FREE LINUX GUIDE

Essential Linux Administration Topics With Practical Tools and Commands

11 Topics

Permissions & Ownership

Package Management

System Services

Network Connections

Network Troubleshooting Tools

Firewall & Security

DNS Tools

Disk & Partition Management

Crontob & Scheduled Tasks

Basic Logs & Torubleshooting

SSH & Remote Access

Linkedin.com/in/ahmedhamza15

This guide provides a practical reference for the most commonly used Linux system administration commands, covering file permissions, networking tools, service management, logs, and remote access utilities and more.

<u>The Contents</u>

Permissions & Ownership

[Tools: chmod, chgrp , chown, umask]

1. chmod

```
chmod -R 755 /var/www/html
```

⇨ Apply changes recursively (to subdirectories and files)

```
chmod -c 751 HR
```

⇨ Apply `rwxr-x--x` on the directory (`-c` like verbos)

```
chmod --reference=file1.txt file2.txt
```

⇨ Copy permissions from file1.txt to file2.txt

2. chgrp

```
chgrp IT /root/DIR
```

⇨ Change the group of `/root/DIR` to IT

3.chown

```
chown -R root:IT /root/file.txt
```

⇨ This changes the group of `/root/file.txt` to `root:IT` and operate on files and directories recursively. (This option does not follow symbolic links unless combined with `-h`, `-H`, `-L`, or `-P`)

```
chown --reference=test1.txt test.txt
```

⇨ Uses the specified reference file's (test1.txt) owner and group to file test.txt

```
chown --from=root:root ahmed:HR
```

⇨ test.txt change the owner and group of the file only if its current owner and/or group match those specified here.

```
chown 1002 test.txt
```

⇨ Change the owner of test.txt file to the user USER1 using ID (To get id you a user use: `id -u USER1`)

```
chown :IT test.txt security.txt
```
change the group only of test.txt and security.txt to IT group

4. umask

umask – the utility umask is used to set the default permissions for files or directories the user creates. (umask works by Bitwise Algorithm)
umask 0022 – First 0 is called the sticky bit (special security feature), the remaining three digits 022 are the octal values of the umask for a file or directory [0-2-2 = user-group-others]

- For a file (default: 666) - {002 means 664} {022 means 644} {000 means 666} and so on.
- For a directory (default: 777) - {002 means 775} {022 means 055} {000 means 777} and so on.

```
Umask 0543
```
This changes the default umask to 543, means default perm for file and dir will be
[for files: 123 = --x-w--wx (the real will be 224 not 123)] [for directories: 234 = -w--wxr--]
Note: it will be 224 not 123, because of x -means execute-, files can't get execute bits (x) by default. so it will be converted from 001=1=**x** to 010=2=**w**, and for group it stills **w**, and for others it will be converted from 011=3=**-wx** to 100=4=**r**)

Package Management

Tool	**Pachage Format**	**Distors**	**Family**
`apt, dpkg`	.dep	Ubuntu, Kali, Debian Debian-based	Debian-based Systems
`yum, dnf, rpm`	.rpm	CentOS, Fedora, RHEL	Red Hat-based

Debian-based Systems

`1.` **`APT`** (apt) - Advanced Package Tool

`apt update`	⇨	Refresh pachage list
`apt upgrade`	⇨	Upgrade installed packages
`apt install <package>`	⇨	Install a package
`apt remove <package>`	⇨	Remove a package (keep configs)
`apt purge <package>`	⇨	Remove package and configs
`apt autoremove`	⇨	Remove unused dependancies
`apt list --installed`	⇨	List all installed packages
`apt search <package>`	⇨	Search for a package

Note:

- *In 1998,* `apt-get` *was released with the Debian 2.0 "Hamm" distribution as a command-line tool for managing packages. In 2015, the more user-friendly* `apt` *command was introduced with Debian 8 "Jessie", combining the functionalities of* `apt-get, apt-cache`*, and others. Today,* `apt` *is preferred for day-to-day interactive use, while apt-get remains essential for scripts due to its stable behavior.*

`2.` **`dpkg`** – Debian Package Manager

`dpkg -i <package.deb>`	⇨	Install a .deb file manually
`dpkg -l`	⇨	List installed packages

`dpkg -s <package>`	⇨	Show details of installed package
`dpkg -r <package>`	⇨	Remove a package (like apt remove)

Red Hat-based Systems

`1. YUM` - (yum) Yellowdog Updater Modified

`yum install <package>`	⇨	Install a package
`yum remove <package>`	⇨	Remove a package
`yum update`	⇨	Update all packages
`yum list installed`	⇨	List installed packages

`2. DNF` - (dnf) Dandified YUM

- Is a revamped replacement for YUM in Fedora/RHEL 8+
- DNF is the next upcoming major version of YUM

`dnf install <package>`	⇨	Install a package
`dnf remove <package>`	⇨	Remove a package
`dnf update <package>`	⇨	Update a package
`dnf list installed`	⇨	List installed packages

`3. RPM` - (rpm) Red Hat Package Manager

`rpm -i package.rpm`	⇨	Install .rpm manually
`rpm -qa`	⇨	List all installed packages
`rpm -qi <package>`	⇨	Show detailed info about a package
`rpm -e <package>`	⇨	Erase a package

System Services

Tools	**Distributions**	**Init System**
`systemctl, journalctl`	Ubuntu, Debian, CentOS 7+, Fedora	Systemd
`service, chkconfig`	CentOS 6, odler Debian	SysVinit (old)

Note:

- *Most modern Linux systems use Systemd*

1. Systemd – `systemctl`

`systemctl start <service>` ⇨ Start a service
`systemctl stop <service>` ⇨ Stop a service
`systemctl restart <service>` ⇨ Restart a service
`systemctl status <service>` ⇨ Check status
`systemctl enable <service>` ⇨ Enable service to start on boot
`systemctl disable <service>` ⇨ Disable auto start
`systemctl is-enabled <service>` ⇨ Make sure it is enabled on boot
`systemctl list-units --type=service`
List active services

`systemctl`
Show all active units (not just services)

2. Lagacy (servic, chkconfig) - (useful in older systems)
`service <name> start` ⇨ Start a service
`service <name> stop` ⇨ Stop a service
`service <name> status` ⇨ Show status
`service <name> on` ⇨ Enable on boot

`service <name> off`	Disable on boot

3. Viewing Service Logs: `journalctl`

`journalctl`	⇨ View full system logs
`journalctl -u <service>`	⇨ Logs for a specific service
`journalctl -xe`	⇨ Show errors and warnings
`journalctl --since today`	⇨ Show logs since today
`journalctl -f`	⇨Follow real-time logs (like tail -f)

Check Which System in use:

`ps -p 1 -o comm=`	⇨ To Check if Systemd system is used

Network Configuration

[Tools: ip, ifconfig, netplan, nmcli, nmtui]

1. ip – Manage network interfaces, routing, and IP addresses.

```
ip addr show
```
⇨ Show the IP addresses assigned to all network interfaces

```
ip link
```
⇨ Display information about network interfaces

```
ip route show
```
⇨ Display the routing table

```
ip addr show dev <int>
```
⇨ Show the IP addresses information of a specific interface

```
ip addr add <IP>/<netmask> dev <int>
```
⇨ Assign an IP add to a network int. for ex: `I`
`p addr 192.168.1.10/24 dev eth0`
for alias network interface use `eth0:0`

```
ip link show dev <int>
```
⇨ Show detailed info about a specific interface

```
ip link set <int> up
```
⇨ Bring a network interface up (to bring it down use "down")

```
ip addr del <IP>/<netmask> dev <int>
```
⇨ Remove an IP add from a network int

```
ip route add <destination> via <gateway>
```
⇨ Add a static route to the routing table. for ex:
```
ip route add 92.168.1.0/24 via 92.168.1.1
ip route del <destination>
```
⇨ Remove a static route from the routing table

```
ip route change <destivation> via <gateway>
```
⇨ Modify an existing route

```
ip neigh show
```
⇨ Show the current ARP cache

```
ip neigh add <IP> lladdr <mac> dev <int>
```
⇨ Add an entry to the ARP cache

```
ip neigh del <IP> dev <int>
```
⇨ Delete an entry from the ARP cache

```
ip -6 addr show
```
⇨ Display IPv6 address (Same as IPv4 but use -6 for IPv6)

```
ip -s link show <int>
```
⇨ Show network interface statistics (RX (Received packets) and TX (Transmitted packets) bytes, errors.

```
ip -s addr show <int>
```
⇨ Display statistics for each address on each interface

```
ip link set <int> mtu <size>
```
⇨ Change the MTU (Maximum Transmission Unit) size of a network interface. for ex:
```
ip link set eth0 mtu 1400
```

```
ip link set <int> promisc on/off
```
⇨ Enable or disable promiscuous mode, which allows the network interface to receive all packets on the network

1. ifconfig – Configure and display network interfaces (deprecated in favor of ip)

```
ifconfig
```
⇨ Show status of the currently active interfaces (or given interface like `ifconig eth0`)

```
ifconfig -a
```
⇨ Show the status of all interfaces, both up and down

```
ifconfig -s
```
⇨ display a short list (like `netstat -i`)

```
ifconfig eth0:0 <IP> up
```
⇨ This to assgin IP to the network interface alias (Alias interface is a virtual adapter attached to a physical one (e.g., physical is eth0 the aliases are `eth0:0`, `eth0:1`, `eth0:2`)

```
ifconfig eth0:0 down
```
⇨ To delete or down alias (or use: `ifconfig eth0:0 0.0.0.0`)
Note: this deletes all scope (`eth0:1 and eth0:2`)

```
ifconfig eth0:0- <IP>
```
⇨ This assigns the IP to the eth0:0 but keeps it inactive (suppress the auto-activation by using – after the alias name)

```
ifconfig eth0 <IP> netmask <mask>
```
⇨ **For assigning an IP address to an interface**

```
ifconfig up eth0
```
⇨ For activating/deactivating (up/down) an interface

2. netplan – Manage network configuration for network interfaces (especially on Ubuntu 18.04 and newer).

Edit `/etc/netplan/00-installer-config.yaml` to define network configuration

To enable or disable DHCP, you would need to interact with the network configuration tools (`Netplan`, `NetworkManager`, `systemd-networkd`, or `/etc/network/interfaces`), **not the** `ip` or `ifconfig` tools. If you want to get or release a DHCP lease on an interface manually, you can use

```
dhclient.
dhclient eth0
```

⇨ Enable DHCP (request a DHCP lease)

```
dhclient -r eth0
```

⇨ Release DHCP lease

---

3. nmcli – NetworkManager CLI – Command-line interface for managing NetworkManager

The Basic Syntax:
`nmcli <option> <object> <command>`
options: `-a`, `-c`, etc..
objects: will be one of these general, device, connection, networking, radio, agent, monitor, help
command: up, down, show, modify, etc..

Object	Description
general	Show overall NetworkManager status
device	Manage physical & virtual devices
connection	Manage network connection profiles

networking	Enable/disable all networking
radio	Manage wireless radios (Wi-Fi, WWAN)

```
nmcli general status
```
⇨ Check NetworkManager Status

```
nmcli device status
```
⇨ List Devices (without more info)

```
nmcli device show
```
⇨ List devices and its all information (powerful comm)

```
nmcli device wifi list
```
⇨ Scan Wi-Fi

```
nmcli device wifi connect "SSID" password
"password"
```
⇨ Connect to a Wi-Fi network

```
nmcli device show eth0
```
⇨ Show detailed device info

```
nmcli connection show
```
⇨ List available connections

```
nmcli connection show --active
```
⇨ Show active connections

```
nmcli connection add type ethernet con-name
"static-eth0" ifname eth0 ip4 192.168.1.100/24
gw4 192.168.1.1
nmcli connection modify "static-eth0" ipv4.dns
"8.8.8.8"
nmcli connection up "static-eth0"
```

⇨ Add a static IP to an enthernet interface

```
nmcli connection modify "eth0" ip4.method auto
nmcli connection up "eth0"
```

⇨ Change an existing connection to use DHCP (use "manuale" istead of "auto" to disable it)

```
nmcli networking off
```

⇨ Disable networking completely (use off/on to disable/enable)

```
nmcli connection edit
```

⇨ Interactive prompt mode (for easier usage)

4. nmtui – NetworkManager TUI – Text-based user interface for managing network connections

- **Easier than nmcli for beginners**
- **Great for headless servers without a full desktop GUI**
- **Can be used over SSH safely**

```
dnf install NetworkManager-tui
```

⇨ For installing nmtui -if missing- (RHEL/CentOS/Fedora)

```
apt install network-manager
```

⇨ For installing nmtui -if missing- (Debian/Unbuntu)

Network Troubleshooting & Diagnostics Tools

[Tool: `ping, traceroute, mtr, netstat, ss, telnet, nc, curl, wget`**]**

1. ping

`ping google.com` ⇨ Send ICMP packets to test connection
`ping -c 4 google.com` ⇨ Send a specific number of packets
`ping -i 5 google.com` ⇨ Set interval between packets (in seconds)
`ping -W 10 google.com` ⇨ Set timeout for a reply
`ping -s 1400 google.com` ⇨ Set packet size (in byte)

2. Traceroute

`traceroute google.com`
`traceroute -n google.com`
⇨ Show IPs only – no DNS resolving (faster)

`traceroute -w 10 google.com`
⇨ Timeout for each reply (in secs)

`traceroute -m 7 google.com`
⇨ Set maximum hops (or maximum TTLs) to max_hops instead of 30

`tracepath`
⇨ (RHEL command instead of traceroute)

3. mtr – (my traceroute) – a network diagnostic tool combines the functionality of the traceroute and ping programs in a single network diagnostic tool

`mtr -c 3 google.com` ⇨ Number of pings per hop
`mtr -r google.com` ⇨ Report mode (good for logs)
`mtr -F /root/hostnames.txt` ⇨ Reads the list of hostnames from the specified file.

4. `netstat` – Show network connections, routing tables, interfaces stats (deprecated, replaced by `ss`)

`netstat [option]`

`netstat -t`	⇨ TCP connections
`netstat -u`	⇨ UDP connections
`netstat -l`	⇨ Listening sockets
`netstat -p`	⇨ Show PID/Program name
`netstat -n`	⇨ Numeric IP/ports
`netstat -a`	⇨ Show connections listening and non-listening
`netstat -r`	⇨ Display the kernal routing table, `netstat -r` and `route -e` product the same output

Example with full command:

```
netstat -tulnp
```

5. `ss` – Socket Statistics - (modern netstat replacement)

`ss [options]`

`ss -t`	⇨ TCP
`ss -u`	⇨ UDP
`ss -l`	⇨ Listening
`ss -n`	⇨ Numeric
`ss -p`	⇨ PID/Program
`ss -a`	⇨ All sockets

Example with full command:

```
ss -tulnp
```

6. `telnet` – Test if a TCP port is open on a remote host

```
telnet [host] [port]
telnet 192.168.1.10 22
```

If it connects = port is open
If it fails = port is closed

Warning:

- ***telnet unencrypted, rarely used for real connections, but still useful for testing open ports in safe environment***

7. `nc` – Netcat **(the preferred tool for attackers for listening and binding connections)**

`nc` – Can connect to ports, open ports for listening, transfer files, and more.

```
nc [options] host port
nc -v 192.168.1.10 445
```

⇨ Verbose (445 is the port number of SMB Protocol)

`nc -n 192.168.1.10 445` ⇨ Numeric IP/port only
`nc -l 192.168.1.10 445` ⇨ Listen mode
`nc -p 192.168.1.10 445` ⇨ Local port
`nc -z 192.168.1.10 445` ⇨ Only checks if port 445 (SMB) is open

`nc -lvnp 4444` ⇨ Sets up a TCP listener on port 4444

8. `curl` – Transfer data from/to a server via various protocols (HTTP/S, FTP, etc.)

```
curl [option] URL
```

`curl -I https://example.com` ⇨ Fetch HTTP headers only

```
curl -o /root/FILE https://example.com
```

⇨ Save output to file

`curl -k https://example.com` ⇨ Ignore SSL certificate validation
`curl -u https://example.com` ⇨ Basic HTTP auth

`9. wget` – Download file over HTTP/S, FTP

```
wget [options] URL
```

```
wget https://example.com/file.zip
```
⇨ Download with default options

```
wget -O /root/File https://example.com/file.zip
```
⇨ Save to specific filename

```
wget --no-check-certificate
https://example.com/file.zip
```
⇨ Ignore invalid SSL

```
wget -q https://example.com/file.zip
```
⇨ Quit mode

Some Tools For Network performance:

`iftop` – Display bandwidth usage on an interface
`noload` – Display network traffic and bandwidth usage
`vnstat` – Network traffic monitor, tracks bandwidth usage over time
`iperf3` – Network performance testing tool (bandwidth measurement)
`bmon` – Bandwidth monitor – Bandwidth monitoring tool for network interfaces

Firewall and Security

[Tools: `iptables, ufw, firewalld`**]**

`1. iptables` – Manage packet filtering and NAT (Network Address Translation)

`iptables -L -v -n`	View rules
`iptables -A INPUT -p tcp --dport 22 -j ACCEPT`	Add a rule
`iptables -D INPUT 1`	Delete a rule
`service iptables save`	Save config

`-A`	→ Append rule
`-D`	→ Delete rule
`-L`	→ List rules
`-p`	→ Protocol (tcp/udp)
`--dport`	→ Destination port
`-j`	→ Jump target (ACCEPT, DROP, REJECT)

`2. ufw` - (Uncomplicated Firewall) - Simplified front-end for iptables (Ubuntu-based systems)

`ufw enable`	⇨ Enable firewall
`sudo ufw allow 22/tcp`	⇨ Allow port 22 (SSH)
`sudo ufw deny 80/tcp`	⇨ Deny port 80 (HTTP)
`sudo ufw status`	⇨ Show status
`sudo ufw disable`	⇨ Disable

Options: (simple compared to others)

`2. firewalld` – Dynamic Firewall Manager (CentOS/RHEL 7+)

Zones Concept in firewalld:

The concept zones in firewalld as security levels for different network interfaces. Each zone defines rules for traffic allowed in or out.

Common Zones:

drop	All incoming connections dropped, only outgoing allowed.
block	All incoming dropped with ICMP rejection.

public	Default. For untrusted networks, allows selected services.
home	For trusted home networks.
internal	Trusted for internal networks.
external	For external, uses masquerading.
dmz	For public-facing servers.
trusted	All traffic allowed.

Start, stop, enable firewalld:

```
systemctl start firewalld
systemctl enable firewalld
systemctl status firewalld
```

Check firewall state:

```
firewall-cmd --state
```

Get active zones (shwo which zone is assigned to which interface)

```
firewall-cmd --get-active-zones
```

Assign Interface to a Zone

```
firewall-cmd --zone=public --change-
interface=ens160
```

List all rules in a zone

```
firewall-cmd --zone=public --list-all
```

Open/Close Ports

```
firewall-cmd --zone=public --add-port=8080/tcp
```

⇨ (Add a port to a zone (temporary – until reboot))

```
firewall-cmd --zone=public --add-port=8080/tcp --
permanent
```

⇨ (Add a port to a zone (permanet))

```
firewall-cmd --reload
```
⇨ (Relaod firewall)

```
firewall-cmd --zone=public --remove-port=8080/tcp
```
⇨ (Remove a port (temporary))

```
firewall-cmd --zone=public --remove-port=8080/tcp
--permanent
```
⇨ (Remove a port permanent)

```
firewall-cmd --reload
```

Service Management (HTTP, SSH, etc.)

```
firewall-cmd --get-services
```
⇨ (List svailable services)

```
firewall-cmd --zone=public --add-service=http
```
⇨ (Allow a service (temporary))

```
firewall-cmd --zone=public --add-service=http --
permanent
```
⇨ (Allow a service (permanent))

```
firewall-cmd --reload
```

```
firewall-cmd --zone=public --remove-service=http
--permanent
```
⇨ (Remove a service)

Masquerading (for NAT / Routing)

```
firewall-cmd --zone=public --add-masquerade --
permanent
```
(Enable Masquerading)

```
firewall-cmd --reload
```

```
firewall-cmd --zone=public --remove-masquerade --
permanent
```

⇨ (Disable Masquerading)

```
firewall-cmd --reload
```

Port Forwarding Example (Redirect Port 80 to 8080)

```
firewall-cmd --zone=public --add-forward-
port=port=80:proto=tcp:toport=8080 –permanent
```

⇨ (Add forwarding rule)

```
firewall-cmd --reload
```

Rich Rules (Advanced)
Allow SSH form only 192.168.1.100

```
sudo firewall-cmd --permanent --zone=public --
add-rich-rule='rule family="ipv4" source
address="192.168.1.100" service name="ssh"
accept'
```

```
sudo firewall-cmd --reload
```

DNS Tools

[Tools: `dig, nslookup, host`]

`1. dig` – Domain Information Groper – is a powerful tool used for querying DNS server to obtain domain-related information.

Common Options

```
dig example.com +short
```

⇨ Provides a brief answer (usually just the IP address for A records)

```
dig @8.8.8.8 example.com
```

⇨ Specifies a custom DNS server to query

```
dig example.com +trace
```

⇨ Traces the path of a DNS query from the root servers to the authoritative nameservers for the domain

```
dig example.com +noall +answer
```

⇨ Displays only the answer section, hiding other sections like additional and authority sections

```
dig example.com +nocomments
```

⇨ Hides comments from the output

```
dig -x 8.8.8.8
```

⇨ Performs a reverse DNS lookup to resolve an IP address to a domain name

```
dig example.com +multiline
```

⇨ Outputs the answer section in a more readable format

`2. nslookup` – Name Server Lookup – a simple tool for querying DNS servers to resolve domain names into IP adds and vice versa

```
nslookup example.com
```
⇨ Performs a DNS query for a domain

```
nslookup -type=MX example.com
```
⇨ Specifies the type of DNS record to query (A, MX, CNAME, etc.)

```
nslookup example.com 8.8.8.8
```
⇨ Specifies a custom DNS server to query

```
nslookup
```
⇨ In interactive mode type a specific record type, for ex: set type=MX example.com

```
nslookup -timeout=5 example.com
```
⇨ Sets the timeout period for the query

`3. host` – a simpler DNS lookup tool compared to dig, primarily used for querying DNS records for a domain

Syntax: `host [option] domain`

```
host -t example.com
```
⇨ Specifies the type of record to query (A, MX, NS, etc.)

```
host -a example.com
```
⇨ Specifies the type of record to query (A, MX, NS, etc.)

```
host -C example.com
```
⇨ Specifies the type of record to query (A, MX, NS, etc.)
Look at man host or `host --help` to see other options

Disk & Partition Management

[Tools: `df`, `du`, `lsblk`, `fdisk`, `mount`]

1. `df` – Check disk usage

```
df -h
```

⇨ Check disk usage (-h human-readable)

```
df --block-size=1G
```

⇨ Show only 1G-blocks (can use in Giga to show 1M-blocks in Miga)

```
df -hT
```

⇨ Print file system type

```
df -h --type=TYPE
```

⇨ Print this file system type TYPE
(for ex: `df -h --type=xfs` OR `df -h --type=xfs,tmpfs`)

```
df -h --exclude-type=TYPE
```

⇨ Exclude this file syste type TYPE to file systems not of type TYPE
(for ex: `df -h --exclude-type=xfs`)

2. `du` - Check directory and file size

```
du -sh /home
```

⇨ Check directory size (-s summarize total size, -h human-readable)

```
du -sh -a HR
```

⇨ Write counts for all files, not just HR directory or directories

```
du -sh /* 2>/dev/null
```

⇨ Check largest directories

Mount a USB/Drive
Mounting: Attaching a filesystem (like a USB or a second drive) to a directory (e.g., /mnt/usb)
1- Insert USB (or attach secondary disk)
2- Check with `lsblk` or `fdisk -l`:

`3. lsblk` – List Block Devices

`lsblk`	⇨	Output and list block devices
`lsblk -f`	⇨	Output info about filesystems (`-f` or `--fs`)

This option is equivalent to
(`lsblk -o NAME,FSTYPE,FSVER,LABEL,UUID,FSAVAIL,FSUSE%,MOUNTPOINTS`)

```
lsblk -m
```
⇨ Output info about devices owner, group and mode, (is equivalent to `-o NAME,SIZE,OWNER,GROUP,MODE`)

`4. fdisk` – Manipulate Disk Partition Table

`fdisk -l`	⇨	List the partition tables

`3. mount`
```
mkdir /mnt/usb
mount /dev/sdb1 /mnt/usb
```
⇨ This mount the driver /dev/sbd1 to mnt/usb

```
unmount /dev/sdb1
```
⇨ This unmount /dev/sbd1 - also can use unmount /mnt/usb

Checking fstab File
(Check `/etc/fstab` to list all permanent disk and partition mount rules)

fstab: A file that stores permanent mount info (auto-mount no boot)
```
locate fstab
cat /etc/fstab
```

Warning:

- ***Editing this incorrectly can prevent your system from booting!***

Crontab & Scheduled Tasks

Crontab stands for “cron table.” It's a configuration file for cron, the Linux job scheduler daemon.

It allows you to schedule:

- **Scripts**
- **Commands**
- **Jobs**

To run automatically at fixed times or intervals.

`crontab -e`	⇨	Edit your crontab file
`crontab -l`	⇨	List currect user's cron jobs
`crontab -r`	⇨	Remove user's crontab (delete all jobs)

```
* * * * * <command_to_run>
| | | | |
| | | | |______ Day of the week (0-7) (0=Sunday)
| | | |_________ Month (1-12)
| | |___________ Day of month (1-31)
| |_____________ _ Hour (0-23) (0=12AM)
|_________________ Minute (0-59)
```

`crontab -e` To edit the crontab file and add a command or a scheduled task

```
* * * * * echo "Hello from cron!" >>
/home/youruser/cron_test.txt
```

This will append that message to cron_test.txt every minute.
When you leave * in a field in **crontab**, it means **"every"** for that particular field. So, *** * * * *** — This means "run the command every minute of every hour, of every day of the month, of every month, and on every day of the week."

Useful Time Examples

Runs At	**Time Expression**
Every day at midnight	0 0 * * *
Every hour	0 * * * *
Every 15 minutes	*/15 * * * *
9 AM, Monday-Friday	0 9 * * 1-5
Once at boot (no time needed)	@reboot
Run on the 26th of April at 7:30 AM	30 7 26 4 *

Basic Logs & Troubleshooting

Linux stores logs mainly under /var/log These logs help you:

- Diagnose authentication failures
- Track system errors
- Investigate boot problems
- Monitor running hardware

Important Log Files

Log File	**Description**
General system messages (Ubuntu/Debian)	`/var/log/syslog`
General system messages (RHEL/CentOS)	`/var/log/messages`
Authentication-related events	`/var/log/auth.log`
Kernal and hardware messages	`/var/log/dmesg`
Boot-related messages	`/var/log/boot.log`
Package manager	`/var/log/apt`
Scheduled task log (if enabled)	`/var/log/cron`

Commands used:

`tail -f`	⇨	Live monitoring - follow log updates
`cat / less / more`	⇨	View logs statically
`grep`	⇨	Filter and search log entires
`dmesg`	⇨	View kernel messages (boot & hardware)
`dmseg | grep disk`	⇨	Filter "disk" plug-in events or issues
`journalctl`	⇨	Modern log viewer for systemd systems - view all logs
`journalctl -xe`	⇨	Show recent errors
`journalctl -b`	⇨	Logs since last boot

SSH & Remote Access

SSH (Secure Shell)

- Is a network protocol used to securely connect to remote systems.
- It's essential for managing servers, especially headless ones (no graphical interface).
- SSH works by connecting a client program to an SSH server, called `sshd`

Some SSH Concepts:

- Public/Private Key Authentication – More secure than passwords
- SCP – Securely copy files between systems
- rsync – Efficient file transfer
- SSH Agent – Manages SSH keys and allows passwordless logins

---

1. SSH Shell Using Password

Get an SSH shell

```
ssh ahmed15@192.168.88.110
```

Passwrd: <pass_of_ahmed15>

Excute a command remotely

```
ssh ahmed15@192.168.88.141 whoami
```

---

2. SSH Using Key Authentication

Set up SSH key authentication.

Connect from **client1@DEV** to **server1@LOC**

Step 1. In client1@DEV Generate an SSH key pair (private/public):

```
ssh-keygen -t rsa -b 2048
```

Generate a key in default path and default name `~/.ssh/id_rsa`
(OR use `-f` to use another name for the key files)

```
ssh-keygen -t rsa -b 2048 -f ~/.ssh/server1_key
```
Generate a key with a unique name using `-f` (for ex: `~/.ssh/server1_key`) (this if you will create more than one key)
Now it should create two files

- `id_rsa`
  - This is the private key file that will be saved in your local machine
- `id_rsa.pub`
  - This is the public key file that you will copy to the remote server

Step 2. In client1@DEV Copy the puplick key to the remote server
```
ssh-copy-id server1@192.168.88.110
```
This copies the public key that created by default in `~/.ssh/id_rsa` & `~/.ssh/id_rsa.pub`
(this sets up passwordless login, using your SSH key instead of a password)
(OR)
```
ssh-copy-id -i ~/.ssh/server1_key.pub
ahmed15@server1
```
This copies the public key with the name you assigned (this sets up passwordless login, using your SSH key instead of a password)
Now it will ask you the password – Only this time.

Step 3. In client1@DEV connect to server1@LOC
```
ssh server1@192.168.88.110
```
Now it will connect to server1 without asking you the password

Note:
-Replace `server1` with the user you're connecting to.
-Replace `192.168.88.110` with the remote machine's IP you connect to

---

3. SCP (Secure Copy Protocol)

SCP is a TCP protocol used to copy files between systems.

Copy a file from local to remote:

```
scp /path/to/local/file
ahmed15@192.168.88.110:/path/to/remote/dir
```

Copy a file from the remote srever to your local machine:

```
scp ahmed15@192.168.88.110:/home/HR/script.sh .
```

---

4. rsync

rsync – a fast, versatile, remote (and local) file-copying tool (for more efficient transfers)

Copy files or directories:

```
rsync -av /path/to/local/dir
ahmed15@192.168.88.110:/path/to/remote/dir
```

`-z` for compression, `-P` for progress and partial file transfer

---

5. SSH Config File

Configure an SSH Config File To Login Easily

Step 1. Create your config file if it doesn't exist

```
mkdir -p ~/.ssh
```

⇨ Create .ssh as a parent directory

```
chmod 700 ~/.ssh
```

⇨ Change permissions of `.ssh` directory to `drwx------` (only user can has full control)

```
touch ~/.ssh/config
```

⇨ Create the `config` file

```
chmod 600 ~/.ssh/config
```
Change permissions of config file to `-rw-------` (only user can read and write)

Step 2. Edit the config file
```
nano ~/.ssh/config
```
Add This:
```
Host myserver1     #the_name_you_want_to_use_(choose_any_name)
  HostName 192.168.88.110  #remote_machine_ip
  User server1                #the_username_you're_connecting_to
  Port 22                  #port_of_ssh
  IdentityFile ~/.ssh/id_rsa  #path_to_your_private_key_(id_rsa)
```

Step 3. connect to server1@LOC
In terminal, type:
```
ssh myserver1
```
Now should be logged into server1

The `config` file is only for easy logging, without having to type long commands like username@192.168.80.110

6. SSH Agent
Use SSH agent for managing keys
- **`ssh-agent`** – A background program that holds your private keys in memory.
- Use it when your private key has a passphrase, and you don't want to type it every time.
- Works until you reboot or stop the agent.

Step 1. Start the ssh-agent
```
eval "$(ssh-agent -s)"
```
⇨ This starts the agent and sets environment variables to communicate with it

Step 2. Add your private key to the agent

```
ssh-add ~/.ssh/id_rsa
```

It'll ask you for the key's passphrase (if set), Then – it keeps the decrypted ke in memory
This allows you to use SSH key authentication without needing to enter your passphrase repeatedly.

Step 3. Run ssh
`ssh server1@192.168.88.110` No passphrase prompt

Some useful options in ssh-agent:
`ssh-add -l` Show or list keys in the agent (list identities)
`ssh-add -d ~/.ssh/id_rse` Delete this key
`ssh-add -D` Delete all keys

Start SSH service
`systemctl start ssh` In Ubuntu/Debian. Can use also stop/restart/status/enable/disable
`systemctl start sshd` In REHL/CentOS. Can use also stop/restart/status/enable/disable
`ufw allow ssh` Open SSH port 22 using ufw -firewall-. ufw command on Ubuntu/Debian/CentOS
`ufw delete allow ssh` Remove the SSH open TCP port 22 firewall rule

Warning:
- **Be careful working with firewalls; take care not to lock yourself out of ssh session when deleting rules.**

SSH Config Files
`/etc/ssh/sshd_config` In Ubuntu/Debian
`/etc/ssh/sshd_config.d` In REHL/CentOS

Configuration Options In `/etc/ssh/sshd_config`

`Port 22` ⇨ SSH port number

`PermitRootLogin no` ⇨ Disable direct root login

`PasswordAuthentication no`

⇨ Disable password login (enforce key-based login – high secure)

`AllowUsers username`

⇨ Allow only specific users to connect via SSH

`MaxAuthTries 3` ⇨ Limit login attempts

Note:

- `sshd_config` ⇨ Is for the SSH server (daemon) behavior
- `ssh_config` ⇨ Is for the SSH client (user connection behavior)

User-Specific SSH Configuration

These files effects only the specific user

`~/.ssh/config`

⇨ User-specific SSH client configuration

`~/.ssh/authorize_keys`

⇨ Stores public keys allowed for that user

`~/.ssh/id_rsa`

⇨ Private and public key pair (can be another name too)

`~/.ssh/id_rsa.pub`

⇨ Private and public key pair (can be another name too)

`~/.ssh/known_hosts`

⇨ List of previously connected servers' fingerprints

Example:
In `~/.ssh/config` (SSH Client — per user)

```
Host server1
    HostName 192.168.88.110
    User yourusername
    Port 2222
    IdentityFile ~/.ssh/id_rsa
```

This allows you to just type ssh server1 instead of the full command

---

Installing SSH:

`apt install openssh-server`	Ubuntu/Debian
`dnf install openssh-server`	REHL 8+ (for older use yum)
`yum install openssh-server`	CentOS 7

To reload config without killing connections

```
systemctl reload sshd
```

Final Word

I hope this guide serves as a helpful and practical reference for anyone working with Linux systems. Whether you're a beginner or someone looking to refresh your knowledge, these commands and options are essential tools for everyday system administration tasks.

Remember — the command line is a powerful environment, and the best way to master it is through regular practice and curiosity.

Thank you for reading.

— Ahmed Hamza linkedin.com/in/ahmedhamza15