run
lrwxrwxrwx   1 root root    8 Dec  5  2020 sbin -> usr/sbin
drwxr-xr-x  23 root root 4.0K Nov  6 05:47 snap
drwxr-xr-x   3 root root 4.0K Nov  3  2022 srv
-rw-----   1 root root 2.0G Dec 29  2020 swapfile
dr-xr-xr-x  13 root root    0 Dec 30 05:12 sys
drwxrwxrwt  27 root root 20K Dec 31 09:53 tmp
drwxr-xr-x  14 root root 4.0K Jul 31  2020 usr
drwxr-xr-x  16 root root 4.0K Dec 25  2021 var
```

Below command stores memory information in a file:

```
motaz@motaz-lenovo-t460:~/linux$ cat /proc/meminfo > memory.txt
motaz@motaz-lenovo-t460:~/linux$ cat memory.txt
MemTotal:       7519972 kB
MemFree:        190408 kB
MemAvailable:   2252292 kB
Buffers:        404780 kB
Cached:         2332044 kB
SwapCached:     4572 kB
Active:         1778512 kB
Inactive:       4552460 kB
Active(anon):   516540 kB
Inactive(anon): 3852304 kB
Active(file):   1261972 kB
Inactive(file): 700156 kB
Unevictable:    239332 kB
Mlocked:        48 kB
SwapTotal:     12582904 kB
SwapFree:      12343064 kB
Dirty:          156 kB
Writeback:      0 kB
AnonPages:     3831052 kB
Mapped:        685784 kB
Shmem:         775832 kB
KReclaimable:  402956 kB
Slab:          558592 kB
SReclaimable:  402956 kB
```

```
SUnreclaim:      155636 kB
KernelStack:     19936 kB
PageTables:      44316 kB
NFS_Unstable:    0 kB
Bounce:          0 kB
WritebackTmp:    0 kB
CommitLimit:    16342888 kB
Committed_AS:    11309868 kB
VmallocTotal:    34359738367 kB
VmallocUsed:     52500 kB
VmallocChunk:    0 kB
Percpu:         4128 kB
HardwareCorrupted: 0 kB
AnonHugePages:   0 kB
ShmemHugePages:  0 kB
ShmemPmdMapped:  0 kB
FileHugePages:   0 kB
FilePmdMapped:   0 kB
HugePages_Total: 0
HugePages_Free:  0
HugePages_Rsvd:  0
HugePages_Surp:  0
Hugepagesize:    2048 kB
Hugetlb:         0 kB
DirectMap4k:     452648 kB
DirectMap2M:     7321600 kB
DirectMap1G:     0 kB
```

To store current date and time in a file, use below command:

```
motaz@motaz-lenovo-t460:~/linux$ date > log.txt
motaz@motaz-lenovo-t460:~/linux$ cat log.txt
Sun 31 Dec 2023 10:00:16 AM CAT
```

Note that every time executing that command, *log.txt* will be erased. To keep it we can use the directive `>>` to append the file instead of overwrite it:

```
motaz@motaz-lenovo-t460:~/linux$ date >> log.txt
motaz@motaz-lenovo-t460:~/linux$ cat log.txt
Sun 31 Dec 2023 10:00:16 AM CAT
Sun 31 Dec 2023 10:01:38 AM CAT
```

## Grep command

grep command is used to search for specific text in files or in another command output, for example if we write below command:

```
grep hello
```

It will let you enter text and search for a match of hello, if found it will display it in red text, and displays the entire line that contains a match, else it will ignore the input, and to exist from it using Ctrl+C

```
motaz@motaz-lenovo-t460:~/linux$ grep hello
Hi
```

[code.sd](#)

```
this is sample text
hello there
hello there
^C
```

Also it could be used to search text in files, for example let us display first this memory file **/proc/cpuinfo** that contains CPU information

```
motaz@motaz-lenovo-t460:~/linux$ cat /proc/cpuinfo
processor      : 0
vendor_id    : GenuineIntel
cpu family   : 6
model        : 78
model name   : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
stepping     : 3
microcode    : 0xf0
cpu MHz      : 699.999
cache size   : 3072 KB
physical id  : 0
siblings     : 4
core id      : 0
cpu cores    : 2
apicid       : 0
initial apicid : 0
fpu          : yes
fpu_exception: yes
cpuid level  : 22
wp           : yes
flags        : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc
art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni
pclmulqdq dtes64 monitor ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1
sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch cpuid_fault epb invpcid_single pti ssbd ibrs ibpb stibp tpr_shadow vnmi
flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx
rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln
pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_lld arch_capabilities
vmx flags    : vnmi preemption_timer invvpid ept_x_only ept_ad ept_lgb flexpriority
tsc_offset vtptr mtf vapic ept vpid unrestricted_guest ple shadow_vmcs pml
bugs         : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass lltf mds swapgs taa
itlb_multihit srbds mmio_stale_data retbleed
bogomips     : 4999.90
clflush size : 64
cache_alignment : 64
address sizes: 39 bits physical, 48 bits virtual
power management:
...
```

We can filter processor line to know core count:

```
motaz@motaz-lenovo-t460:~/linux$ grep proc /proc/cpuinfo
processor      : 0
processor      : 1
processor      : 2
processor      : 3
```

To search for a text with space add double quotations ("):

```
motaz@motaz-lenovo-t460:~/linux$ grep "model name" /proc/cpuinfo
```

[code.sd](#)

```
model name : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
model name : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
model name : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
model name : Intel(R) Core(TM) i5-6300U CPU @ 2.40GHz
```

Here is memory information file, note that text is case sensitive:

```
motaz@motaz-lenovo-t460:~/linux$ grep Mem /proc/meminfo
MemTotal:      7519960 kB
MemFree:       132300 kB
MemAvailable:  4012588 kB
```

To ignore letters case, add -i option:

```
motaz@motaz-lenovo-t460:~/linux$ grep -i Mem /proc/meminfo
MemTotal:      7519960 kB
MemFree:       130924 kB
MemAvailable:  4008912 kB
Shmem:         229380 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
```

Here is an example of redirecting result in a file:

```
motaz@motaz-lenovo-t460:~/linux$ grep -i Mem /proc/meminfo > memory.log
```

We can run multiple commands using semi-column (;)

```
motaz@motaz-lenovo-t460:~/linux$ date; ls; cat memory.log
Mon 01 Jan 2024 06:53:42 AM CAT
log.txt  memory.log  memory.txt  newfile.txt  root.lst  secondcopy.txt
MemTotal:      7519960 kB
MemFree:       146080 kB
MemAvailable:  3993808 kB
Shmem:         257748 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
```

We can benefit from this feature to add date and time to redirected memory log file:

```
date > memory.log; grep -i Mem /proc/meminfo >> memory.log
motaz@motaz-lenovo-t460:~/linux$ cat memory.log
Mon 01 Jan 2024 06:55:53 AM CAT
MemTotal:      7519960 kB
MemFree:       148448 kB
MemAvailable:  3996832 kB
Shmem:         257844 kB
ShmemHugePages: 0 kB
ShmemPmdMapped: 0 kB
```

Header for above output file will contain current date and time, in which this information has get

[code.sd](https://code.sd)

## Piping

Piping is to send output from command to another command or program, example, displaying all text files contents using *cat* command and search for lines that contains specific text using *grep*:

```
motaz@motaz-lenovo-t460:~/linux$ cat *.txt | grep 20
Sun 31 Dec 2023 10:00:16 AM CAT
Sun 31 Dec 2023 10:01:38 AM CAT
Cached:          2332044 kB
Hugepagesize:    2048 kB
written on Sunday 31 December 2023 at 8:06 am
written on Sunday 31 December 2023 at 8:06 am
```

Here another example to search for errors in system log:

```
cat /var/log/syslog | grep error
Jan  1 10:08:18 motaz-lenovo-t460 ntpd[1274]: error resolving pool 3.ubuntu.pool.ntp.org:
Temporary failure in name resolution (-3)
Jan  1 10:08:28 motaz-lenovo-t460 ntpd[1274]: error resolving pool 0.ubuntu.pool.ntp.org:
Temporary failure in name resolution (-3)
Jan  1 10:08:46 motaz-lenovo-t460 ntpd[1274]: error resolving pool ntp.ubuntu.com:
Temporary failure in name resolution (-3)
Jan  1 10:08:56 motaz-lenovo-t460 ntpd[1274]: error resolving pool 2.ubuntu.pool.ntp.org:
Temporary failure in name resolution (-3)
Jan  1 10:09:06 motaz-lenovo-t460 ntpd[1274]: error resolving pool 1.ubuntu.pool.ntp.org:
Temporary failure in name resolution (-3)
```

Here is useful list files parameter to sort files according to their updated time, parameter is **(t)**, it will display newer files at top:

```
motaz@motaz-lenovo-t460:~/linux$ ls -lt
total 56
-rw-rw-r-- 1 motaz motaz      10 Jan  4 07:49 lastfile.txt
-rwxrwxr-x 1 motaz motaz      48 Jan  3 16:24 second.sh
-rwxrwxr-x 1 root  www-data    42 Jan  2 16:15 first.sh
drwxrwxr-x 2 motaz www-data 4096 Jan  2 15:55 subfolder
-rw----- 1 motaz www-data     5 Jan  2 13:32 nano.save
-rw----- 1 motaz www-data   850 Jan  2 13:30 nohup.out
--wxr--r-- 1 motaz www-data     5 Jan  2 09:15 test.txt
-rw-r--r-- 1 motaz www-data    12 Jan  2 09:11 second.text
-rw-r--r-- 1 motaz www-data   200 Jan  1 06:55 memory.log
-r--r--r-- 1 motaz www-data    64 Dec 31 10:01 log.txt
-rw-r--r-- 1 motaz www-data  1419 Dec 31 09:58 memory.txt
-rw-r--r-- 1 motaz www-data  1289 Dec 31 09:57 root.lst
-rw-r--r-- 1 motaz www-data    73 Dec 31 09:22 secondcopy.txt
-rw-r--r-- 1 motaz www-data    73 Dec 31 08:06 newfile.txt
```

If you want to get new and updated files at the bottom (which I prefer) add **(r)** parameter:

```
motaz@motaz-lenovo-t460:~/linux$ ls -ltr
total 56
-rw-r--r-- 1 motaz www-data    73 Dec 31 08:06 newfile.txt
-rw-r--r-- 1 motaz www-data    73 Dec 31 09:22 secondcopy.txt
-rw-r--r-- 1 motaz www-data  1289 Dec 31 09:57 root.lst
-rw-r--r-- 1 motaz www-data  1419 Dec 31 09:58 memory.txt
-r--r--r-- 1 motaz www-data    64 Dec 31 10:01 log.txt
-rw-r--r-- 1 motaz www-data   200 Jan  1 06:55 memory.log
-rw-r--r-- 1 motaz www-data    12 Jan  2 09:11 second.text
--wxr--r-- 1 motaz www-data     5 Jan  2 09:15 test.txt
```

```
-rw----- 1 motaz www-data 850 Jan 2 13:30 nohup.out
-rw----- 1 motaz www-data 5 Jan 2 13:32 nano.save
drwxrwxr-x 2 motaz www-data 4096 Jan 2 15:55 subfolder
-rwxrwxr-x 1 root www-data 42 Jan 2 16:15 first.sh
-rwxrwxr-x 1 motaz motaz 48 Jan 3 16:24 second.sh
-rw-r--r-- 1 motaz motaz 0 Jan 3 16:25 first.hcd
-rw-rw-r-- 1 motaz motaz 10 Jan 4 07:49 lastfile.txt
```

**wc** (Words Count) command is used with piping, to calculate lines number or words number in text file, but first let us use it without file, execute `wc` command and write below text then press Enter then CTRL+D:

```
motaz@motaz-lenovo-t460:~$ wc
Hello there,
This is words count command (WC) in Linux
      2      10      55
```

It shows lines count (2), and words count (10) and characters count (55)

To show lines count only add `-l` option:

```
motaz@motaz-lenovo-t460:~$ wc -l
first line
second line
third line
3
```

We can use it with piping with other commands such as `cat`:

```
motaz@motaz-lenovo-t460:~$ cat /proc/cpuinfo | wc -l
112
```

Also with `grep` to search for specific word to count it, such as *processor* in *cpuinfo* file to count how many cores we have:

```
motaz@motaz-lenovo-t460:~$ cat /proc/cpuinfo | grep processor
processor      : 0
processor      : 1
processor      : 2
processor      : 3

motaz@motaz-lenovo-t460:~$ cat /proc/cpuinfo | grep processor | wc -l
4
```

We can use it with Apache or NGINX log to count specific requests for certain URL, method or client IP:

```
admin@server:~$ sudo cat /var/log/nginx/access.log | grep iplocation | wc -l
253
```

[code.sd](https://code.sd)



**history** command retrieves previous commands that has executed in terminal, also we could traverse through previous history commands in terminal using up arrow, if you don't want to store current command in history, press space before typing command. History command could be used with **grep** to search for specific command:

```
motaz@motaz-lenovo-t460:~$ history | grep cat
1282  cat /proc/cpuinfo | wc -l
1283  cat /proc/cpuinfo | grep CPU
1284  cat /proc/cpuinfo
1285  cat /proc/cpuinfo | grep processor
1286  cat /proc/cpuinfo | grep processor | wc -l
1334  cat /etc/resolv.conf
```

## Processes

Processes in Linux are programs, services, and threads that runs for specific time or running all the time while machine is running. Each process has unique process ID and owner that has initialized and run that process, except for **init** process that has process number 1.

**init** process is the first starting process when booting the system, and it runs other main processes, also it adopts orphans that owners has disowned or crashed.

**Top** command is used to display top processes and system load and memory usage:

```
motaz@motaz-lenovo-t460:~/linux$ top

top - 14:21:33 up 9:00, 1 user, load average: 0.13, 0.24, 0.33
Tasks: 271 total, 1 running, 270 sleeping, 0 stopped, 0 zombie
%Cpu(s): 6.4 us, 5.1 sy, 0.0 ni, 88.5 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7343.7 total, 129.5 free, 2636.5 used, 4577.7 buff/cache
MiB Swap: 12288.0 total, 11767.7 free, 520.3 used. 4030.4 avail Mem

  PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM    TIME+  COMMAND
 3018 motaz    20   0 5136776 260024 73940 S  11.1   3.5   25:41.99  gnome-shell
 1865 couchdb  20   0 3739848  38360  7588 S   5.6   0.5   12:50.58  beam.smp
10477 motaz    20   0 407636  49080 33940 S   5.6   0.7    0:19.05  gnome-terminal-
15232 motaz    20   0  12100   3832  3160 R   5.6   0.1    0:00.04  top
   1 root     20   0 169128  10700  6712 S   0.0   0.1    0:06.90  systemd
   2 root     20   0     0     0     0 S   0.0   0.0    0:00.02  kthreadd
   3 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  rcu_gp
   4 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  rcu_par_gp
   5 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  slub_flushwq
   6 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  netns
   8 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  kworker/0:0H-
events_highpri
  10 root     0 -20     0     0     0 I   0.0   0.0    0:00.00  mm_percpu_wq
```

Press **q** or **Ctrl+C** to exist **top** command.

[code.sd](#)

To display all processes type **ps -e**

```
motaz@motaz-lenovo-t460:~/linux$ ps -e
  PID TTY          TIME CMD
    1 ?            00:00:07 systemd
    2 ?            00:00:00 kthreadd
    3 ?            00:00:00 rcu_gp
    4 ?            00:00:00 rcu_par_gp
    5 ?            00:00:00 slub_flushwq
    6 ?            00:00:00 netns
    8 ?            00:00:00 kworker/0:0H-events_highpri
   10 ?            00:00:00 mm_percpu_wq
   11 ?            00:00:00 rcu_tasks_rude_
   12 ?            00:00:00 rcu_tasks_trace
   13 ?            00:00:02 ksoftirqd/0
   14 ?            00:00:37 rcu_sched
 2943 tty2        00:00:00 gdm-wayland-ses
 2946 tty2        00:00:00 gnome-session-b
```

It displays process ID, Terminal name, CPU usage time and command or program name

For more information about processes, add (f) as parameter:

```
motaz@motaz-lenovo-t460:~/linux$ ps -ef
UID          PID    PPID  C  STIME TTY          TIME CMD
root           1      0  0  05:21 ?            00:00:07 /sbin/init splash
root           2      0  0  05:21 ?            00:00:00 [kthreadd]
root           3      2  0  05:21 ?            00:00:00 [rcu_gp]
root           4      2  0  05:21 ?            00:00:00 [rcu_par_gp]
root           5      2  0  05:21 ?            00:00:00 [slub_flushwq]
root           6      2  0  05:21 ?            00:00:00 [netns]
root           8      2  0  05:21 ?            00:00:00 [kworker/0:0H-events_highpri]
root          10      2  0  05:21 ?            00:00:00 [mm_percpu_wq]
root          11      2  0  05:21 ?            00:00:00 [rcu_tasks_rude_]
root          12      2  0  05:21 ?            00:00:00 [rcu_tasks_trace]
root          13      2  0  05:21 ?            00:00:02 [ksoftirqd/0]
root          14      2  0  05:21 ?            00:00:37 [rcu_sched]
root          15      2  0  05:21 ?            00:00:00 [migration/0]
root          16      2  0  05:21 ?            00:00:00 [idle_inject/0]
root          18      2  0  05:21 ?            00:00:00 [cpuhp/0]
root         18767      2  0  15:37 ?            00:00:00 [kworker/2:2-events]
motaz        18768    10482  0  15:37 pts/0        00:00:00 ping 1.1.1.1
motaz        18770    2856  31  15:37 ?            00:03:14 /usr/lib/firefox/firefox -ne
```

It adds new columns, such as user name, parent process ID, starting time or day

We can search for specific process by piping output of **ps -ef** to **grep**, for example searching for terminal process:

```
motaz@motaz-lenovo-t460:~/linux$ ps -ef | grep term
motaz        10477    2856  0  10:06 ?            00:00:48 /usr/libexec/gnome-terminal-server
motaz        19356    15050  0  15:53 pts/1        00:00:00 grep --color=auto term
```

Found process is the process ID of 10477, and has been started at 10:06 am, second line is the **grep** command. If process not found we will get only the **grep** process, example:

```
motaz@motaz-lenovo-t460:~/linux$ ps -ef | grep noexist-process
motaz        19375    15050  0  15:56 pts/1        00:00:00 grep --color=auto noexist-process
```

[code.sd](#)

using aux parameters will display even more process information including CPU and memory usage:

```
motaz@motaz-lenovo-t460:~$ ps aux
USER          PID  %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1   0.0  0.1 169128 10688 ?        Ss   05:21   0:08 /sbin/init splash
root           2   0.0  0.0      0     0 ?        S    05:21   0:00 [kthreadd]
root           3   0.0  0.0      0     0 ?        I<   05:21   0:00 [rcu_gp]
root           4   0.0  0.0      0     0 ?        I<   05:21   0:00 [rcu_par_gp]
root           5   0.0  0.0      0     0 ?        I<   05:21   0:00 [slub_flushwq]
root           6   0.0  0.0      0     0 ?        I<   05:21   0:00 [netns]
root           8   0.0  0.0      0     0 ?        I<   05:21   0:00 [kworker/0:0H-events_highpri]
root          10   0.0  0.0      0     0 ?        I<   05:21   0:00 [mm_percpu_wq]
root          11   0.0  0.0      0     0 ?        S    05:21   0:00 [rcu_tasks_rude_]
root          12   0.0  0.0      0     0 ?        S    05:21   0:00 [rcu_tasks_trace]
root          13   0.0  0.0      0     0 ?        S    05:21   0:02 [ksoftirqd/0]
root          14   0.1  0.0      0     0 ?        I    05:21   0:43 [rcu_sched]
root          15   0.0  0.0      0     0 ?        S    05:21   0:00 [migration/0]
root          16   0.0  0.0      0     0 ?        S    05:21   0:00 [idle_inject/0]
root          18   0.0  0.0      0     0 ?        S    05:21   0:00 [cpuhp/0]
root          19   0.0  0.0      0     0 ?        S    05:21   0:00 [cpuhp/1]
root          20   0.0  0.0      0     0 ?        S    05:21   0:00 [idle_inject/1]
root          21   0.0  0.0      0     0 ?        S    05:21   0:00 [migration/1]
motaz         3018  5.1  3.4 5146048 261252 ?        Rsl  05:21  35:20 /usr/bin/gnome-shell
motaz        18770 22.0  5.6 3905096 427524 ?        Sl   15:37  16:45 /usr/lib/firefox/firefox
motaz        20406 52.6  7.9 3478884 597304 ?        Sl   16:51  1:17 /usr/lib/thunderbird/thunderbird
```

## Killing processes

Sometimes we need to stop process or program that is not responding, or that run in background, we can stop it using one of two below methods:

1. **killall**, this kills process by name, example:

```
motaz@motaz-lenovo-t460:~/linux$ killall gnome-terminal-server
```

Use tab to complete process name. This will close current terminal shell. If killing process gracefully has failed, we can add **-9** parameter to kill it by force.

```
killall -9 processname
```

2. **kill**, this requires process id, to get it, we can search using **ps -ef | grep** commands

To test it, run **nano** command in another terminal window, and keep it open, then search for it:

```
motaz@motaz-lenovo-t460:~$ ps -ef | grep nano
motaz      20781    20085  0 17:01 pts/3      00:00:00 nano
motaz      20783    20076  0 17:01 pts/2      00:00:00 grep --color=auto nano
```

```
kill 20781
```

Then we can check process again to make sure it has been closed

```
motaz@motaz-lenovo-t460:~$ ps -ef | grep nano
motaz      20797    20076  0 17:03 pts/2      00:00:00 grep --color=auto nano
```

In other terminal window we will get this message after closing nano:

[code.sd](#)

```
Received SIGHUP or SIGTERM
```

Also `-9` to force process close can be used:

```
kill -9 20781
```

If you want to kill a process owned by another user or root, you need to use root privilege to execute it, by adding `sudo` command:

```
sudo kill -9 20781
```

**pkill** is used to kill process by name the same as **killall** command, but it doesn't tell anything if process has found or not, unless we add `-e` parameter will display killed process name and ID:

```
motaz@motaz-lenovo-t460:~$ pkill -e nano
nano killed (pid 26065)
```

## Switching commands to background

If some commands and tools requires time to execute, we can set it to work in background and resume work in the same terminal window. An example is using editor, or coping files that took long time. After we finish our interrupted work, we can go back to our first job, for example suppose that we are writing a text file using *nano*, and need some information from terminal then return back to that editor:

```
nano test.txt
```

Then start writing text, then click `CTRL+Z` to send it to background and return to terminal window:

```
motaz@motaz-lenovo-t460:~/linux$ nano test.txt
```

```
Use "fg" to return to nano.
```

```
[1]+  Stopped                  nano test.txt
```

we can check that *nano* process is still running by using `ps` command:

```
motaz@motaz-lenovo-t460:~/linux$ ps -ef | grep nano
motaz      14012    11460  0 09:11 pts/0    00:00:00 nano test.txt
motaz      14057    11460  0 09:12 pts/0    00:00:00 grep --color=auto nano
```

Note that the third column of process list is parent process id, in this case it is (**11460**), if we check it using `ps` command we will get the owner process of *nano*, which is `bash`, if we close `bash` (terminal) *nano* will be closed.

```
motaz@motaz-lenovo-t460:~/linux$ ps -n 11460
  PID TTY          STAT       TIME COMMAND
 11460 pts/0    Ss          0:00   bash
```

After finish we can return back to *nano* using **fg** command or **fg 1** using the ID that we got when we send it to background, this number is useful to run multiple commands or applications in background:

```
fg
```

## CPU and Memory usage

One of important system administration is to know the resources usage, first command is to use **nproc** to know how many processor cores in our hardware:

```
motaz@motaz-lenovo-t460:~$ nproc
4
```

In this example we get 4 cores.

To know memory information use command **free**:

```
motaz@motaz-lenovo-t460:~$ free
              total        used         free       shared  buff/cache   available
Mem:           7519976       3586680       1139396       470460       2793900       3181312
Swap:          12582904          11776       12571128
```

For values in Megabytes add **-m** parameter:

```
motaz@motaz-lenovo-t460:~$ free -m
              total        used         free       shared  buff/cache   available
Mem:           7343          3510          1104          459       2728          3099
Swap:          12287           11          12276
```

**top** command combines information about CPU usage, memory consumption and running processes, it runs continuously and displays current usage, we can quit it when pressing Q letter:

```
motaz@motaz-lenovo-t460:~$ top

top - 07:30:45 up 2:00, 1 user, load average: 0.21, 0.33, 0.56
Tasks: 296 total, 2 running, 294 sleeping, 0 stopped, 0 zombie
%Cpu(s):  9.5 us,  3.6 sy,  0.0 ni, 86.8 id,  0.2 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  7343.7 total,  959.5 free,  3592.4 used,  2791.9 buff/cache
MiB Swap: 12288.0 total, 12276.5 free,   11.5 used. 2981.5 avail Mem

   PID USER      PR  NI   VIRT   RES   SHR  S  %CPU  %MEM     TIME+ COMMAND
 13956 motaz    20   0 1299116 94980 56544 S  20.4   1.3   0:01.66 minetest
  2984 motaz    20   0 4862984 285044 114672 R  12.8   3.8  12:39.31 gnome-shell
  2829 motaz     9  -11 3578616 20176 15012 S   7.2   0.3   4:50.64 pulseaudio
  1846 couchdb  20   0 3739080 45680  9992 S   2.6   0.6   3:25.77 beam.smp
  3009 motaz    20   0 338516 74576 48392 S   2.6   1.0   4:00.92 Xwayland
  1845 asterisk -11   0 2220996 64540 31472 S   1.6   0.9   1:29.15 asterisk
```

[code.sd](#)

1438	mysql	20	0	2445796	397700	36652	S	1.3	5.3	1:14.47	mysqld
13904	motaz	20	0	12104	3816	3144	R	1.3	0.1	0:00.15	top
1376	redis	20	0	604824	526188	3512	S	0.7	7.0	0:27.07	redis-server
7892	motaz	20	0	4705016	125424	91820	S	0.7	1.7	0:44.42	FortiClient
12355	root	20	0	0	0	0	I	0.7	0.0	0:02.33	kworker/1:2-events
14	root	20	0	0	0	0	I	0.3	0.0	0:10.50	rcu_sched
22	root	20	0	0	0	0	S	0.3	0.0	0:29.19	ksoftirqd/1
1007	message+	20	0	9984	6296	3600	S	0.3	0.1	0:07.56	dbus-daemon
1097	motaz	20	0	1233992	9952	6748	S	0.3	0.1	0:00.43	temprature
1276	tomcat	20	0	5381712	721884	16320	S	0.3	9.6	0:39.74	java

An important line is the third line (%Cpu) which displays CPU utilization information:

**us:** means User processes usage percentage of CPU, in this sample 9.5%

**sy:** System processes or Kernel processes, in this sample it is 3.6%

**id:** Means idle CPU percentage, and in this sample it is 86.8%

Total summation of this line including other values Nice process, IO Waiting, Hardware and Software Interrupts, and CPU Steal time for virtual host result is 100%

Second two lines are for memory and swap consumption.

After that detailed processes sorted by most usage of CPU, as in our example top one is **minetest** with **20.4%** measured for one core, and gnome-shell as **12.8%** of one core. Note that measuring with one core doesn't mean that process is using single core, for example **minetest** could use more than one core, but the total of usage is 20.4, it could be 10.4 for one core and 10 for the second core. Sometime processes could use more than 100% of single core measure, for example Java in below example are using 322.6% which means at least it uses 4 cores, and this has increased total user processes CPU usage to 82.5%.

```
top - 07:59:13 up 2:28, 1 user, load average: 0.69, 0.76, 0.54
Tasks: 283 total, 1 running, 282 sleeping, 0 stopped, 0 zombie
%Cpu(s): 82.5 us, 3.0 sy, 0.0 ni, 14.3 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7343.7 total, 712.4 free, 3804.8 used, 2826.5 buff/cache
MiB Swap: 12288.0 total, 12269.7 free, 18.2 used. 2775.5 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
15492	motaz	20	0	4692168	317520	44600	S	<b>332.6</b>	4.2	0:10.56	<b>java</b>
2984	motaz	20	0	4896948	300888	115548	S	2.3	4.0	13:42.46	gnome-shell
2829	motaz	9	-11	3578944	20552	15160	S	1.3	0.3	5:31.04	pulseaudio
1846	couchdb	20	0	3739080	45852	9992	S	1.0	0.6	4:13.62	beam.smp
1438	mysql	20	0	2445796	397700	36652	S	0.7	5.3	1:32.48	mysqld
1845	asterisk	-11	0	2220996	63260	30192	S	0.7	0.8	1:50.64	asterisk

We can kill process while using top command when pressing K letter, then write process number from shown processes below, in our example we have selected process # **15941** which is top process itself :

```
MiB Mem : 7343.7 total, 990.3 free, 3513.1 used, 2840.4 buff/cache
MiB Swap: 12288.0 total, 12269.7 free, 18.2 used. 3058.3 avail Mem
PID to signal/kill [default pid = 2984]
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
```

[code.sd](#)

```

2984 motaz      20    0 4888904 300796 115612 S   6.6   4.0 14:00.44 gnome-shell
1846 couchdb    20    0 3739080 45852   9992 S   5.0   0.6  4:24.80 beam.smp
2829 motaz      9   -11 3578944 20552  15160 S   2.0   0.3  5:39.59 pulseaudio
1845 asterisk  -11   0 2220996 63260  30192 S   1.3   0.8  1:55.25 asterisk
1438 mysql      20    0 2445796 397700 36652 S   1.0   5.3  1:36.30 mysqld
15941 motaz      20    0 12104   4064   3272 R   0.3   0.1  0:00.08 top

```

Then select signal

```
Send pid 15941 signal [15/sigterm]
```

We can write (9) which means force closing

This will close **top** utility.

Another command for CPU usage is **uptime**, it shows current total CPU load, previous 5 minutes usage, and previous 15 minutes load as in below example:

```

motaz@motaz-lenovo-t460:~$ uptime
08:12:21 up 2:41, 1 user, load average: 1.22, 0.68, 0.59

```

Current is **1.22**, before 5 minutes was **0.68**, and before 15 minutes was **0.59**

Reading this values depends on processors core count, if value is 1, it means 100% of one core usage, for 4 processor cores value 4 means 100% of CPU usage, and it could exceed 100% indicating that there are waiting processes.

Note that *uptime* load average values are already part of *top* command:

```

top - 07:59:13 up 2:28, 1 user, load average: 0.69, 0.76, 0.54
Tasks: 283 total, 1 running, 282 sleeping, 0 stopped, 0 zombie
%Cpu(s): 82.5 us, 3.0 sy, 0.0 ni, 14.3 id, 0.2 wa, 0.0 hi, 0.0 si, 0.0 st
MiB Mem : 7343.7 total, 712.4 free, 3804.8 used, 2826.5 buff/cache
MiB Swap: 12288.0 total, 12269.7 free, 18.2 used. 2775.5 avail Mem

```

## Installing packages

Most of our used command tools are available by default in Ubuntu and most Linux distributions, but we always need to install additional packages and programs, such as database servers, web servers, and advanced tools, which are not installed by default, here is an example of not exist by default tool : **sensors** which displays hardware sensors values such as CPU temperature and indicate if fan is working and it's speed, battery voletage, etc:

```

motaz@motaz-VirtualBox:~$ sensors
Command 'sensors' not found, but can be installed with:
sudo apt install lm-sensors

```

[code.sd](#)

As suggested by terminal, we need to install **sensors** by typing **sudo apt install lm-sensors** package, but before that we need to update packages to refresh packages versions and locations local database, specially if this update doesn't happen for long time, without update we may get package not found error.

```
sudo apt update
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://sd.archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://sd.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages
[1,102 kB]
...
...
Metadata [472 B]
Fetched 897 kB in 58s (15.5 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
570 packages can be upgraded. Run 'apt list --upgradable' to see them.
```

After that we can install the package:

```
sudo apt install lm-sensors
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  fancontrol read-edid i2c-tools
The following NEW packages will be installed:
  lm-sensors
0 upgraded, 1 newly installed, 0 to remove and 570 not upgraded.
Need to get 91.0 kB of archives.
After this operation, 404 kB of additional disk space will be used.
Ign:1 http://sd.archive.ubuntu.com/ubuntu jammy/universe amd64 lm-sensors amd64
1:3.6.0-7ubuntu1
Get:1 http://sd.archive.ubuntu.com/ubuntu jammy/universe amd64 lm-sensors amd64
1:3.6.0-7ubuntu1 [91.0 kB]
Fetched 91.0 kB in 38s (2,376 B/s)
Selecting previously unselected package lm-sensors.
(Reading database ... 164282 files and directories currently installed.)
Preparing to unpack .../lm-sensors_1%3a3.6.0-7ubuntu1_amd64.deb ...
Unpacking lm-sensors (1:3.6.0-7ubuntu1) ...
Setting up lm-sensors (1:3.6.0-7ubuntu1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/lm-sensors.service →
/lib/systemd/system/lm-sensors.service.
Processing triggers for man-db (2.10.2-1) ..
```

Then we can run **sensors** tool:

```
motaz@motaz-lenovo-t460:~$ sensors
iwlwifi_1-virtual-0
Adapter: Virtual device
temp1:          +38.0°C

pch_skylake-virtual-0
Adapter: Virtual device
```

[code.sd](#)



```
temp1:          +39.5°C

BAT0-acpi-0
Adapter: ACPI interface
in0:            12.14 V

coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +42.0°C (high = +100.0°C, crit = +100.0°C)
Core 0:        +40.0°C (high = +100.0°C, crit = +100.0°C)
Core 1:        +41.0°C (high = +100.0°C, crit = +100.0°C)

thinkpad-isa-0000
Adapter: ISA adapter
fan1:           0 RPM
CPU:            +42.0°C
GPU:            N/A
temp3:          +0.0°C
temp4:          +0.0°C
temp5:          +0.0°C
temp6:          +0.0°C
temp7:          +0.0°C
temp8:          +0.0°C

BAT1-acpi-0
Adapter: ACPI interface
in0:            10.68 V

acpitz-acpi-0
Adapter: ACPI interface
temp1:          +42.0°C (crit = +128.0°C)
```

## Files Permissions

Users can create files, modify them, read, and delete by default for their home directory, each user has a directory in /home directory

If we list our sample directory (*linux*) in our home directory we will get below information:

```
motaz@motaz-lenovo-t460:~/linux$ ls -lh
total 48K
-rw-r--r-- 1 motaz motaz 64 Dec 31 10:01 log.txt
-rw-r--r-- 1 motaz motaz 200 Jan 1 06:55 memory.log
-rw-r--r-- 1 motaz motaz 1.4K Dec 31 09:58 memory.txt
-rw----- 1 motaz motaz 5 Jan 2 13:32 nano.save
-rw-r--r-- 1 motaz motaz 73 Dec 31 08:06 newfile.txt
-rw----- 1 motaz motaz 850 Jan 2 13:30 nohup.out
-rw-r--r-- 1 motaz motaz 1.3K Dec 31 09:57 root.lst
-rw-r--r-- 1 motaz motaz 73 Dec 31 09:22 secondcopy.txt
-rw-r--r-- 1 motaz motaz 12 Jan 2 09:11 second.text
drwxrwxr-x 2 motaz motaz 4.0K Jan 2 15:55 subfolder
-rw-r--r-- 1 motaz motaz 5 Jan 2 09:15 test.txt
```

We will find in first column in detailed files list below permission information for each file:

```
-rw-r--r--
```

It consists of four parts:

```
- rw- r-- r--
```

First part is one character indicate wither it is normal file or directory, for directories it shows (**d**), and for normal files it shows (-)

Second part is permission for file or directory **owner** which is (-**rw**)

Third part is permission for **group** (**r--**),

Fourth part is permissions for **others**, which are not owner nor group (**r--**)

**r** means read permission

**w** means write permission

**x** means execute permission

To change file/directory permission use **chmod** command, for example if we want to prevent write, delete and modify permission for **log.txt** file, execute below commands

```
motaz@motaz-lenovo-t460:~/linux$ chmod -w log.txt
motaz@motaz-lenovo-t460:~/linux$ ls -lh log.txt
-r--r--r-- 1 motaz motaz 64 Dec 31 10:01 log.txt
```

Note that (w) permission has been removed, if we tried to edit it using nano we will get below message:

```
[ File 'log.txt' is unwritable ]
```

Note that this file were writable only for owner, if we want to make it writable for owner and group we can do it using below command:

```
motaz@motaz-lenovo-t460:~/linux$ chmod ug+w log.txt
motaz@motaz-lenovo-t460:~/linux$ ls -lh log.txt
-rw-rw-r-- 1 motaz motaz 64 Dec 31 10:01 log.txt
```

u means for owner user, and g for group, combination ug means for both, remaining is other (o), we can make it wriable for others also using below command:

```
motaz@motaz-lenovo-t460:~/linux$ chmod o+w log.txt
motaz@motaz-lenovo-t460:~/linux$ ls -lh log.txt
-rw-rw-rw- 1 motaz motaz 64 Dec 31 10:01 log.txt
```

To revoke write permission from all we can use (-) instead of (+):

```
motaz@motaz-lenovo-t460:~/linux$ chmod oug-w log.txt
motaz@motaz-lenovo-t460:~/linux$ ls -lh log.txt
-r--r--r-- 1 motaz motaz 64 Dec 31 10:01 log.txt
```

Execute permission is used for shell script files, or files that contains commands. Let us first write simple script file:

```
nano first.sh
```

put below commands in the file and then save and exist:

```
echo "Hello"
echo "Date and time is"
date
```

Then give execute permission to that file:

```
motaz@motaz-lenovo-t460:~/linux$ chmod +x first.sh
motaz@motaz-lenovo-t460:~/linux$ ls -lh first.sh
-rwxrwxr-x 1 motaz motaz 42 Jan  2 16:15 first.sh
```

To run it prefix it with ./

```
motaz@motaz-lenovo-t460:~/linux$ ./first.sh
Hello
Date and time is
Tue 02 Jan 2024 04:17:27 PM CAT
```

We still can run this script file without execute permission using (**sh**) command (shell command), or (**bash**):

```
motaz@motaz-lenovo-t460:~/linux$ sh first.sh
```

[code.sd](#)

```
Hello
Date and time is
Tue 02 Jan 2024 04:18:17 PM CAT
```

Second part of files and directories permission is the ownership and groups of that files and directories. By default user who creates file or directory becomes the owner, and his/her default group will be the group of that file or directory, later it could be changed using **chown** command. In most cases taking or granting ownership from/to another user or root user requires higher privilege by using **sudo**, example granting *first.sh* script file to root user:

```
motaz@motaz-lenovo-t460:~/linux$ sudo chown root first.sh
motaz@motaz-lenovo-t460:~/linux$ ls -lh first.sh
-rwxrwxr-x 1 root motaz 42 Jan  2 16:15 first.sh
```

Also group could be changed along with changing owner:

```
motaz@motaz-lenovo-t460:~/linux$ sudo chown root:www-data first.sh
motaz@motaz-lenovo-t460:~/linux$ ls -lh first.sh
-rwxrwxr-x 1 root www-data 42 Jan  2 16:15 first.sh
```

Or to change group separately using **chgrp** command:

```
motaz@motaz-lenovo-t460:~/linux$ sudo chgrp www-data first.sh
```

Directory owner and group also could be changed using the same previous commands, but this will not change it's contained files and directories, to change all directory files and sub-directories recursively add the parameter **-R**:

```
motaz@motaz-lenovo-t460:~$ sudo chgrp www-data linux/ -R
motaz@motaz-lenovo-t460:~$ ls -lh linux/
total 48K
-rwxrwxr-x 1 root www-data 42 Jan  2 16:15 first.sh
-r--r--r-- 1 motaz www-data 64 Dec 31 10:01 log.txt
-rw-r--r-- 1 motaz www-data 200 Jan  1 06:55 memory.log
-rw-r--r-- 1 motaz www-data 1.4K Dec 31 09:58 memory.txt
-rw----- 1 motaz www-data  5 Jan  2 13:32 nano.save
-rw-r--r-- 1 motaz www-data  73 Dec 31 08:06 newfile.txt
-rw----- 1 motaz www-data 850 Jan  2 13:30 nohup.out
-rw-r--r-- 1 motaz www-data 1.3K Dec 31 09:57 root.lst
-rw-r--r-- 1 motaz www-data  73 Dec 31 09:22 secondcopy.txt
-rw-r--r-- 1 motaz www-data  12 Jan  2 09:11 second.text
drwxrwxr-x 2 motaz www-data 4.0K Jan  2 15:55 subfolder
--wxr--r-- 1 motaz www-data  5 Jan  2 09:15 test.txt
```

## Files information

There are many commands and tools in Linux for files information, search and comparison. First command is file command that tells file type and show some details, for example when we apply it to one of our text file we get below result:

```
motaz@motaz-lenovo-t460:~/linux$ file first.sh
first.sh: ASCII text
```

Telling that *first.sh* is ASCII text file.

If we check binary file such as bash we will get below information:

```
motaz@motaz-lenovo-t460:~/linux$ file /bin/bash
/bin/bash: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked,
interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=2a9f157890930ced4c3ad0e74fc1b1b84aad71e6, for GNU/Linux 3.2.0, stripped
```

It tells that this is 64-bit executable file and uses dynamic libraries.

Here is another example for image file:

```
motaz@motaz-lenovo-t460:~/linux$ file ~/Pictures/Books.jpg
/home/motaz/Pictures/Books.jpg: JPEG image data, JFIF standard 1.01, resolution (DPI),
density 300x300, segment length 16, Exif Standard: [TIFF image data, little-endian,
direntries=16, manufacturer=HUAWEI, model=CRO-U00, orientation=upper-left,
xresolution=222, yresolution=230, resolutionunit=2, software=GIMP 2.10.28,
datetime=2022:04:09 16:03:28], progressive, precision 8, 800x600, components 3
```

And here are examples for Video and Audio files:

```
motaz@motaz-lenovo-t460:~/linux$ file ~/Videos/Attari.mp4
/home/motaz/Videos/Attari.mp4: ISO Media, MP4 v2 [ISO 14496-14]

motaz@motaz-lenovo-t460:~/linux$ file ~/Videos/q-111-1645617108.0.wav
/home/motaz/Videos/q-111-1645617108.0.wav: RIFF (little-endian) data, WAVE audio,
Microsoft PCM, 16 bit, mono 8000 Hz
```

Another information for files is checksum, to get a signature from file contents to check if two files in different location has the same contents or version or different, for example if you have multiple binary files in a server and in different server or locally, we can generate checksum for them to see if they are identical or not. Also if we download a large file from Net, for example Linux ISO file, they put hash code for that file, for example sha1, and after we download it we can check sha1 hash locally to make sure that both files are identical and file has been downloaded without any modification.

This is an example of **md5sum** command:

```
motaz@motaz-lenovo-t460:~/linux$ md5sum first.sh
a641d0f2dfbc541ab22f56d523439686 first.sh
```

This returns MD5 hash of text file, also it could be done for binary file:

[code.sd](#)

```
motaz@motaz-lenovo-t460:~/linux$ md5sum /bin/bash
23c415748ff840b296d0b93f98649dec /bin/bash
```

And here is **sha1sum** command:

```
motaz@motaz-lenovo-t460:~/linux$ sha1sum first.sh
c947aa48826a5dc7d3a6053f965f5229886103c7 first.sh

motaz@motaz-lenovo-t460:~/linux$ sha1sum /bin/bash
439667f622b84ecb9f381be93cc9139f83a92f66 /bin/bash
```

To compare two text files in contents use **diff** command. For example we have copied **first.sh** file into **second.sh** and do changes in **second.sh** contents then compare them

```
motaz@motaz-lenovo-t460:~/linux$ cp first.sh second.sh
motaz@motaz-lenovo-t460:~/linux$ nano second.sh
motaz@motaz-lenovo-t460:~/linux$ diff first.sh second.sh
1c1
< echo "Hello"
---
> echo "Hello From Linux"
```

## Connecting to remote server

This books commands and configurations are mostly useful on server side, because in client side Linux users may use GUI more than command line, but servers, such as Ubuntu server it has no GUI, so that the standard way to use it is to use terminal and commands.

First command is **ssh**, which is used to connect to remote ssh server using ssh client using port 22, so that we have to make sure we can access the remote server via port 22.

We can access server by it's IP or name, if we don't have DNS name for that server and we use it frequently, we can add local DNS name for it locally in **/etc/hosts** file

Example:

```
ssh user@remote-server.com
```

We will be prompt for user password on that remote server and we will be logged into home directory of that server, and you will get many information such as last login time and IP, and if system requires update:

```
motaz@motaz-lenovo-t460:~$ ssh user@remote-server.com
Welcome to Ubuntu 20.04.6 LTS (GNU/Linux 5.4.0-152-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage
```

```
System information as of Tue 23 Jan 2024 06:55:24 AM CAT

System load:  0.0                Processes:            156
Usage of /:   20.5% of 37.23GB    Users logged in:    0

Expanded Security Maintenance for Applications is not enabled.

48 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

6 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '22.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Sun Jan 21 07:50:57 2024 from 197.252.214.146
code@remote-server:~$
```

Then we can do execute and manage server according to our privilege. At the end we can close that remote terminal window using **exit** command.

```
user@remote-server:~$ exit
logout
Connection to remote-server closed.
motaz@motaz-lenovo-t460:~$
```

As we have mentioned previously we can set a name locally in our computer to our remote server IP if it does not have a DNS name by editing **/etc/hosts** file:

```
motaz@motaz-lenovo-t460:~$ sudo nano /etc/hosts
```

then add name and IP of that server and save the file, then we can use that name in ssh or in browser.

```
212.0.20.13      remote-server.com
```

Adding names in hosts file is useful for local naming, for example we can add our local router or WiFi access point, for example if that router has the IP 192.168.1.1 we can add it in **/etc/hosts** file as:

```
192.168.1.1     home-router.local
```

Then we can use that name instead of IP, the benefit is that we can use (tab) key to complete the name, for example if we type **ping hom** and press tab, the name will be completed

```
motaz@motaz-lenovo-t460:~$ ping home-router.local
PING home-router.local (192.168.1.1) 56(84) bytes of data.
```

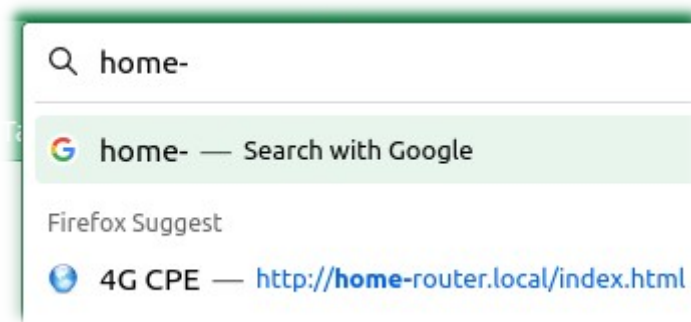
[code.sd](https://code.sd)

```
64 bytes from home-router.local (192.168.1.1): icmp_seq=1 ttl=64 time=6.17 ms
64 bytes from home-router.local (192.168.1.1): icmp_seq=2 ttl=64 time=1.92 ms
64 bytes from home-router.local (192.168.1.1): icmp_seq=3 ttl=64 time=3.23 ms
^C
--- home-router.local ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 1.921/3.772/6.165/1.774 ms
```

We can have multiple names for the same IP, example:

```
192.168.1.1    home-router.local  tplink-router.com
```

Also in browser we can use that name <http://home-router.local> in address and typing **home-** in address box of browser will display suggestion for the DNS name to access admin portal of router.



Second command is **scp** (secure copy) to copy files from local computer to remote server or vice versa:

we have to add colon at the end of server name or IP, then location, and make sure the user has write access in that location, and we use home directory (~) for provided user, and we will be prompted for the password:

```
motaz@motaz-lenovo-t460:~/linux$ scp log.txt user@remote-server.com:~ log.txt
user@remote-server.com's password:
100% 64 0.3KB/s 00:00
motaz@motaz-lenovo-t460:~/linux$
```

If we own that remote server, we could send ssh public key to our user in that server to login and copy files without password prompt, in spite of that it will reduce security but we could use scp command for scheduled jobs to backup files from or to remote servers, here are the steps:

First we need to run **ssh-keygen** to generate public and private key for ssh, we can press enter as empty input to use defaults:

[code.sd](#)



```

motaz@motaz-VirtualBox:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/motaz/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/motaz/.ssh/id_rsa
Your public key has been saved in /home/motaz/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:EB0KHNvqaEuIh20HZ/mIpTMe9LfsABZxFfnrzTeyus8 motaz@motaz-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|  ..+o++..      |
|    +.o.        |
|  . . +.       |
|   . o ..      |
|  *   S.o       |
|. * % o .      |
|+ & * o. o     |
| * +* + .+ o   |
|   .+ o+E+ .o+  |
+-----[SHA256]-----+

```

After that we can copy public key to remote server using **ssh-copy-id** command:

```

motaz@motaz-VirtualBox:~/.ssh$ ssh-copy-id -i user@remote-server.com

```

This should be the last password prompt, next ssh login or copying files using scp to that server using that user will not require password, so that we could put it in a crontab job to do backup from or to that server

## Crontab jobs

As we have prepared scp to do files copy without password prompt, we have mentioned it is used by crontab scheduler, which enables jobs or scripts to be executed in a specific time, either regularly or once.

To show current crontab type **crontab -l**:

in case of no crontab schedule exist you will get this result:

```
motaz@motaz-VirtualBox:~$ crontab -l
no crontab for motaz
```

To add or edit crontab schedule for current user use **-e** parameter, and for the first time a default editor choice will appear, in my case I always use nano:

```
motaz@motaz-VirtualBox:~$ crontab -e
no crontab for motaz - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          <---- easiest
 2. /usr/bin/vim.tiny
 3. /bin/ed

Choose 1-3 [1]: 1
```

We will see below information and you can add new schedule at the end of file:

```
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * /home/motaz/linux/first.sh
```

Note that in last comments there is a description how to use it, first column is for **minutes**, then **Hours**, **Day** of Month, **Month**, and **Day** of Week. \* means any, as above example this script will run at any minute, any time, any month, etc, and below example:

```
1 * * * * /home/motaz/linux/first.sh
```

This will run the script at minute (1) of every hour, at every day,

and below one will run script at 3:01 am every day

```
1 3 * * * /home/motaz/linux/first.sh
```

and below one will run every two minutes:

```
*/2 * * * * /home/motaz/linux/first.sh
```

Note that this crontab schedule will run in current user privilege, if we need to run a command or script using root privilege then we need to edit sudo crontab instead, which is different configuration than current user privilege:

```
sudo crontab -l
```

To run script or command at system boot, use **@reboot** tag in crontab :

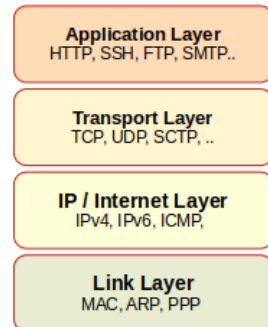
```
crontab -e  
  
# For more information see the manual pages of crontab(5) and cron(8)  
#  
# m h dom mon dow   command  
  
@reboot /home/motaz/linux/first.sh
```

## Networking commands

Linux has many commands for networking, we will start with IP and networking devices information and configuration.

First command is: `ip`, which can be used with many parameters for different use:

**`ip link` or `ip l`**: to show available networking interfaces, including Loopback, Ethernet, Wireless, and Bluetooth, but in case of Bluetooth networking logical interface, it will show the device if it is On.



```
motaz@motaz-lenovo-t460:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state
DOWN mode DEFAULT group default qlen 1000
    link/ether ff:ff:aa:00:cc:10 brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
mode DORMANT group default qlen 1000
    link/ether ff:ee:bb:dd:d1:22 brd ff:ff:ff:ff:ff:ff
```

It shows three devices: (**lo**) Loopback which is used for localhost networking access, Ethernet (**enp0s31f6**) and Wireless interface device (**wlp4s0**), also it shows MAC addresses for that device, which is unique for physical devices, but I have changed the actual values for security concerns. Note that IPs are not shown using that command.

**`ip address`, or `ip a`**: it shows interfaces and IPs:

```
motaz@motaz-lenovo-t460:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    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: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state
DOWN group default qlen 1000
    link/ether ff:ff:aa:00:cc:10 brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether ff:ee:bb:dd:d1:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.207/24 brd 192.168.10.255 scope global dynamic noprefixroute
wlp4s0
        valid_lft 2684sec preferred_lft 2684sec
    inet6 fe80::1965:26a9:d5ca:a6c4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

We can add additional IP on one of already configured interfaces, for example in Wireless interface using **ip address add** command, and this command requires higher privilege using *sudo*, after that we queried again interfaces to see the new IP 192.168.10.298:

```
motaz@motaz-lenovo-t460:~$ sudo ip address add 192.168.10.208/24 dev wlp4s0
[sudo] password for motaz:
motaz@motaz-lenovo-t460:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    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: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state
DOWN group default qlen 1000
    link/ether ff:ff:aa:00:cc:10 brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether ff:ee:bb:dd:d1:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.207/24 brd 192.168.10.255 scope global dynamic noprefixroute
wlp4s0
        valid_lft 3459sec preferred_lft 3459sec
    inet 192.168.10.208/24 scope global secondary wlp4s0
        valid_lft forever preferred_lft forever
    inet6 fe80::1965:26a9:d5ca:a6c4/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

**/24** means the net-mask of **255.255.255.0**

This IP will be added temporarily, if restart to machine happens or interface has been disabled and enabled again it will be removed.

We can ping this new IP address to make sure it has been configured correctly and interface device is up:

```
motaz@motaz-lenovo-t460:~$ ping 192.168.10.208
PING 192.168.10.208 (192.168.10.208) 56(84) bytes of data.
64 bytes from 192.168.10.208: icmp_seq=1 ttl=64 time=0.054 ms
64 bytes from 192.168.10.208: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from 192.168.10.208: icmp_seq=3 ttl=64 time=0.086 ms
^C
--- 192.168.10.208 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.054/0.065/0.086/0.014 ms
```

To remove this new IP use **ip address del** command:

```
motaz@motaz-lenovo-t460:~$ sudo ip address del 192.168.10.208/24 dev wlp4s0
```

Example of adding IPv6 address:

```
motaz@motaz-lenovo-t460:~$ sudo ip -6 address add fd00::11:cc:5/7 dev wlp4s0
motaz@motaz-lenovo-t460:~$ ip a
```

[code.sd](#)

```

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    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: enp0s31f6: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state
DOWN group default qlen 1000
    link/ether ff:ff:aa:00:cc:10 brd ff:ff:ff:ff:ff:ff
3: wlp4s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP
group default qlen 1000
    link/ether ff:ee:bb:dd:d1:22 brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.207/24 brd 192.168.10.255 scope global noprefixroute wlp4s0
        valid_lft forever preferred_lft forever
    inet6 fd00::11:cc:5/7 scope global
        valid_lft forever preferred_lft forever

```

We can ping that IP using **ping6**:

```

motaz@motaz-lenovo-t460:~$ ping6 fd00::11:cc:5
PING fd00::11:cc:5(fd00::11:cc:5) 56 data bytes
64 bytes from fd00::11:cc:5: icmp_seq=1 ttl=64 time=0.051 ms
64 bytes from fd00::11:cc:5: icmp_seq=2 ttl=64 time=0.100 ms
64 bytes from fd00::11:cc:5: icmp_seq=3 ttl=64 time=0.073 ms
64 bytes from fd00::11:cc:5: icmp_seq=4 ttl=64 time=0.075 ms
^C
--- fd00::11:cc:5 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3071ms
rtt min/avg/max/mdev = 0.051/0.074/0.100/0.017 ms

```

## Static and Automatic IPs

As a client machine, default configuration for any interface is Automatic DHCP, such as Wifi connection to router, in this case router will assign dynamic IPs for connected clients in the same range of router, range of DHCP for such routers are always private range: 192.168.x.x, 10.x.x.x, 172.16, 172.17.x.x, until 172.31.x.x.

For servers and manually configured networking, when we need to configure IP manually, we have to assign that IP in networking configuration file, for Ubuntu it is located in etc/netplan, configuration extension is **.yaml**, example:

```

cat /etc/netplan/01-network-manager-all.yaml

network:
  ethernets:
    ens160:
      addresses:
        - 192.168.20.10/24
      nameservers:

```

[code.sd](https://code.sd)

```
addresses:
- 8.8.8.8
search:
- ourdomain.net
routes:
- to: default
  via: 192.168.20.1
version: 2
```

Another example:

```
cat /etc/netplan/50-cloud-init.yaml
network:
  ethernets:
    ens160:
      addresses:
        - 172.20.1.12/24
      gateway4: 172.20.1.1
      nameservers:
        addresses:
          - 172.20.1.4
version: 2
```

And here is an example of IPv6 IP configuration (note that actual Ips and MAC has been randomly modified for security reason, because it is a public IP):

```
cat /etc/netplan/50-cloud-init.yaml
network:
  version: 2
  ethernets:
    eth0:
      addresses:
        - 4b20:20331:8ccc:1122::1/64
      dhcp4: true
      match:
        macaddress: 16:00:02:12:31:ff
      nameservers:
        addresses:
          - 4b20:aff:ffff::add:1
          - 4b20:aff:ffff::add:8
      routes:
        - on-link: true
          to: ::/0
          via: fe80::1
      set-name: eth0
```

If you done any change in netplan file, we need to apply it using below file:

```
sudo netplan apply
```

[code.sd](#)

Note that *netplan* file is restricted about indentation, it parses items and its properties according to that, so that any bad indentation could result on error while applying modified *netplan* file.

## Routing and gateways

When using only one IP (rather than loopback IP) or multiple IPs in the same range, we always use one router, but when using multiple Ips in different subnets, multiple routers has to be used, and one could be default route. Here is an example of one IP routing table using **ip route** command or abbreviate it as **ip r**:

```
motaz@motaz-lenovo-t460:~$ ip route
default via 192.168.44.1 dev bnep0 proto dhcp metric 750
169.254.0.0/16 dev bnep0 scope link metric 1000
192.168.44.0/24 dev bnep0 proto kernel scope link src 192.168.44.211 metric 750
```

All traffic in this example will go through the router: **192.168.44.1** which represents Access point or mobile device.

Here is an example of two active interfaces containing two IPs:

```
motaz@motaz-lenovo-t460:~$ ip r
default via 192.168.215.46 dev wlp4s0 proto dhcp metric 600
default via 192.168.44.1 dev bnep0 proto dhcp metric 750
169.254.0.0/16 dev bnep0 scope link metric 1000
192.168.44.0/24 dev bnep0 proto kernel scope link src 192.168.44.211 metric 750
192.168.215.0/24 dev wlp4s0 proto kernel scope link src 192.168.215.207 metric 600
```

Note that we have two subnets:

1. **192.168.44.0/24** with gateway **192.168.44.1** and
2. **192.168.215.0/24** with gateway **192.168.215.46**

To force networking traffic to go through specific gateway, then we have to remove not wanted and keep only one default gateway, for example we want to keep 192.168.44.1, we have to remove other one, this requires *sudo* privilege:

```
motaz@motaz-lenovo-t460:~$ sudo ip r del default via 192.168.215.46
[sudo] password for motaz:
```

Then we show routing table again to find that default gateway has removed:

```
motaz@motaz-lenovo-t460:~$ ip r
default via 192.168.44.1 dev bnep0 proto dhcp metric 750
169.254.0.0/16 dev bnep0 scope link metric 1000
192.168.44.0/24 dev bnep0 proto kernel scope link src 192.168.44.211 metric 750
192.168.215.0/24 dev wlp4s0 proto kernel scope link src 192.168.215.207 metric 600
```



If we want traffic to specific subnet(s) go through that removed gateway, for example any traffic to **172.20.0.0** to go through second gateway we can add it as:

```
motaz@motaz-lenovo-t460:~$ sudo ip r add 172.20.0.0/16 via 192.168.215.46
```

And then show modified routing table:

```
motaz@motaz-lenovo-t460:~$ ip r
default via 192.168.44.1 dev bnep0 proto dhcp metric 750
169.254.0.0/16 dev bnep0 scope link metric 1000
172.20.0.0/16 via 192.168.215.46 dev wlp4s0
192.168.44.0/24 dev bnep0 proto kernel scope link src 192.168.44.211 metric 750
192.168.215.0/24 dev wlp4s0 proto kernel scope link src 192.168.215.207 metric 600
```

Now all traffic that matches the network **172.20.0.0/16** will go through that second gateway, and any other traffic will go through default gateway.

This is useful if we connected to VPN or private network and we want to access network using specific gateway.

Using ip command is temporary, for permanent gateways configurations we have to configure it in **netplan** filename

## Networking troubleshooting

There are many tools for troubleshooting networking problem, starting from ping to make sure that desired IP is accessible, also to make sure there is Internet connectivity and it's response time, we can **ping** 1.1.1.1 or Google DNS 8.8.8.8:

```
motaz@motaz-lenovo-t460:~$ ping 1.1.1.1
PING 1.1.1.1 (1.1.1.1) 56(84) bytes of data.
64 bytes from 1.1.1.1: icmp_seq=1 ttl=53 time=464 ms
64 bytes from 1.1.1.1: icmp_seq=2 ttl=53 time=159 ms
64 bytes from 1.1.1.1: icmp_seq=3 ttl=53 time=180 ms
64 bytes from 1.1.1.1: icmp_seq=4 ttl=53 time=159 ms
^C
--- 1.1.1.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 158.780/240.411/463.907/129.306 ms
```

Also we can ping public servers by name such as google.com to make sure that DNS is configured correctly:

```
motaz@motaz-lenovo-t460:~$ ping google.com
PING google.com (142.250.27.138) 56(84) bytes of data.
64 bytes from ra-in-f138.1e100.net (142.250.27.138): icmp_seq=1 ttl=102 time=329 ms
64 bytes from ra-in-f138.1e100.net (142.250.27.138): icmp_seq=2 ttl=102 time=217 ms
```

[code.sd](#)

```
64 bytes from ra-in-f138.1e100.net (142.250.27.138): icmp_seq=3 ttl=102 time=191
ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 190.927/245.474/328.693/59.785 ms
```

If ping to public IP such as 1.1.1.1 is replying while pinging to public name (such as google.com) is not replying, that means we need to check DNS configuration in our router, or in our local IP configuration in *netplan*, or configure it temporary in */etc/resolve.conf*, but this will be changed when reconnecting.

We can add global Google DNS server 8.8.8.8 as DNS:

```
motaz@motaz-lenovo-t460:~$ sudo nano /etc/resolv.conf
[sudo] password for motaz:
nameserver 8.8.8.8
```

Then ping again to make sure that problem was in DNS

Another important network and network services troubleshooting is to access service ports to make sure that they are running and it is accessible through firewall for remote services.

Example of that tracing is to check web server (Apache or Nginx) on remote server on port 80:

1. using **nc** command (NetCat):

```
motaz@motaz-VirtualBox:~$ nc -v 10.0.2.2 80
Connection to 10.0.2.2 80 port [tcp/http] succeeded!
^C
```

It tells us the application layer type for this port, and in our case it is [tcp/http]

2. using **telnet**:

```
motaz@motaz-VirtualBox:~$ telnet 10.0.2.2 80
Trying 10.0.2.2...
Connected to 10.0.2.2.
Escape character is '^]'.
^CConnection closed by foreign host.
```

It establishes connection to server and wait to send request according to protocol, we can close it using CTRL+C

And here are examples of trying to access not opened port, or port that has no listening service:

```
motaz@motaz-VirtualBox:~$ telnet 10.0.2.2 50
Trying 10.0.2.2...
telnet: Unable to connect to remote host: Connection refused
```

[code.sd](#)

```
motaz@motaz-VirtualBox:~$ nc -v 10.0.2.2 50
nc: connect to 10.0.2.2 port 50 (tcp) failed: Connection refused
```

In both cases the utility tells us that it can access the server but port couldn't be accessed, which means there is no listening service or application on that port.

Another test is to deny access using firewall (we will do examples later) for our client and do testing again trying to access ports, in this time the tool will hang up waiting for long time until timeout happen, or close it using CTRL+C

```
motaz@motaz-VirtualBox:~$ telnet 10.0.2.2 80
Trying 10.0.2.2...
^C
motaz@motaz-VirtualBox:~$ nc -v 10.0.2.2 50
^C
motaz@motaz-VirtualBox:~$ nc -v 10.0.2.2 80
^C
```

and where if we wait for long time we will get timeout

```
motaz@motaz-VirtualBox:~$ nc -v 10.0.2.2 80
nc: connect to 10.0.2.2 port 80 (tcp) failed: Connection timed out
```

Note that difference between Connection refused in case of application not listening on that port, and connection timeout.

Here is another example of unreachable or not exist IP, and waiting about 2 minutes for timeout:

```
motaz@motaz-VirtualBox:~$ nc -v 172.10.200.100 80
nc: connect to 172.10.200.100 port 80 (tcp) failed: Connection timed out
```

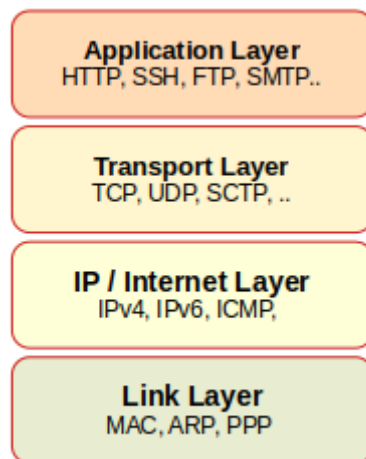
Here an example of closing web server (Nginx) and trying to access the port:

```
motaz@motaz-lenovo-t460:~$ sudo service nginx stop
motaz@motaz-lenovo-t460:~$ nc -v localhost 80
nc: connect to localhost port 80 (tcp) failed: Connection refused
```

As a summary:

- **Connection refused** means that the IP of server is accessible, but **port is not opened** due to no listening service or application
- **Connection timeout** means that remote **IP is not reachable (as networking)** or port is closed by a firewall.

[code.sd](#)



UDP ports: UDP as a transport protocol is different than TCP, by default **nc** and **telnet** are connecting to TCP port, and telnet is using TCP transport only, so that we have to use nc in our testing for UDP ports. We have to add -u parameter when using UDP protocol with nc utility:

```

motaz@motaz-lenovo-t460:~$ nc -vu localhost 5060
Connection to localhost 5060 port [udp/sip] succeeded!
^C
  
```

here is an example of connecting to Asterisk server which listening on UDP port 5060, and protocol name as indicated above is SIP

In case of no service listening to that port, it will close immediately and tells nothing:

```

motaz@motaz-lenovo-t460:~$ nc -vu localhost 5090
motaz@motaz-lenovo-t460:~$
  
```

Another network security troubleshooting is to check which ports are opened in specific server and for which applications, in this case we added -z parameter and ports range to scan them:

```

motaz@motaz-lenovo-t460:~$ nc -zv localhost 20-80
nc: connect to localhost port 20 (tcp) failed: Connection refused
Connection to localhost 21 port [tcp/ftp] succeeded!
Connection to localhost 22 port [tcp/ssh] succeeded!
nc: connect to localhost port 23 (tcp) failed: Connection refused
nc: connect to localhost port 24 (tcp) failed: Connection refused
Connection to localhost 25 port [tcp/smtp] succeeded!
nc: connect to localhost port 26 (tcp) failed: Connection refused
...
...
Connection to localhost 80 port [tcp/http] succeeded!
  
```

It shows that ports: 21, 22, 25, and 80 are open on that range and indicates transport and application protocols

Here is another example of scanning UDP ports:

```
motaz@motaz-lenovo-t460:~$ nc -zvu localhost 100-6000
Connection to localhost 123 port [udp/ntp] succeeded!
Connection to localhost 161 port [udp/snmp] succeeded!
Connection to localhost 631 port [udp/*] succeeded!
Connection to localhost 4569 port [udp/iax] succeeded!
Connection to localhost 5060 port [udp/sip] succeeded!
```

**nc** also could be used as socket server listening on specific port, but we need to make sure that there is no other application is running on that port on the same machine.

```
motaz@motaz-lenovo-t460:~$ nc -l 1024
```

It will keep listening on port 1024, note that less than 1024 port number requires root permission, but higher (from 1024 until 65536)

Then we keep that command running and open new terminal window and start **nc** as client and connect to that server, after connection we can send any text to the server:

```
motaz@motaz-lenovo-t460:~$ nc localhost 1024
Hello
```

This will be sent and displayed in other window in which **nc** is listening:

```
motaz@motaz-lenovo-t460:~$ nc -l 1024
Hello
Hi
```

We can reply back from server side, this test if done through different servers (**nc** listening on a server and other **nc** client connects from second server) means there is a network accessibility and this port is not closed

Also UDP protocol could be used in this socket connection in both sides:

In server side:

```
motaz@motaz-lenovo-t460:~$ nc -lu 1024
```

In client side:

```
motaz@motaz-lenovo-t460:~$ nc -u localhost 1024
```

Another tool for troubleshooting http (Application layer) is **curl**, but it is not installed by default, we need to install it first using:

```
motaz@motaz-lenovo-t460:~$ sudo apt-get install curl
```

**curl** can be used to test and call web services and web pages or web servers:

[code.sd](#)

For example calling IP two Country name web service:

```
motaz@motaz-lenovo-t460:~$ curl http://ip2c.org/212.0.1.1
1;DE;DEU;Germany
```

Another web service to display my current IP and country:

```
motaz@motaz-lenovo-t460:~$ curl http://services.codesoft.sd/iplocation/myip
IP=197.252.219.122
Country=Sudan
Code=SD
```

and this is for testing web server on current server, in case of using Nginx server:

```
motaz@motaz-lenovo-t460:~$ curl http://localhost
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h2>Nginx, it Works!</h2>
<div>
<img src='nginx.png'>

<a href="quran">Quran</a>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p>
</div>
<em>Thank you for using nginx.</em></p>
</body>
</html>
```

To display page or web service with header information add (-i) parameter:

```
motaz@motaz-lenovo-t460:~$ curl -i http://services.codesoft.sd/iplocation/myip
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Thu, 01 Feb 2024 05:30:55 GMT
Content-Type: text/plain; charset=utf-8
```

[code.sd](http://code.sd)

```
Content-Length: 41
Connection: keep-alive
Access-Control-Allow-Origin: *

IP=197.252.219.122
Country=Sudan
Code=SD
```

Similar to curl, is wget, which downloads contents as file, it is also used to download files from web servers:

```
motaz@motaz-VirtualBox:~$ wget http://10.0.2.2
--2024-02-01 07:32:33-- http://10.0.2.2/
Connecting to 10.0.2.2:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 674 [text/html]
Saving to: 'index.html'

index.html          100% [=====>]          674  --.-KB/s   in 0s

2024-02-01 07:32:33 (28.1 MB/s) - 'index.html' saved [674/674]
```

## NGINX Web server

Nginx and Apache are web servers and load balancer servers, but Nginx is more simpler in configuration, specially in reverse proxy and load balancer configurations.

We need to install it first:

```
motaz@motaz-VirtualBox:~$ sudo apt update
motaz@motaz-VirtualBox:~$ sudo apt install nginx
```

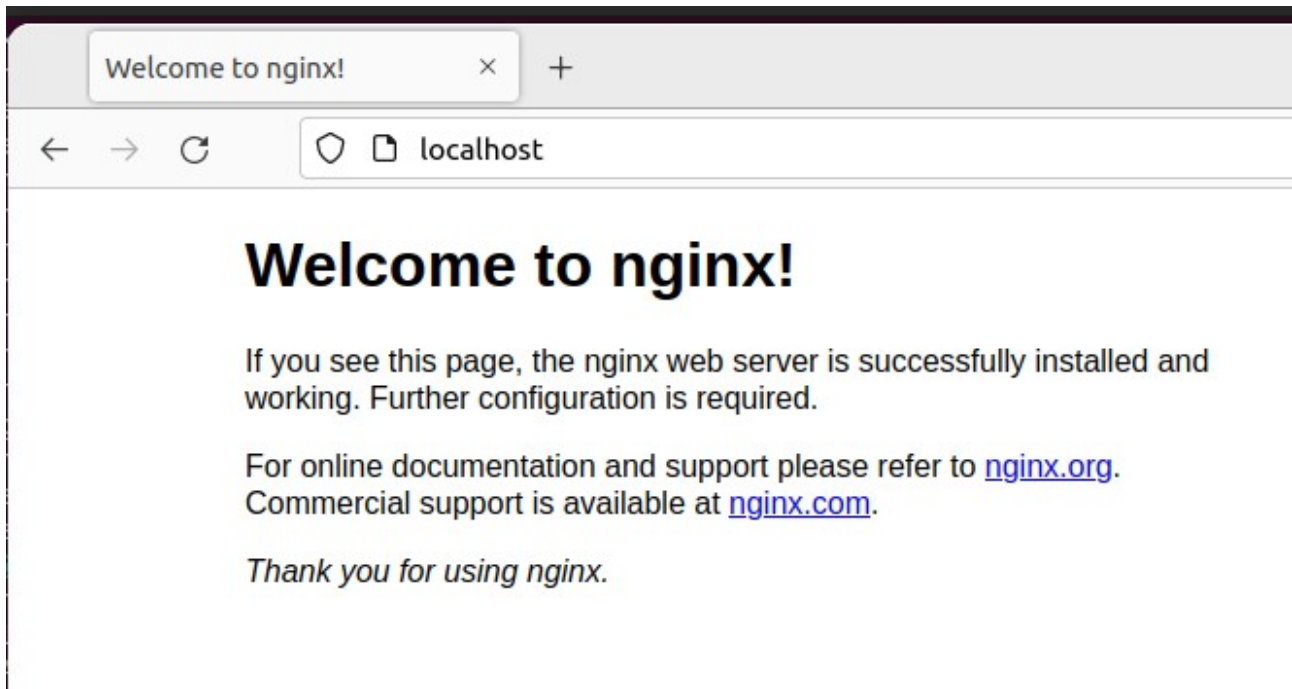
After successful installation we can check **nginx** service to make sure it is running (Active):

```
motaz@motaz-VirtualBox:~$ sudo service nginx status
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: >
   Active: active (running) since Thu 2024-02-01 08:06:27 CAT; 29s ago
     Docs: man:nginx(8)
   Process: 26148 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_proc>
   Process: 26149 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (>
  Main PID: 26261 (nginx)
    Tasks: 2 (limit: 2292)
   Memory: 3.7M
      CPU: 37ms
   CGroup: /system.slice/nginx.service
           └─26261 "nginx: master process /usr/sbin/nginx -g daemon on; maste>
             └─26264 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "">

08:06:27 01 فير motaz-VirtualBox systemd[1]: Starting A high performance web se>
08:06:27 01 فير motaz-VirtualBox systemd[1]: Started A high performance web ser>
motaz@motaz-VirtualBox:~$
```

[code.sd](#)

Second test is from browser to check <http://localhost>



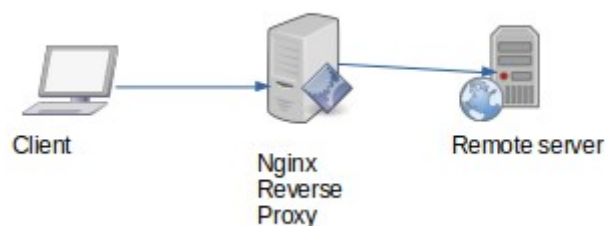
This default page is located in `/var/www/html`:

```
motaz@motaz-VirtualBox:~$ ls /var/www/html/  
index.nginx-debian.html
```

It can be replaced by **index.html** file with different content and it requires root privilege to replace or edit

## Reverse proxy

Reverse proxy is a middleware proxy that helps to connect and control connectivity between client and remote server, for many reasons one of them is that client couldn't access remote server directly, so that reverse proxy let client access that server. In our example suppose that remote server is located in Internet, and our local server (Client) has no internet access, but our Nginx proxy server has Internet, so that we could provide some Internet services through reverse proxy to our Local Client server as in picture below:



[code.sd](http://code.sd)



We need to access the site [ip2c.org](https://about.ip2c.org/) in our example, so that we need to prepare this configuration first to be inserted in Nginx http configuration:

```
location /ip2c/ {
    proxy_pass https://about.ip2c.org/;
}
```

We need to edit `/etc/nginx/sites-enabled/default` file :

```
motaz@motaz-VirtualBox:~$ sudo nano /etc/nginx/sites-enabled/default
```

and put that configurations inside server section:

```
# Default server configuration
#
server {
    listen 80 default_server;
    listen [::]:80 default_server;

#..

    root /var/www/html;

    # Add index.php to the list if you are using PHP
    index index.html index.htm index.nginx-debian.html;

    server_name _;

    location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
    }

    location /ip2c/ {
        proxy_pass https://about.ip2c.org/;
    }

#....
}
```

This means that any request to our local server, e.g <http://localhost/ip2c> will be redirected to <https://about.ip2c.org/> and result will come back through Nginx proxy to our local server.

We need to restart Nginx service to apply that change, but in real-production servers we have to check the configuration first, because if we add invalid configuration Nginx will not start again until this bad configuration been fixed, to test it use below command:

```
motaz@motaz-VirtualBox:~$ sudo nginx -t
[sudo] password for motaz:
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
```

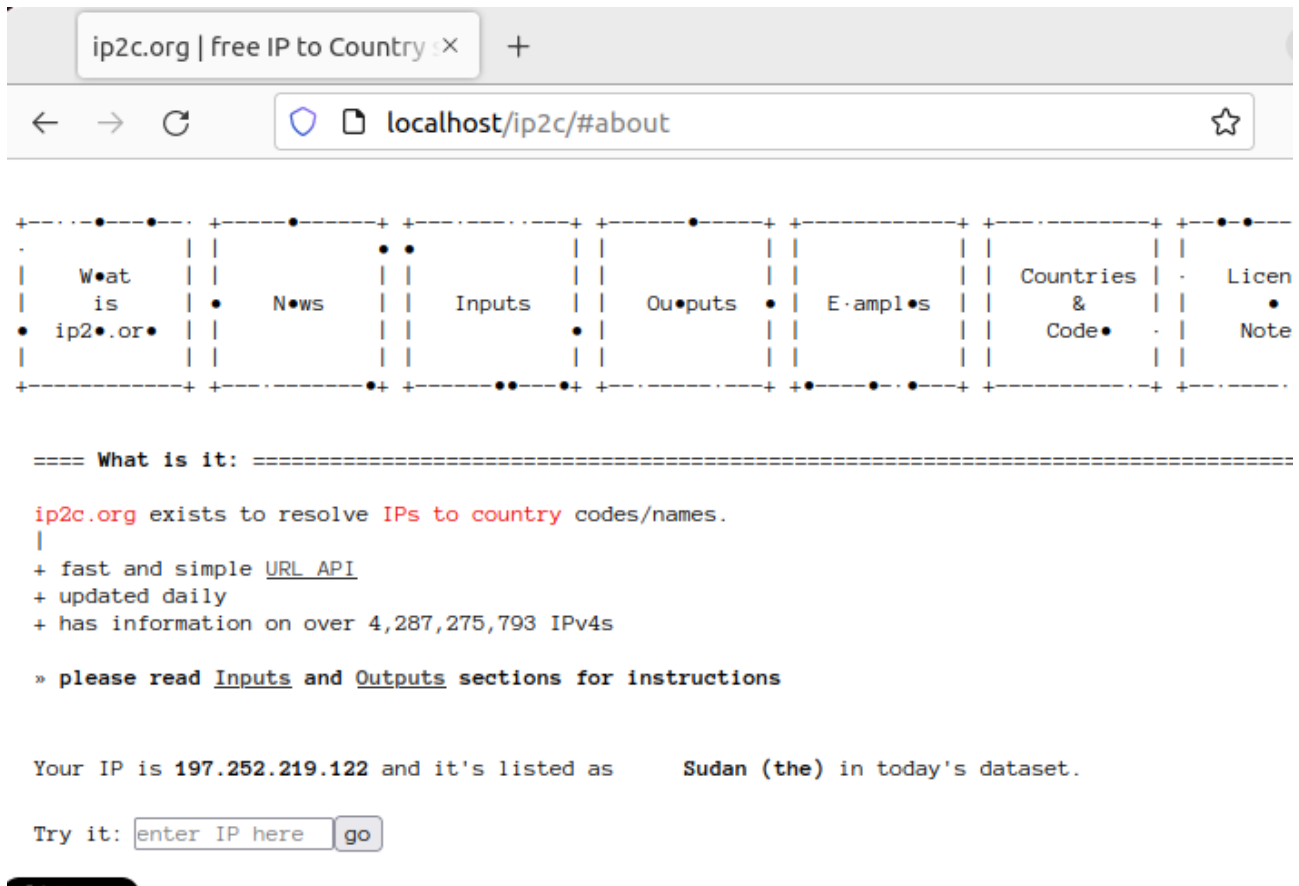
[code.sd](#)

```
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

After we get confirmation for successful configuration then we can restart it:

```
motaz@motaz-VirtualBox:~$ sudo service nginx restart
```

After that we can test it from browser:



Later we could access our local server from remote client using local server IP, for example: 192.168.1.10 as:

<http://192.168.1.10/ip2c/>

Reverse proxy is more suitable for web services, and for API integration. Here is another example of provide reverse proxy for a web service located in Internet to be available inside our local network that has no Internet except for our reverse proxy:

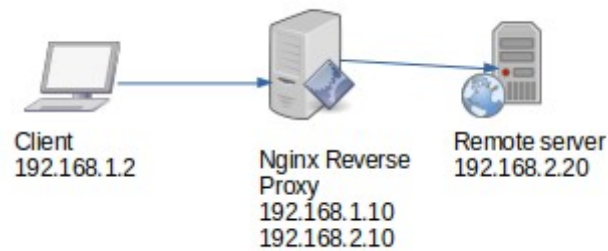
```
location /time/ {
    proxy_pass http://services.codesoft.sd/time/;
}
```

After applying it in our Nginx proxy, we could test it in the same server using **curl** as:

[code.sd](http://code.sd)

```
motaz@motaz-VirtualBox:~$ curl http://localhost/time/
{"success":true,"errormessage":"","time":"2024-02-01
09:29:15","zonename":"Africa/Khartoum","countryname":"Sudan","gmt_offset_minutes
":120,"gmt_offset_hours":2,"zone_abbreviation":"CAT"}
```

Another scenario is to access a resource (a web service) in another server in different sub-net, and proxy could access both subnets, for example:



In this case Nginx server has two Ips and could access both sub-nets, suppose that we have web service running on remote server on port 8080 as **http://192.168.2.20:8080/getinfo** and we want to provide it through Nginx proxy via port 80, we could do this configuration:

```
location /getinfo/ {
    proxy_pass http://192.168.2.20:8080/getinfo/;
}
```

And it will be accessed from clients as

<http://192.168.1.10/getinfo>

This is another reason to use reverse proxy, is to provide different web service in different servers and in different ports in one server in a default port either HTTP port 80 or HTTPS port 443. We can build a service-bus middleware server using Nginx proxy by providing different micro-services that scattered in many servers and make them available through one server.

There is an issue to remote server when using reverse proxy in the middle, which is that it hides real client Ips, so that remote server see only Nginx proxy IP, if there are many clients accessing web service in remote server through Nginx proxy, all of them will appear as Nginx proxy IP. To overcome this problem we need to add below header configuration either inside each reverse proxy **location** section, or as general inside **server** section after **listen** configuration:

```
listen [::]:80 default_server;
proxy_set_header    X-Real-IP          $remote_addr;
proxy_set_header    X-Forwarded-For $proxy_add_x_forwarded_for;
proxy_set_header    Host               $host;
```

After this configuration the remote server will still get remote address of clients as Nginx proxy IP, but when reading X-Forwarded-For header information it will get the real IP of remote client

[code.sd](#)

To prove this we need to do this interesting experiment.

We want to use **nc** tool to listen on port 9090 as a fake web service, and make proxy for it in Nginx, all that before adding above forwarding of IP address header

```
motaz@motaz-VirtualBox:~$ nc -l 9090
```

Then add reverse proxy for it in Nginx configuration:

```
location /ip/ {
    proxy_pass http://localhost:9090/;
}
```

Then call it from either curl or in browser, but note that we will not get result back to curl or browser, because there is no real web-service behind it, we want to trace the request only,

```
motaz@motaz-VirtualBox:~$ curl http://localhost/ip/
```

Then come back to nc server terminal to see that has sent to the server, we will get this:

```
motaz@motaz-VirtualBox:~$ nc -l 9090
GET / HTTP/1.0
Host: localhost:9090
Connection: close
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:99.0) Gecko/20100101 Firefox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

All that are header information, it does not contain real client IP as **X-Forwarded-For** parameter.

We need to do the test again after adding **X-Forwarded-For** headers above and run **nc** server listener again and call curl again to see the difference in request in **nc** server, we will get this result.

```
motaz@motaz-VirtualBox:~$ nc -l 9090
GET / HTTP/1.0
X-Real-IP: 127.0.0.1
X-Forwarded-For: 127.0.0.1
Host: localhost
Connection: close
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:99.0) Gecko/20100101 Firefox/99.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
```

[code.solidstate.in](https://code.solidstate.in)

```
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

In the result: Blue header information is new after doing the configuration of forwarding the IP. Web service developer has to read that values from header to know the source address of clients.

## Load balancing

Nginx can be configure as load balancing to distribute traffic for web services, also work as fail-over to recover requests to go to healthy server. Suppose that we have web service deployed in two servers on port 8080:

first server 192.168.1.20

second server 192.168.1.22

First we need to add that servers before server section of Nginx /etc/nginx/sites-enabled/default file, suppose that service name is **sms**:

```
upstream sms {
    server 192.168.1.20:8080 fail_timeout=20s;
    server 192.168.1.22:8080 fail_timeout=20s;
}
```

Then inside server section we have to add this important line to let Nginx go to next server if current called server returns one of below errors, and more http errors could be added:

```
proxy_next_upstream error timeout invalid_header http_500 http_502 http_503 http_504;
```

And last configuration part is location reverse proxy:

```
location /sms/ {
    proxy_pass http://sms/;
}
```

Requests to this endpoint <http://servername/sms/> will call distribute calls between above servers

## IPTables

**iptables** utility is used to filter networking traffic to current server, we can deny access to specific port or deny access from specific IP, for example this command is used to block traffic from this IP: 192.168.8.100

```
motaz@motaz-lenovo-t460:~$ sudo iptables -A INPUT -s 192.168.8.100 -j DROP
```

and we can check current policies using **-L** parameter:

```
motaz@motaz-lenovo-t460:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  192.168.8.100         anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

Now even ping from that IP to our server will be dropped.

To deny access to our server from all sub-net of 192.168.8.x we can execute this command:

```
motaz@motaz-lenovo-t460:~$ sudo iptables -A INPUT -s 192.168.8.0/24 -j DROP
```

To check it:

```
motaz@motaz-lenovo-t460:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
DROP      all  --  192.168.8.100         anywhere
DROP      all  --  192.168.8.0/24        anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

To allow access for that sub-net to port 80 only, we execute below command:

```
sudo iptables -I INPUT -p tcp -s 192.168.8.0/24 --dport 80 -j ACCEPT
```

and to allow ping:

```
sudo iptables -I INPUT -p icmp -s 192.168.8.0/24 -j ACCEPT
```

Note that these rules are temporary and will gone if system has rebooted. To save that rules use **iptables-save** and redirect output to a text file:

```
motaz@motaz-lenovo-t460:~$ sudo iptables-save > rules.txt
motaz@motaz-lenovo-t460:~$ cat rules.txt
# Generated by iptables-save v1.8.4 on Fri Feb  2 08:12:00 2024
*filter
:INPUT ACCEPT [3969:3135688]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [3291:385406]
-A INPUT -s 192.168.8.0/24 -p icmp -j ACCEPT
-A INPUT -s 192.168.8.0/24 -p icmp -j ACCEPT
-A INPUT -s 192.168.8.0/24 -p tcp -m tcp --dport 80 -j ACCEPT
-A INPUT -s 192.168.8.100/32 -j DROP
-A INPUT -s 192.168.8.0/24 -j DROP
COMMIT
# Completed on Fri Feb  2 08:12:00 2024
```

After that we can remove all rules using **-F** parameter:

```
motaz@motaz-lenovo-t460:~$ sudo iptables -F
```

and show again current rules to make sure there there is no left one:

```
motaz@motaz-lenovo-t460:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

We can restore it using **iptables-restore** command and redirect input from previously saved text file:

```
motaz@motaz-lenovo-t460:~$ sudo iptables-restore < rules.txt
motaz@motaz-lenovo-t460:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT    icmp -- 192.168.8.0/24        anywhere
ACCEPT    icmp -- 192.168.8.0/24        anywhere
ACCEPT    tcp  -- 192.168.8.0/24        anywhere          tcp dpt:http
DROP      all  -- 192.168.8.100        anywhere
DROP      all  -- 192.168.8.0/24        anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
```

[code.sd](#)

We can put that restore operation in a @reboot tag of sudo crontab

## Display system information

Here are some commands to return information about OS

First one **hostnamectl** that display information about host name, host type, either physical host or virtual machine, and architecture, Linux kernel version, and OS version:

```
motaz@motaz-VirtualBox:~$ hostnamectl
Static hostname: motaz-VirtualBox
    Icon name: computer-vm
    Chassis: vm
    Machine ID: abdd7334d343e9810a988c712b12399
    Boot ID: 11209a2233a8812048df61290176541
    Virtualization: oracle
Operating System: Ubuntu 22.04 LTS
    Kernel: Linux 5.15.0-25-generic
    Architecture: x86-64
    Hardware Vendor: innotek GmbH
    Hardware Model: VirtualBox
```

Another command to show current time and time zone

```
motaz@motaz-VirtualBox:~$ timedatectl
    Local time: 08:24:57 02-02-2024 ⌚ CAT
    Universal time: 06:24:57 02-02-2024 ⌚ UTC
    RTC time: 06:24:55 02-02-2024 ⌚
    Time zone: Africa/Khartoum (CAT, +0200)
System clock synchronized: no
    NTP service: active
    RTC in local TZ: no
```

To change current timezone, we can do reconfigure for timezone package, which allows re-select country and city for timezone:

```
motaz@motaz-VirtualBox:~$ sudo dpkg-reconfigure tzdata
```

