

Linux System Reboot Methods

How to reboot linux system

You can reboot a Linux system using any of the following commands, depending on your access and system configuration:

Standard Way (requires sudo or root access):

bash

```
sudo reboot
```

This is the most common and safe method to reboot the system.

Scheduled reboot with shutdown :

bash

```
sudo shutdown -r now
```

- -r means reboot.
- now means do it immediately.

You can also delay the reboot like:

bash

```
sudo shutdown -r +10 "System will reboot in 10 minutes"
```

Using systemctl (for systemd-based systems):

bash

```
sudo systemctl reboot
```

⚠ Force immediate reboot (use with caution):

```
bash
```

```
sudo reboot -f
```

Or:

```
bash
```

```
echo 1 | sudo tee /proc/sys/kernel/sysrq  
echo b | sudo tee /proc/sysrq-trigger
```

This method bypasses standard shutdown procedures. **Only use if the system is unresponsive.**

📌 Verify if you have rebooted (after reboot):

```
bash
```

```
uptime
```

Or:

```
bash
```

```
who -b
```

This shows the last boot time.

Let me know if you're using a specific Linux distribution (Ubuntu, CentOS, etc.) or if you're managing a remote server — I can tailor the commands for your use case.

How to reset master password of Linux system

Resetting the **root (master) password** on a Linux system requires physical or root-level access. Below are the common methods depending on whether you're using **GRUB** (most modern Linux distributions), or have **rescue/live media** access.

🔧 Method 1: Reset Root Password via GRUB (most common)

- ✓ Requires physical access or console access via cloud/VPS provider
- Works for Ubuntu, CentOS, RHEL, Debian, etc.

Steps:

1. Reboot the system.
2. At the GRUB menu, highlight your boot entry (usually the first one) and press `e` to edit it.
3. Find the line starting with:

```
bash  
  
linux /boot/vmlinuz-...
```

4. At the end of this line, add:

```
ini  
  
init=/bin/bash
```

5. Press `Ctrl + X` or `F10` to boot with the modified parameters.
6. The system will boot into a shell prompt with root access.
7. Remount the root filesystem as read-write:

```
bash  
  
mount -o remount,rw /
```

8. Change the password:

```
bash
```

```
passwd
```

9. Reboot:

```
bash
```

```
exec /sbin/init
```

Or simply:

```
bash
```

```
reboot -f
```

🔧 Method 2: Using a Live CD/USB (if GRUB is locked or broken)

1. Boot from a Live Linux USB/CD.
2. Open terminal and find your root partition:

```
bash
```

```
lsblk
```

3. Mount your root partition:

```
bash
```

```
sudo mount /dev/sdXn /mnt  
sudo chroot /mnt
```

4. Reset the root password:

```
bash
```

```
passwd
```

5. Exit and reboot:

```
bash
```

```
exit
```

```
sudo reboot
```

⚠ Notes:

- If your system uses SELinux (e.g., CentOS/RHEL), touch the `.autorelabel` file before rebooting:

```
bash
```

```
touch /.autorelabel
```

- If you are using Ubuntu, `root` user is disabled by default. You can reset any `sudo` user's password instead.

🔒 If LUKS or GRUB Password Is Set

If full disk encryption (LUKS) or a GRUB password is set, you'll need that password first — bypass is not possible without it.

Would you like steps tailored to a specific Linux distro (Ubuntu, RHEL, CentOS, Kali, etc.)?

How to see the users list and groups list in linux

You can view the list of `users` and `groups` in Linux using the following commands:

View All Users:

1. From /etc/passwd file:

```
bash
```

```
cut -d: -f1 /etc/passwd
```

This lists all user accounts, including system users.

2. To see only human users (UID >= 1000):

```
bash
```

```
awk -F: '$3 >= 1000 && $3 < 65534 {print $1}' /etc/passwd
```

(Excludes system users)

View All Groups:

1. From /etc/group file:

```
bash
```

```
cut -d: -f1 /etc/group
```

2. List all groups for the current user:

```
bash
```

```
groups
```

3. List groups for a specific user:

```
bash
```

```
groups username
```

🔧 Additional Commands:

Check current user:

```
bash
```

```
whoami
```

List all logged-in users:

```
bash
```

```
who
```

List users with login shell (real users):

```
bash
```

```
grep -E '/bin/bash|/bin/sh' /etc/passwd | cut -d: -f1
```

Would you like to get only admin/sudo users or users in a specific group as well?

How to reset an user password in linux

To reset a user's password in Linux, you need to have **root** or **sudo** privileges.

Reset Password for a User

-  If you're root or using `sudo`:

```
bash
```

```
sudo passwd username
```

Example:

```
bash
```

```
sudo passwd vinay
```

You'll be prompted to **enter the new password twice**.

What Happens Internally?

- The `passwd` command updates the password in `/etc/shadow`.
- It automatically handles encryption (usually with SHA-512).

Error: `passwd: Authentication token manipulation error`

If you get this error in **rescue mode** or after booting with `init=/bin/bash`, fix it like this:

```
bash
```

```
mount -o remount,rw /
```

Then retry:

```
bash
```

```
passwd username
```

✓ Confirm Password Change

Try switching to the user:

```
bash
```

```
su - username
```

You should be able to log in with the new password.

! Tip: Expire Password (Force Reset on Next Login)

```
bash
```

```
sudo chage -d 0 username
```

Let me know if you want to reset the password for the **root** user or using a **script** for bulk reset.

How to add or increase or decrease the swap memory in linux

To add, increase, or decrease swap memory in Linux, you can either use:

- **Swap file** (more flexible, common on modern systems)
- **Swap partition** (older method, less flexible)

Here's how to add/increase/decrease swap using both methods.

🧠 Check Existing Swap

bash

```
swapon --show  
free -h
```

📌 Method 1: Add or Increase Swap using a **Swap File** (Recommended)

🔧 Steps to Add a 2GB Swap File:

bash

```
sudo fallocate -l 2G /swapfile  
# OR use dd if fallocate is not available  
# sudo dd if=/dev/zero of=/swapfile bs=1M count=2048
```

bash

```
sudo chmod 600 /swapfile  
sudo mkswap /swapfile  
sudo swapon /swapfile
```

✅ Verify:

bash

```
swapon --show  
free -h
```

📌 Make It Persistent (after reboot)

Add to `/etc/fstab`:

bash

```
echo '/swapfile none swap sw 0 0' | sudo tee -a /etc/fstab
```

Decrease Swap Size (if swap file exists)

1. Turn off and remove current swap:

bash

```
sudo swapoff /swapfile  
sudo rm /swapfile
```

2. Create smaller swap file, e.g., 1 GB:

bash

```
sudo fallocate -l 1G /swapfile  
sudo chmod 600 /swapfile  
sudo mkswap /swapfile  
sudo swapon /swapfile
```

3. Confirm and update `/etc/fstab` if needed.

Method 2: Add or Resize Swap Partition (Advanced)

 Requires partitioning and system reboot. Riskier.

1. Use `lsblk` or `fdisk -l` to check swap partition.
2. Turn off swap:

bash

```
sudo swapoff /dev/sdXN
```

3. Resize partition using tools like `gparted` (GUI) or `fdisk` / `parted`.

4. Format and enable:

```
bash
```

```
sudo mkswap /dev/sdXN  
sudo swapon /dev/sdXN
```

5. Update `/etc/fstab` if necessary.

⚙️ Optional: Tune Swap Usage

You can control swap tendency with `swappiness`:

```
bash
```

```
cat /proc/sys/vm/swappiness
```

Set a lower value (like `10`) to reduce swap usage:

```
bash
```

```
sudo sysctl vm.swappiness=10
```

To make it permanent, add to `/etc/sysctl.conf`:

```
conf
```

```
vm.swappiness = 10
```

Would you like a script to automate swap addition? Or are you using a specific distro like Ubuntu, RHEL, or CentOS?

From which disk fallocate will take space

Great question! When you use the `fallocate` command to create a swap file, the space is allocated from the **filesystem** where the file is being created — typically your **root (/)** **partition**, unless you specify another location.

🔍 Example:

bash

```
sudo fallocate -l 2G /swapfile
```

- This creates the file `/swapfile` in the root filesystem.
- So the **space will be taken from the disk mounted as /**.
- You can check this with:

bash

```
df -h /
```

📦 If you want to place swap on a different disk or mount point:

Let's say you mounted another disk at `/mnt/data`, and you want swap space from there:

bash

```
sudo fallocate -l 2G /mnt/data/swapfile
sudo chmod 600 /mnt/data/swapfile
sudo mkswap /mnt/data/swapfile
sudo swapon /mnt/data/swapfile
```

Then, make it persistent by adding this line to `/etc/fstab`:

```
bash
```

```
/mnt/data/swapfile none swap sw 0 0
```

✓ Summary:

Command	Disk space used from
<code>fallocate -l 2G /swapfile</code>	Root (/) partition
<code>fallocate -l 2G /mnt/data/swapfile</code>	Mounted disk at /mnt/data

Let me know if you want to check which mount points map to which disks (`lsblk`, `df -h`, etc.), or if you're dealing with cloud VMs (AWS, Azure, etc.).

Explain the contents in `/etc/fstab`

The `/etc/fstab` file in Linux is a **static configuration file** that contains information about **disk partitions and other filesystems** that should be **automatically mounted** at boot time.

📁 /etc/fstab Format

Each line in `/etc/fstab` represents one filesystem and has the following **6 fields**:

```
php-template
```

```
<file system> <mount point> <type> <options> <dump> <pass>
```

Example /etc/fstab

bash

```
UUID=123e4567-e89b-12d3-a456-426614174000 / ext4 defaults 0 1
UUID=456f1234-d456-7e89-a123-426614174abc /home ext4 defaults 0 2
/dev/sda1 /boot ext4 defaults 0 2
/dev/sdb1 none swap sw 0 0
```

Field-by-Field Breakdown

Field	Description
1. File System	Can be a device name (<code>/dev/sda1</code>), UUID (<code>UUID=...</code>), LABEL (<code>LABEL=...</code>), or network FS
2. Mount Point	Where the filesystem will be mounted (e.g. <code>/</code> , <code>/home</code> , <code>swap</code> , or <code>none</code>)
3. Type	Filesystem type (e.g. <code>ext4</code> , <code>xfs</code> , <code>vfat</code> , <code>ntfs</code> , <code>swap</code> , <code>nfs</code> , <code>tmpfs</code> , etc.)
4. Options	Mount options like <code>defaults</code> , <code>ro</code> , <code>rw</code> , <code>noexec</code> , <code>nosuid</code> , <code>nodev</code> , <code>noatime</code> , <code>user</code> , <code>nofail</code> , etc.
5. Dump	Used by <code>dump</code> command to decide if the partition should be backed up (usually <code>0</code> or <code>1</code>)
6. Pass	Used by <code>fsck</code> to determine the order of filesystem checks at boot: <code>0</code> = do not check <code>1</code> = check first (usually <code>/</code>) <code>2</code> = check later (other partitions)

Common Entries

Root filesystem:

```
ini  
  
UUID=xxxx-xxxx / ext4 defaults 0 1
```

Swap:

```
bash  
  
/swapfile none swap sw 0 0
```

CD/DVD drive:

```
bash  
  
/dev/sr0 /mnt/cdrom iso9660 ro,user,noauto 0 0
```

External drive (vfat):

```
bash  
  
/dev/sdb1 /mnt/usb vfat defaults,noauto,user 0 0
```

📌 Tips

- Use `blkid` to get UUIDs of devices.
- Use `mount -a` to test mount all entries without reboot.
- Use `lsblk`, `df -h`, or `findmnt` to explore mounted filesystems.

Would you like to validate or modify your existing `/etc/fstab` safely? I can guide you.

How to see the ports that are being used and free

To see which ports are being used (listening or established) and which ones are free on a Linux system, you can use several tools. Here's a comprehensive guide:

💡 View Used (Open/Listening) Ports

1. Using `ss` (modern replacement for `netstat`)

bash

```
sudo ss -tuln
```

- `-t` = TCP
- `-u` = UDP
- `-l` = Listening
- `-n` = Show numeric (no DNS)

⭐ To include process info:

bash

```
sudo ss -tulnp
```

2. Using `netstat` (older but available)

bash

```
sudo netstat -tuln
```

With process info:

```
bash
```

```
sudo netstat -tulnp
```

3. Using `lsof` (List Open Files)

```
bash
```

```
sudo lsof -i -P -n
```

Only listening ports:

```
bash
```

```
sudo lsof -i -P -n | grep LISTEN
```

✓ Example Output (from `ss -tuln`):

```
nginx
```

Netid	State	Local Address:Port	...
tcp	LISTEN	0.0.0.0:22	
tcp	LISTEN	127.0.0.1:5432	
udp	LISTEN	0.0.0.0:68	

Means:

- Port 22 (SSH) is listening on all interfaces
- Port 5432 (PostgreSQL) on localhost
- Port 68 (DHCP client) via UDP

💻 How to Check Free/Unused Ports

There's no built-in "free ports list", but you can find one by checking which ports are not in use.

1. Check if a port is free:

```
bash  
  
sudo ss -tuln | grep ':8080'
```

If no output, port 8080 is probably free.

2. Scan for free TCP ports (e.g. in a script):

```
bash  
  
for port in {8000..8010}; do  
  if ! ss -tuln | grep -q ":$port "; then  
    echo "Port $port is free"  
  fi  
done
```

💼 Bonus: Use nmap to Scan Local Open Ports

```
bash  
  
sudo nmap -sT -O localhost
```

Or for a specific range:

```
bash  
  
sudo nmap -p 1-10000 localhost
```

🎯 Summary

Task	Command
View listening ports	<code>sudo ss -tuln</code>
View ports + process	<code>sudo ss -tulnp</code>
Check if port is free	<code>'ss -tuln</code>
Scan port range	<code>nmap -p 8000-8100 localhost</code>
Show all sockets	<code>sudo lsof -i</code>

Let me know if you want to monitor a specific service's port or need a script to dynamically find an unused port.

How to kill a process in linux

To **kill a process in Linux**, you can use several commands depending on what information you have (PID, name, etc.). Here's a clear breakdown:

✓ 1. Find the Process

🔍 By Name:

bash

```
ps aux | grep process_name
```

🔍 With pidof (for simple cases):

```
bash
```

```
pidof process_name
```

✖ 2. Kill the Process

🔗 By PID (Process ID):

```
bash
```

```
kill PID
```

Example:

```
bash
```

```
kill 1234
```

To force kill (use if normal kill doesn't work):

```
bash
```

```
kill -9 PID
```

🔗 3. Kill by Process Name

```
bash
```

```
pkill process_name
```

To force kill:

```
bash
```

```
pkill -9 process_name
```

4. Kill All Instances

```
bash
```

```
killall process_name
```

Example:

```
bash
```

```
killall firefox
```

To force:

```
bash
```

```
killall -9 firefox
```

Extra: Check if Process is Gone

```
bash
```

```
ps aux | grep process_name
```

Or:

```
bash
```

```
pgrep process_name
```

Permissions

- You can only kill processes that **you own**.

- To kill processes started by **other users**, use `sudo`:

```
bash
```

```
sudo kill -9 PID
```

Common Use Cases

Task	Command
Gracefully kill by PID	<code>kill PID</code>
Force kill by PID	<code>kill -9 PID</code>
Kill by name	<code>pkill name</code>
Kill all processes by name	<code>killall name</code>
Kill all Java processes	<code>pkill java</code>

Let me know if you want to **kill processes by port number, memory usage, or interactive tool like `htop`**.

When a user switched on the cpu switch. How does the linux system get started what are the files that being involved and what is the sequence and how does they help in booting

Great question! When you power on a Linux system, a complex but structured **booting sequence** takes place. It transitions from bare hardware to a fully running user-space environment (like your terminal or GUI). Here's the **step-by-step sequence**, including **key files** and **components** involved:



Linux Boot Process (Step-by-Step)



1. Power ON and BIOS/UEFI Initialization

- What happens:

CPU starts executing firmware stored on the motherboard: **BIOS** (Legacy) or **UEFI** (Modern systems).

- Tasks:

- Hardware initialization (CPU, RAM, disk, keyboard)
- POST (Power-On Self Test)
- Load boot device (from HDD, USB, PXE, etc.)

- Files involved: Firmware-level, no Linux files yet.



2. Bootloader Stage (GRUB or similar)

- What happens:

BIOS/UEFI loads the **bootloader** from the **MBR** (Master Boot Record) or **EFI** partition.

- Common bootloaders: GRUB, GRUB2, LILO, systemd-boot, etc.

- Tasks:

- Presents boot menu (if configured)
- Loads the **Linux kernel** and **initramfs/initrd** into memory

- Files involved:

- /boot/grub/grub.cfg – GRUB config file
- /boot/vmlinuz-* – the Linux **kernel image**
- /boot/initrd.img-* or /boot/initramfs-* – **Initial RAM disk**

3. Kernel Stage

- What happens:
 - The **Linux kernel** takes control.
 - Mounts the **initramfs/initrd** (temporary root FS in memory).
 - Initializes drivers, devices, memory, etc.
 - Looks for the **real root filesystem** to boot from (like `/dev/sda1`).
- Files involved:
 - Kernel modules in `/lib/modules/$(uname -r)/`
 - `init` inside initramfs (mini init system)
 - `/etc/fstab` – used later to mount actual filesystems

4. Init System Starts (PID 1)

- What happens:
 - Kernel passes control to **PID 1**, the **init system**.
 - In modern Linux, this is usually `systemd`, but could also be `SysVinit` or `OpenRC`.
- Tasks:
 - Mount real root filesystem (`/`)
 - Mount other partitions from `/etc/fstab`
 - Start essential services, configure networking, logs, etc.
- Files involved:
 - `/sbin/init` or `/lib/systemd/systemd`
 - `/etc/fstab`
 - `/etc/systemd/system/*.service`
 - `/etc/inittab` (only on SysVinit systems)



5. User-Space Initialization

- What happens:
 - Init system starts **target/runlevel units** (multi-user, graphical, etc.)
 - Starts system services (cron, sshd, networking, etc.)
 - Starts **login prompt** (`getty` on console or `gdm/lightdm` for GUI)
- Files involved:
 - `/etc/systemd/system/default.target`
 - `/etc/profile`, `/etc/bashrc` (for shell login setup)



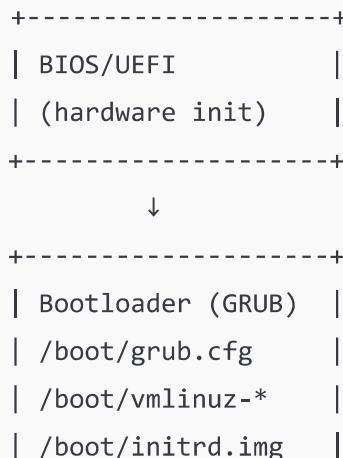
6. User Login and Shell Start

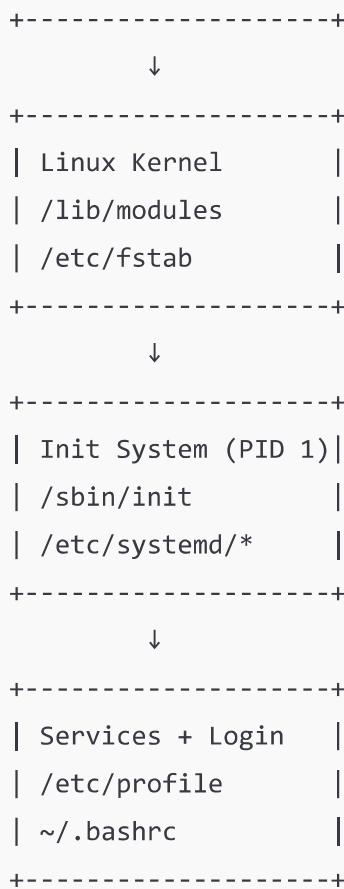
- You log in via terminal or GUI.
- Shell environment (`.bashrc`, `.profile`) is initialized.
- You get full access to the Linux user space.



Visual Summary:

text





⚙️ Commands to Explore Boot:

- Show boot logs:

```
bash  
  
journalctl -b
```

- Show boot performance:

```
bash  
  
systemd-analyze  
systemd-analyze blame
```

- List all services started:

```
bash
```

```
systemctl list-units --type=service
```

Let me know if you want this boot sequence for Ubuntu, CentOS, or a server vs desktop version. I can also show you how to **customize the boot process** or debug boot issues.

How to configure the service and its logs in linux

To **configure a service** and manage its **logs** in Linux, the exact steps depend on whether it's a **custom service**, a **systemd-managed service**, or a **third-party service** like Apache, MySQL, etc.

I'll walk you through both **creating/configuring a systemd service** and **managing its logs** using `journalctl` or log files.



Part 1: Configure a Service using `systemd`

Systemd is the most common init system in modern Linux distros (Ubuntu, RHEL, CentOS, Debian, etc.).

Step 1: Create a systemd service file

Create a unit file in:

```
bash
```

```
sudo nano /etc/systemd/system/myapp.service
```

Example content:

```
ini
```

```
[Unit]
Description=My Custom App Service
After=network.target

[Service]
ExecStart=/usr/local/bin/myapp.sh
Restart=always
User=myuser
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

Replace `/usr/local/bin/myapp.sh` with your script or binary.

⚙️ Step 2: Reload systemd and enable the service

```
bash

sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable myapp.service
sudo systemctl start myapp.service
```

Check status:

```
bash

systemctl status myapp.service
```

📋 Part 2: Configure Logs for the Service

🛠 Option 1: Use systemd journal (default and recommended)

- Logs go to `journald`
- View logs:

```
bash  
  
journalctl -u myapp.service
```

- To follow logs in real-time:

```
bash  
  
journalctl -u myapp.service -f
```

- Logs are stored in `/var/log/journal/`

🔧 Configure journald (optional):

Edit `/etc/systemd/journald.conf` to:

- Increase log size
- Persist logs across reboots:
Set `Storage=persistent`

Then:

```
bash  
  
sudo systemctl restart systemd-journald
```

🛠 Option 2: Send output to a log file

Edit the service to:

```
ini  
  
[Service]  
ExecStart=/usr/local/bin/myapp.sh
```

```
StandardOutput=append:/var/log/myapp.log
```

```
StandardError=append:/var/log/myapp.err
```

Or use redirection in the command:

```
ini
```

```
ExecStart=/usr/local/bin/myapp.sh >> /var/log/myapp.log 2>&1
```

Ensure:

```
bash
```

```
sudo touch /var/log/myapp.log  
sudo chmod 644 /var/log/myapp.log
```

💡 Log Rotation (Optional)

If your log is written to a file like `/var/log/myapp.log`, configure log rotation:

Create a file:

```
bash
```

```
sudo nano /etc/logrotate.d/myapp
```

Example:

```
conf
```

```
/var/log/myapp.log {  
    daily  
    rotate 7  
    compress  
    missingok  
    notifempty  
    create 644 root root  
}
```

Test with:

```
bash
```

```
sudo logrotate -d /etc/logrotate.d/myapp
```

📌 Summary

Task	Command/Location
Create service	/etc/systemd/system/myapp.service
Start/Enable service	systemctl start/enable myapp
View logs	journalctl -u myapp
Redirect logs	Use StandardOutput or >>
Rotate logs	/etc/logrotate.d/

Let me know if you're working with a specific app (e.g., Nginx, Node.js, Java, Python), or want to forward logs to remote syslog or ELK.

What are the different available logs in linux to see the root causes

In Linux, various **log files** are maintained under `/var/log/` to help diagnose issues and find the **root cause** of problems related to the system, services, network, kernel, hardware, and more.

Here's a categorized guide to the most important log files you can inspect to troubleshoot:

📁 Location of Logs

Most logs are located in:

```
bash
```

```
/var/log/
```

You can list them with:

```
bash
```

```
ls -l /var/log
```

📊 Major Types of Logs

1. System and Kernel Logs

Log File	Description
/var/log/messages	General system messages (RedHat/CentOS)
/var/log/syslog	General system log (Ubuntu/Debian)
/var/log/kern.log	Kernel-related messages
dmesg (command)	Boot-time kernel ring buffer (hardware, drivers, etc.)

Use `sudo dmesg | less` to explore hardware/kernel events.

2. Authentication and Security Logs

Log File	Description
/var/log/auth.log	Auth logs (Ubuntu/Debian) – login, sudo, ssh

Log File	Description
/var/log/secure	Auth logs (RedHat/CentOS)
/var/log/faillog	Failed login attempts
/var/log/btmp	Binary log of failed login attempts
/var/log/wtmp	Login/logout records
/var/log/lastlog	Last login for all users

Commands to view:

```
bash
lastlog
last
faillog
```

3. Boot and Init Logs

Log File	Description
journalctl -b	System boot logs (systemd)
/var/log/boot.log	Boot process messages

4. Systemd Logs (Most Modern Distros)

```
bash
journalctl -xe          # Shows recent errors and service crashes
journalctl -u nginx.service # Logs for a specific service
```

```
journalctl --since "1 hour ago"
```

Systemd logs everything into a binary format under `/var/log/journal/`.

5. Service-Specific Logs

Service	Common Log Files
Apache/Nginx	<code>/var/log/httpd/</code> or <code>/var/log/nginx/</code>
MySQL/MariaDB	<code>/var/log/mysql/</code> or <code>/var/log/mysqld.log</code>
SSH	<code>/var/log/auth.log</code> or <code>/var/log/secure</code>
Cron	<code>/var/log/cron</code> or <code>/var/log/syslog</code>
Docker	<code>journalctl -u docker</code> or <code>/var/lib/docker/containers/*/*.log</code>

6. Package and Update Logs

File	Description
<code>/var/log/apt/</code>	Apt logs (Ubuntu/Debian)
<code>/var/log/yum.log</code>	Yum logs (RHEL/CentOS)
<code>/var/log/dpkg.log</code>	Debian package log

7. Hardware and System Events

- `dmesg` – BIOS, disk, memory errors, driver loading
- `/var/log/Xorg.0.log` – GUI/X11 logs (graphics issues)

- `/var/log/hardware.log` or vendor-specific tools
-

💼 Useful Log Inspection Commands

bash

```
tail -f /var/log/syslog  
less /var/log/auth.log  
grep error /var/log/syslog  
journalctl -xe
```

💡 Pro Tip: Use `logwatch` or `goaccess` for summary reports

- `logwatch` – daily system logs report
 - `goaccess` – real-time web server log analyzer
-

💡 How to Diagnose Issues?

Issue Type	Check These Logs
Boot problems	<code>journalctl -b</code> , <code>dmesg</code> , <code>/var/log/boot.log</code>
Login/auth failures	<code>/var/log/auth.log</code> , <code>/var/log/secure</code> , <code>faillog</code>
SSH issues	<code>/var/log/auth.log</code> , <code>/var/log/secure</code>
Kernel crashes	<code>/var/log/kern.log</code> , <code>dmesg</code>
App crashes	<code>journalctl -xe</code> , specific app log
Network issues	<code>dmesg</code> , <code>/var/log/syslog</code> , <code>journalctl -u NetworkManager</code>

Let me know what specific issue you want to troubleshoot, and I'll guide you to the right log and filters.

How to see the installed packages or softwares in the system along with their versions

You can view all **installed packages and their versions** in a Linux system using commands that depend on your **package manager and distribution**.

✓ 1. Ubuntu/Debian-based Systems (APT)

🔍 Show all installed packages:

bash

```
dpkg -l
```

📦 With version info only:

bash

```
dpkg-query -W -f='${binary:Package} ${Version}\n'
```

🔍 Filter by name:

bash

```
dpkg -l | grep nginx
```

📋 Better formatted list:

bash

```
apt list --installed
```

✓ 2. RedHat/CentOS/Fedora/Rocky (YUM/DNF)

🔍 Show all installed packages:

bash

```
rpm -qa
```

📦 With package name and version:

bash

```
rpm -qa --queryformat '%{NAME} %{VERSION}-%{RELEASE}\n'
```

🔍 Filter by name:

bash

```
rpm -qa | grep httpd
```

💻 Using yum or dnf:

bash

```
yum list installed  
# or  
dnf list installed
```

✓ 3. Arch Linux (pacman)

```
bash
```

```
pacman -Q
```

Filter by name:

```
bash
```

```
pacman -Qs nginx
```

✓ 4. Flatpak or Snap Packages

If you're using modern packaging systems:

Flatpak:

```
bash
```

```
flatpak list
```

Snap:

```
bash
```

```
snap list
```



BONUS: Check Application Version Directly

```
bash
```

```
nginx -v  
python3 --version
```

```
node -v  
java -version
```

(Use the command name + `--version` or `-v`)

💡 Summary by Distribution

Distro	Command
Ubuntu/Debian	<code>dpkg -l</code> , <code>apt list --installed</code>
RHEL/CentOS	<code>rpm -qa</code> , <code>yum list installed</code>
Fedora	<code>dnf list installed</code>
Arch Linux	<code>pacman -Q</code>
Snap	<code>snap list</code>
Flatpak	<code>flatpak list</code>

Let me know your distro (`Ubuntu`, `RHEL`, `Alma`, etc.), and I can tailor a full script or command to export all packages to a file (for backup, audit, or comparison).

How does patch will be done for vulnerabilities management in os

Patching for **vulnerability management** in a Linux OS involves **identifying, testing, and applying** updates to software and system components that fix known security issues. Here's a breakdown of how it's done step-by-step:

🔒 1. Identify Vulnerabilities

- Use vulnerability scanners:

- `Lynis`, `OpenVAS`, `Nessus`, `Qualys`, `Rapid7`, `Tenable`

- Use OS built-in tools:

- Ubuntu/Debian:

```
bash
```

```
sudo apt list --upgradable
```

or

```
bash
```

```
sudo unattended-upgrade --dry-run --debug
```

- RedHat/CentOS:

```
bash
```

```
sudo yum check-update
```

```
sudo dnf updateinfo list security all
```

- Check vendor security advisories:

- [Ubuntu Security Notices \(USN\)](#)

- [Red Hat CVE Database](#)

🛠 2. Patch/Update the System

✓ For Ubuntu/Debian:

```
bash
```

```
sudo apt update
```

```
sudo apt upgrade      # Installs all upgradable packages
```

```
sudo apt full-upgrade      # Also upgrades kernel and system packages
```

For security updates only:

```
bash
```

```
sudo unattended-upgrade
```

For RHEL/CentOS:

```
bash
```

```
sudo yum update          # For older RHEL/CentOS  
sudo dnf update          # For newer systems (RHEL 8+)
```

For security patches only:

```
bash
```

```
sudo dnf updateinfo list security  
sudo dnf upgrade --security
```

You can also enable automatic security updates using `yum-cron` or `dnf-automatic`.

3. Automate with Patch Management Tools

Use tools for enterprise patching:

Tool	Description
WSUS	Windows-centric, limited Linux
Red Hat Satellite	Patch RHEL systems at scale
Canonical Landscape	Patch/manage Ubuntu at scale
Ansible + apt/yum modules	Automate patching via playbooks

Tool	Description
AWS Systems Manager	Patch Linux/Windows EC2 instances
Chef/Puppet/SaltStack	Infrastructure as Code patching

4. Validate the Patching

- Check logs:
 - `/var/log/apt/history.log` (Debian/Ubuntu)
 - `/var/log/yum.log` or `dnf.log` (RHEL/CentOS)
- Reboot if kernel or core system packages were updated.

```
bash
```

```
sudo reboot
```

5. Post-Patching Actions

- Run a security scan again
- Check services are running (`systemctl status`)
- Monitor logs (`journalctl -xe`, `/var/log/syslog`, etc.)

Best Practices

- Always patch test systems before production
- Subscribe to security bulletins for your distro
- Automate **weekly or daily patching schedules**

- Use vulnerability baselines (CIS Benchmarks)

If you're in an enterprise environment, I can help you set up **automated patching workflows** using `Ansible`, `AWS SSM`, or `cron` jobs. Let me know your Linux distro and environment scale (on-prem/cloud/etc.).

How to upgrade all services ? How can do individually

Upgrading services (like `nginx`, `mysql`, `ssh`, etc.) on a Linux system depends on your **package manager**. You can either upgrade **all services at once** or **individually**.

1. Upgrade All Services (System-Wide Upgrade)

Ubuntu / Debian:

```
bash

sudo apt update
sudo apt upgrade          # Upgrade all packages
sudo apt full-upgrade     # Also upgrades kernel and system components
```

Red Hat / CentOS / Fedora:

```
bash

sudo yum update          # For RHEL/CentOS 7
sudo dnf upgrade          # For RHEL/CentOS 8 and Fedora
```

This updates all packages (including services) to the latest available versions from the repositories.

✓ 2. Upgrade a Specific Service

You can upgrade individual services/packages using their names.

📌 Ubuntu / Debian:

bash

```
sudo apt install --only-upgrade <package-name>
```

◆ Example:

bash

```
sudo apt install --only-upgrade nginx  
sudo apt install --only-upgrade openssh-server
```

📌 Red Hat / CentOS:

bash

```
sudo yum update <package-name>          # RHEL/CentOS 7  
sudo dnf upgrade <package-name>          # RHEL/CentOS 8+
```

◆ Example:

bash

```
sudo dnf upgrade nginx  
sudo dnf upgrade httpd
```

🔍 3. Check Current Version of a Service

bash

```
nginx -v  
httpd -v
```

```
mysql --version
```

Or using the package manager:

- Ubuntu/Debian:

```
bash
```

```
dpkg -l | grep nginx
```

- RHEL/CentOS:

```
bash
```

```
rpm -qa | grep nginx
```

4. Restart Service After Upgrade

After upgrading, restart the service to apply changes:

```
bash
```

```
sudo systemctl restart <service-name>
```

- ◆ Example:

```
bash
```

```
sudo systemctl restart nginx
```

Tip: Upgrade Kernel (if needed)

If kernel packages were updated:

```
bash
```

```
sudo reboot
```

Summary Table

Task	Ubuntu/Debian	RHEL/CentOS/Fedora
Update package list	<code>sudo apt update</code>	N/A
Upgrade all packages/services	<code>sudo apt upgrade</code>	<code>sudo dnf upgrade</code> or <code>yum update</code>
Upgrade single package	<code>sudo apt install --only-upgrade <pkg></code>	<code>sudo dnf upgrade <pkg></code>
Restart service	<code>sudo systemctl restart <svc></code>	<code>sudo systemctl restart <svc></code>

Let me know which service you're targeting (e.g. `nginx`, `docker`, etc.) and your OS, and I can give a precise command or script.

When to use update and when to use upgrade

Great question! Understanding the difference between `update` and `upgrade` is essential for properly managing packages in a Linux system.

1. `update` – Fetch Latest Package Info (Metadata)

Purpose:

`update` fetches the latest list of available packages and their versions from the repositories. It **does not install or upgrade** any packages by itself.

✓ Example:

```
bash

sudo apt update      # Debian/Ubuntu
sudo yum check-update # RHEL/CentOS 7
sudo dnf check-update # RHEL/CentOS 8+
```

💡 Think of `update` as **checking what's available**.

🔗 2. `upgrade` – *Install Newer Versions of Installed Packages*

Purpose:

`upgrade` actually **installs the newer versions** of currently installed packages (if available), based on the latest metadata fetched by `update`.

✓ Example:

```
bash

sudo apt upgrade      # Debian/Ubuntu
sudo yum update       # RHEL/CentOS 7
sudo dnf upgrade      # RHEL/CentOS 8+
```

💡 Think of `upgrade` as **doing the actual update work**.

📌 Full Workflow Example (Ubuntu):

```
bash

sudo apt update      # Step 1: Update package metadata
sudo apt upgrade      # Step 2: Upgrade installed packages
```



Special Case (Ubuntu): `full-upgrade`

`bash`

```
sudo apt full-upgrade
```

This upgrades all packages and allows removal of old packages if needed (e.g., kernel updates).

Summary Table:

Command	What It Does	When To Use
<code>apt update</code>	Refreshes package list (metadata)	Before running any upgrade
<code>apt upgrade</code>	Installs available updates for installed packages	To apply regular updates
<code>apt full-upgrade</code>	Like <code>upgrade</code> , but also removes conflicting packages	When you want full system upgrade
<code>yum check-update</code>	Lists available updates	To see what will be updated (RHEL 7)
<code>yum update</code>	Applies all updates	Regular package maintenance
<code>dnf upgrade</code>	Applies all updates	On newer RHEL-based distros

Let me know which Linux distro you're using, and I can tailor the commands accordingly!

Do I need to stop the service before upgrading it

Great question! Here's a clear breakdown to help you decide:

Do You Need to Stop the Service Before Upgrading It?

◆ Short Answer:

Usually, no — you don't need to manually stop a service before upgrading it.

Most Linux package managers (like `apt`, `yum`, `dnf`) handle the stopping and restarting of services automatically during an upgrade.

What Happens Internally?

When you upgrade a service like `nginx`, `mysql`, or `sshd`:

- The package manager will stop the service if required,
- Replace the necessary binaries/configs,
- Then restart the service.

This is managed through service hooks in the package scripts (pre/post install scripts).

When You Might Want to Stop It Manually

You should consider stopping the service manually before upgrade if:

- The service is in heavy use, and you want a graceful shutdown (e.g., a production DB).
 - You want to avoid any brief downtime/restart surprises during upgrade.
 - You're doing a major version upgrade (e.g., `MySQL 5.7 → 8.0`) which may change data formats.
 - You're planning to backup configs/data before the upgrade.
 - You're running a clustered or load-balanced setup and want to drain traffic first.
-

How to Restart After Upgrade (if not automatic)

If the upgrade doesn't automatically restart the service or if you stopped it manually:

```
bash
```

```
sudo systemctl restart <service-name>
```

Example:

```
bash
```

```
sudo systemctl restart nginx
```

📌 Best Practice (Production Systems):

1. Notify users or drain traffic (if applicable).
2. Backup configuration and data.
3. Optionally stop the service (for safe state).
4. Perform the upgrade.
5. Start or verify the service status.

```
bash
```

```
sudo systemctl status <service-name>
```

Let me know the specific service you're upgrading (e.g., docker, mysql, nginx), and I can guide you more precisely.

Ok.. so basically apt update.... Will connect to that apt and gets the recent metadata,i.e available new release version details right ?

Yes, exactly! 

When you run:

```
bash
```

```
sudo apt update
```

🔍 What Happens Behind the Scenes:

1. APT connects to the package repositories configured in your system (listed in `/etc/apt/sources.list` and `/etc/apt/sources.list.d/*`).
2. It downloads the latest metadata — which includes:
 - A list of all available packages
 - Latest versions of each package
 - Dependencies
 - Security/bug fix info
 - Where the `.deb` files are stored
3. This metadata is stored **locally on your system**, so your next commands (like `apt install` or `apt upgrade`) know what is available.

📁 Local Cache Path (Optional Nerdy Info):

APT stores this metadata in:

```
swift
```

```
/var/lib/apt/lists/
```

🧠 So Remember:

Command	What it does
apt update	Syncs local metadata with latest info from the repositories
apt upgrade	Installs the newer versions based on that metadata

If you don't run `apt update` first, `apt upgrade` won't know about any new versions available.

Let me know if you want to explore how to add custom apt repositories, or what's inside `sources.list`.

How to read about a package

To read about a package in Linux, you can use several tools depending on what information you want. Here's a complete breakdown:

🔍 1. View Package Details (installed or available)

Debian/Ubuntu (APT-based systems):

bash

```
apt show <package-name>
```

Example:

bash

```
apt show nginx
```

This will give you:

- Version

- Description
 - Maintainer
 - Dependencies
 - Installation size
 - Homepage URL, etc.
-

2. Check If a Package is Installed

```
bash
```

```
dpkg -l | grep <package-name>
```

Or use:

```
bash
```

```
which <binary-name>
```

3. Read Manual Pages for Installed Software

```
bash
```

```
man <command-name>
```

Example:

```
bash
```

```
man nginx
```

This shows the usage, options, examples, etc.

4. Check Installed Files of a Package

bash

```
dpkg -L <package-name>      # Debian/Ubuntu  
rpm -ql <package-name>      # RHEL/CentOS
```

This lists all files and their paths that belong to the package.

5. Check Config Files Location

You can find common config file locations via:

bash

```
apt show <package-name> | grep Config
```

Or by inspecting installed file paths (`/etc`, `/usr/lib`, etc.) from the `dpkg -L` command.

6. View Vulnerabilities and Changelog

For Debian/Ubuntu:

bash

```
apt changelog <package-name>
```

Or search the security tracker:

<https://security-tracker.debian.org/>

7. Explore Online

For Ubuntu:

- <https://packages.ubuntu.com>

For Debian:

- <https://packages.debian.org>

For RHEL/CentOS:

- <https://access.redhat.com/downloads/>

✓ Example: Full Info About curl

bash

```
apt update  
apt show curl  
apt changelog curl  
man curl  
dpkg -L curl
```

Let me know the package name you're curious about, and I'll walk you through it specifically!

Is it possible to add a custom package source? If yes how

Yes, absolutely! ✓ You can add custom package sources (also called APT repositories) to your system. This is often done when:

- You want newer versions of packages than what Ubuntu/Debian provides.
- You need software from a third party (e.g., Docker, Node.js, Google Chrome).
- You have an internal repo (e.g., Artifactory or Nexus).

🔧 How to Add a Custom APT Repository (Debian/Ubuntu)

◆ Step 1: Add the GPG key

This ensures packages from the source are **trusted**.

Example:

```
bash
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker.gpg
```

◆ Step 2: Add the repository to sources list

You can either:

- Add it to `/etc/apt/sources.list`, or
- Preferably, create a `.list` file in `/etc/apt/sources.list.d/`

Example (Docker on Ubuntu):

```
bash
```

```
echo "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" \
| sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

`$(lsb_release -cs)` inserts your Ubuntu codename (e.g., `focal`, `jammy`)

◆ Step 3: Update the package index

```
bash
```

```
sudo apt update
```

Now your system knows about packages from this new source.

◆ Step 4: Install the package

```
bash
```

```
sudo apt install <package-name>
```

📁 Example: File Structure

File	Purpose
/etc/apt/sources.list	Main sources list
/etc/apt/sources.list.d/<custom>.list	Extra sources (recommended)
/etc/apt/trusted.gpg.d/<name>.gpg	Trusted GPG keys for sources

✓ Example: Add Node.js Repository

bash

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install nodejs
```

This script:

- Adds the NodeSource repo
- Adds the GPG key
- Installs Node.js from it

🛠 Bonus: Remove a Custom Source

bash

```
sudo rm /etc/apt/sources.list.d/<custom>.list  
sudo apt update
```

Let me know the exact software or package you want to add, and I'll give you the commands tailored for that.

How to see the process state using htop

To see the process state using `htop`, follow these steps:

✓ Launch `htop`

bash

htop

If it's not installed:

bash

```
sudo apt install htop      # Debian/Ubuntu  
sudo yum install htop     # RHEL/CentOS
```

🔍 Understanding Process State in `htop`

In the main `htop` interface, one of the columns (typically labeled "S") shows the process state using a single character:

Symbol	Meaning
R	Running
S	Sleeping (idle, interruptible)
D	Waiting (uninterruptible sleep)
T	Stopped (by job control signal)
Z	Zombie (terminated but not reaped)
I	Idle (kernel thread, very low CPU use)

You can see this under the "S" column or labeled as "STATE" depending on your view settings.



Customize Columns in htop

You can customize what columns to show:

1. Press `F2` (Setup)
 2. Go to **Columns**
 3. Add **State** or other process info
 4. Press `F10` to exit
-



Sort by State (optional)

You can sort by different columns by pressing:

- `<F6>` and then choosing "State"
 - Or use your arrow keys and Enter
-

📌 Example Process Line:

sql

```
PID USER PRI NI VIRT RES SHR S CPU% MEM% TIME+ Command
1001 root 20 0 ... ... ... S 0.0 0.1 00:00.01 sshd
```

Here, `S` means the process is in **sleeping state**.

Let me know if you want to filter or monitor only specific states like `Z` (zombie) or `D` (uninterruptible I/O).

Show me the syntaxes and etc related to `fg` and `bg` in linux

Sure! `fg` and `bg` are used in Linux to **manage jobs (processes)** in the foreground and background of your terminal shell. These are especially useful when you've suspended a task using `Ctrl+z`.

🎯 Basic Concepts

Command	Meaning
<code>fg</code>	Brings a background/suspended job to the foreground
<code>bg</code>	Resumes a suspended job in the background
<code>jobs</code>	Lists the current jobs running in the shell

🧪 Use Case Example

1. Start a command and suspend it

```
bash
```

```
sleep 300
```

Now press `Ctrl+Z` to suspend it.

You'll see:

```
bash
```

```
[1]+ Stopped sleep 300
```

2. Check current jobs

```
bash
```

```
jobs
```

Output:

```
bash
```

```
[1]+ Stopped sleep 300
```

3. Send to background

```
bash
```

```
bg %1
```

or just:

```
bash
```

`bg`

This resumes job 1 in the background.

4. Bring it back to foreground

`bash``fg %1`

or simply:

`bash``fg`

This brings job 1 to the foreground.



Syntax Summary

Command	Description
<code>fg</code>	Resume most recent job in foreground
<code>fg %1</code>	Resume job number 1 in foreground
<code>bg</code>	Resume most recent job in background
<code>bg %2</code>	Resume job number 2 in background
<code>jobs</code>	List all background and suspended jobs
<code>kill %1</code>	Kill job 1
<code>disown %1</code>	Remove job 1 from job table (won't be killed when terminal closes)

 `fg` | Resume most recent job in foreground | `fg %1` | Resume job number 1 in foreground | `bg` | Resume most recent job in background | `bg %2` | Resume job number 2 in background | `jobs` | List all background and suspended jobs | `kill %1` | Kill job 1 | `disown %1` | Remove job 1 from job table (won't be killed when terminal closes) |

💡 Pro Tips

- Jobs are only valid **in the current shell session**.
- `&` at the end of a command runs it directly in the background:

```
bash  
  
sleep 300 &
```

- Use `jobs -l` to see the PID of each job too.

Let me know if you want examples involving real-world commands like `tail`, `top`, or `python`.

What are the `.bashrc`, `.bash_profile`, `.bash_logout` files in the user's home directory? What are the equivalent files on system level

Great question! These shell configuration files—`.bashrc`, `.bash_profile`, and `.bash_logout`—control how the Bash shell behaves during different phases of a user's session. Let's break each down clearly, along with their **system-level equivalents**.

👤 User-Level Files (in `~`)

File	When it's executed	Purpose
<code>~/.bashrc</code>	On interactive non-login shell (e.g., new terminal tab)	Sets aliases, functions, environment variables for terminal
<code>~/.bash_profile</code> or <code>~/.profile</code>	On interactive login shell (e.g., SSH, first terminal login)	One-time session config: sets <code>\$PATH</code> , env vars, calls <code>.bashrc</code>

File	When it's executed	Purpose
<code>~/.bash_logout</code>	When you logout of an interactive login shell	Cleanup commands like clearing terminal, unmounting

  means the user's home directory, e.g., `/home/username/`

System-Level Equivalents

File	Description
<code>/etc/profile</code>	System-wide equivalent of <code>.bash_profile</code> . Runs for all users on login shells.
<code>/etc/bash.bashrc</code>	System-wide equivalent of <code>.bashrc</code> . Executes for all users in interactive non-login shells.
<code>/etc/environment</code>	System-wide environment variable definitions. Doesn't support shell functions or logic.
<code>/etc/profile.d/*.sh</code>	Directory for additional shell scripts sourced during login from <code>/etc/profile</code> .

Typical Execution Flow

For Login Shell (like SSH or first login):

1. `/etc/profile`
 - Loads scripts from `/etc/profile.d/`
2. `~/.bash_profile` (or `~/.profile`)
 - Usually sources `~/.bashrc`

bash

```
# Common inside .bash_profile
if [ -f ~/.bashrc ]; then
```

```
. ~/.bashrc  
fi
```

For Non-login Interactive Shell (like a new terminal tab):

- `~/.bashrc`
- Also loads `/etc/bash.bashrc` first

When you logout:

- `~/.bash_logout` is executed
 - Often clears screen: `clear`

TL;DR Summary

Shell Type	Files Loaded
Login Shell	<code>/etc/profile</code> → <code>~/.bash_profile</code> → <code>~/.bashrc</code>
Non-login Shell	<code>/etc/bash.bashrc</code> → <code>~/.bashrc</code>
Logout	<code>~/.bash_logout</code>

Let me know if you want a visual boot/login flow or how to customize these for security or dev environments.

[Generate a PDF with all contents in this chat](#)

Here is the PDF containing the summary of all Linux-related questions and answers from this chat:

👉 [Download Linux_Chat_Summary.pdf](#)