Linux preparation for Capgemini

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_----

Roles & responsibilities:

In my current organization, we have to maintain 2000+ servers in both prod & non-prod environment

Coming to my daily tasks:

1.installing & managing the linux servers

2.we work on user a/c management-------creating a user, adding to the groups, modifying the users name, locking and unlocking the users account.& deleting the user as per the requests.

3.and, we work on file system management----by adding the disk, work on LVM, mounting & unmounting the fs. & also we extending the swap space

4.we handle reuests like give the permissions, if any user is unable to open the files etc

5. Automated patching across 2000+ servers using Ansible, reducing manual effort by 70%.

6.I wrote few shell scripts to automate the repetative tasks like taking the backups etc



 Wrote 10+ shell scripts to automate user and backup tasks, improving efficiency and reducing errors.

7.monitoring the servers

**Monitoring and Troubleshooting: the servers**

* Monitoring server health using tools like top, htop, vmstat, iostat, and netstat.
* Troubleshooting performance issues, system crashes, and logs analysis (/var/log/, journalctl).
* Addressing CPU, memory, disk space, and network bottlenecks.

**🔸 CPU Bottlenecks**

**Symptoms**: High CPU utilization, slow processing, frequent spikes.

**How to Address**:

* **Identify processes** using high CPU (top, htop, ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu)
* **Optimize code/configurations** of high-CPU applications
* **Distribute load** (e.g., load balancing, parallel processing)
* **Upgrade CPU resources** or scale horizontally (add more servers)

**🔸 Memory Bottlenecks**

**Symptoms**: High memory usage, swapping, out-of-memory (OOM) errors.

**How to Address**:

* Use free -m, vmstat, top, or htop to check usage
* **Identify memory-hungry processes**
* **Optimize applications** for better memory handling
* **Add more RAM** or optimize cache settings (e.g., for databases)
* **Use swap space** cautiously — it's slow compared to RAM

**🔸 Disk Space Bottlenecks**

**Symptoms**: Full disks, I/O wait, slow read/write.

**How to Address**:

* Use df -h, du -sh \*, and iostat to check usage and I/O
* **Clean up unused files/logs**
* **Use faster disks** (SSD vs HDD)
* Implement **RAID or storage tiering**
* **Optimize database storage** (archiving, indexing)

**🔸 Network Bottlenecks**

**Symptoms**: High latency, packet loss, timeouts.

**How to Address**:

* Use tools like iftop, netstat, iperf, ping, traceroute
* Check for **bandwidth saturation or errors**
* **Optimize network configurations** (MTU, offloading)
* **Upgrade NICs**, switches, or increase bandwidth
* Use **CDNs, load balancers**, or **traffic shaping**

**8. Documentation and Reporting:**

* Maintaining documentation for system changes, procedures, and SOPs.
* Preparing regular uptime, incident, and patching reports.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Linux interview questions  
  
📍Basic Level:  
  
What is Linux?

Linux is an open-source, Unix-like operating system used for servers, desktops, and embedded systems. It’s known for its stability, security, and is widely used in enterprise environments.  
Briefly explain the Linux operating system and its main components.  
What is the difference between Linux and UNIX?

Linux is an open-source operating system inspired by Unix, and it can run on a wide range of hardware platforms. Unix is a proprietary OS originally developed in the 1970s, mainly used in specialized enterprise systems.  
Discuss the differences between these two operating systems, including their origins, structure, and usage.  
What is the kernel in Linux?

The kernel is the core component of the Linux operating system that manages hardware resources and system calls. It acts as a bridge between user applications and the underlying hardware.  
Explain the role of the Linux kernel and its responsibilities in the system.

The Linux kernel is the core of the operating system that directly interacts with the hardware. Its main responsibilities include:

1. **Process Management** – Handles creation, scheduling, and termination of processes.
2. **Memory Management** – Allocates and manages RAM efficiently among processes.
3. **Device Management** – Controls communication with hardware devices using drivers.
4. **File System Management** – Provides access to data on storage devices via supported file systems.
5. **System Calls & Security** – Offers an interface for user applications to interact with the system and enforces permissions and access control.

What are the different types of file permissions in Linux?  
  
📍Intermediate Level:  
  
What are runlevels in Linux?

Runlevels are predefined states that define what services and processes should be running on a Linux system. Each runlevel represents a different mode of operation, such as:

* **0** – Halt (system shutdown)
* **1** – Single-user mode (maintenance)
* **3** – Multi-user mode without GUI
* **5** – Multi-user mode with GUI
* **6** – Reboot

Explain the different runlevels (e.g., 0 to 6) and their functions in system startup and shutdown.  
How do you search for a string in a file?  
Discuss commands like grep to search for text within files.  
What is the difference between hard links and symbolic links?  
Define and contrast hard links and soft (symbolic) links in Linux.

**Is it necessary to create hard or soft links in Linux? (Interview Answer):**

It’s not always necessary, but creating **hard or soft links** is useful for easier file management, backup references, or creating shortcuts. They help avoid duplication and maintain consistent access to files from multiple locations.

Explain the concept of pipes and redirection in Linux.  
Discuss how pipes (|) and redirection (>, >>, <) work in Linux to manage input and output.  
What are cron jobs? How do you schedule a job in Linux?  
Explain the cron daemon and how to schedule tasks using the crontab command.  
How would you check system resource usage in Linux?

You can check system resource usage using commands like:

* top or htop – for real-time CPU, memory, and process usage.
* free -m – to view memory usage.
* df -h – to check disk space usage.
* vmstat, iostat, and sar – for detailed performance metrics.

These tools help monitor system health and identify bottlenecks.

Discuss tools like top, htop, vmstat, free, and iostat for monitoring system resources.  
Explain the difference between fork() and exec() system calls.

What is zombie

A **zombie** process is a process that has completed execution but still has an entry in the process table because its parent process has not yet read its exit status. These processes have released most of their resources, but the process ID (PID) remains, consuming system table entries. Zombie processes typically occur when the parent fails to call wait() to collect the exit status of the terminated child.

Zombie processes can be identified using the ps aux | grep Z command. They do not consume CPU or memory, but if left unchecked, they can cause a resource leak in the process table.

Provide an explanation of these two fundamental system calls in process creation and execution.  
  
📍Advanced Level:  
  
What is the difference between a process and a thread in Linux?

A **process** is an independent program with its own memory space, while a **thread** is a lightweight unit of execution within a process that shares memory and resources with other threads in the same process. Threads are faster and more memory-efficient compared to processes.  
Describe the difference in terms of execution, memory sharing, and management.  
What is the Linux boot process?  
Explain the steps involved, including BIOS/UEFI, bootloader (GRUB), kernel loading, and init/systemd.

 **Bootloader (GRUB)**:

* GRUB (Grand Unified Bootloader) is responsible for loading the kernel. It presents a boot menu to the user, allowing the selection of a kernel or operating system. GRUB loads the kernel into memory and passes control to it.

 **Kernel Loading**:

* The kernel initializes the system by detecting hardware, mounting the root filesystem, setting up memory management, and launching the init process (or systemd on modern systems).

 **Init/Systemd**:

* init (or systemd in newer Linux distributions) is the first user-space process. It reads configuration files (like /etc/inittab for init or /etc/systemd/system for systemd) to start essential services, set up the user environment, and manage the system’s lifecycle.

Discuss memory management concepts like virtual memory, swapping, memory paging, and allocation.

 **Virtual Memory**:

* Virtual memory allows processes to have their own address space, enabling efficient multitasking by using both physical memory and disk space. It ensures isolation and extends the usable memory beyond RAM.

 **Swapping**:

* Swapping moves inactive process pages from RAM to disk (swap space) when physical memory is full. This helps manage memory but can slow down the system if excessive swapping occurs.

 **Memory Paging**:

* Paging divides memory into fixed-size blocks (pages) and allocates them to processes. It allows non-contiguous memory allocation, reducing fragmentation and improving memory management.

 **Memory Allocation**:

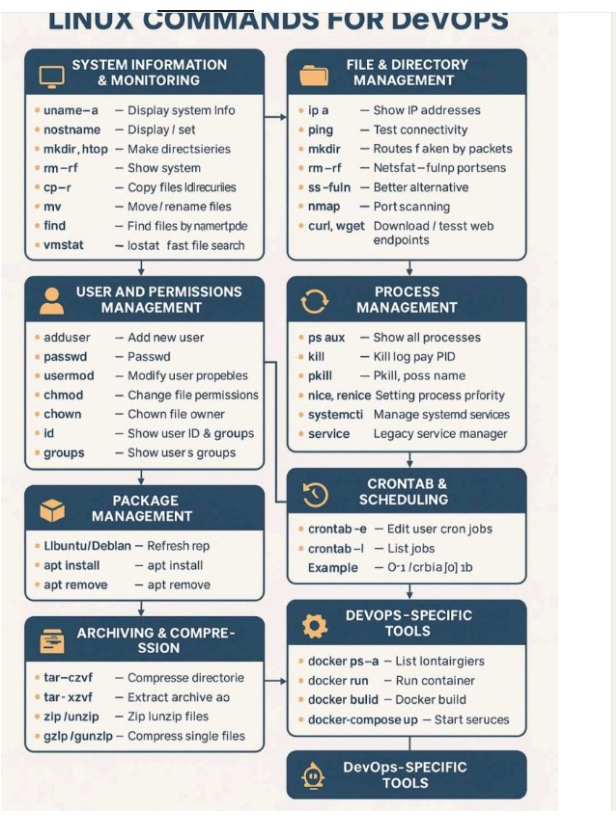
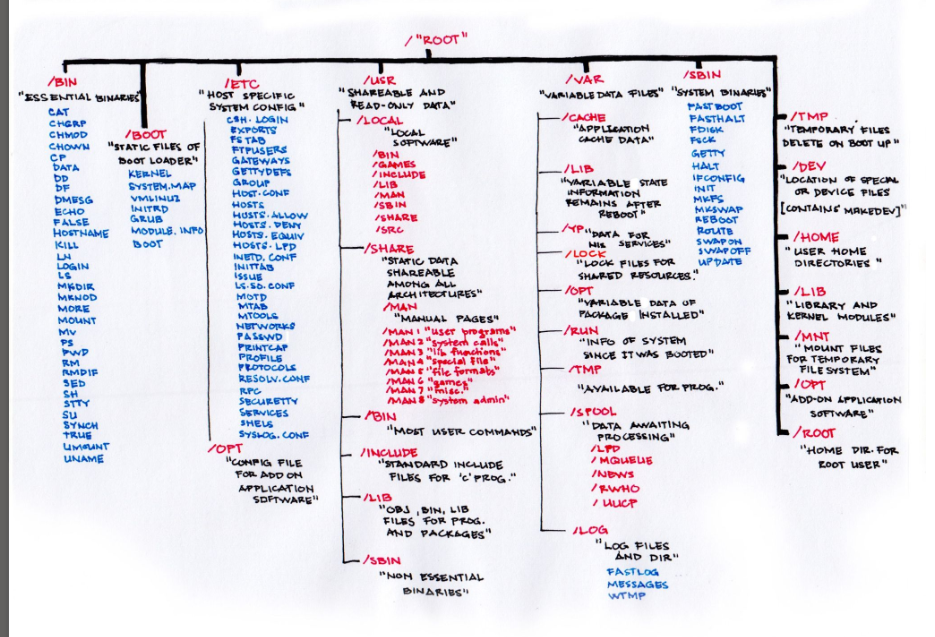
* Memory allocation assigns memory blocks to processes, with the kernel using techniques like buddy systems and slab allocators for efficient memory use. User processes typically rely on malloc for dynamic memory allocation.

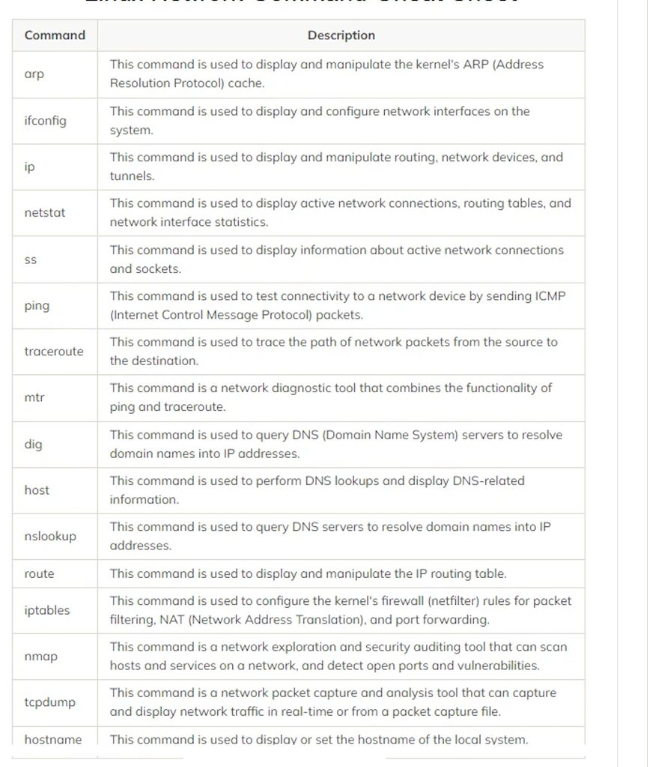
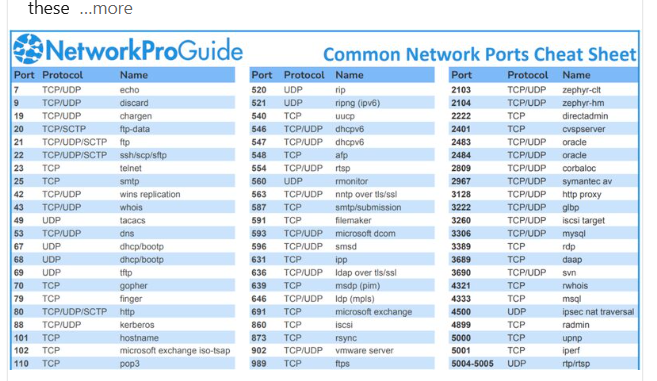
Explain the concept of Linux namespaces.

Linux namespaces provide a mechanism for isolating resources and processes within a system. Each namespace allows a process to have its own view of certain system resources, creating a virtual environment that appears independent of others. The main types of namespaces in Linux are:

1. **PID namespace** – Isolates process IDs, allowing processes to have their own process ID space.
2. **Mount namespace** – Provides separate views of the filesystem, enabling different mounts for different processes.
3. **Network namespace** – Allows processes to have their own network interfaces, IP addresses, and routing tables.
4. **UTS namespace** – Isolates hostname and domain name, enabling different hostnames for different processes.
5. **IPC namespace** – Provides isolated inter-process communication (IPC) resources like semaphores and message queues.
6. **User namespace** – Allows different user and group IDs to be mapped inside the namespace, providing security isolation.
7. **Cgroup namespace** – Isolates control group settings and resource limits, enabling resource management for different processes.

Discuss how namespaces are used to isolate resources in Linux, such as in containers.  
How does the ls command work in Linux?  
Explain how ls lists files and directories, including options for detailed listings  
Explain how ldd is used to show shared library dependencies of a program.  
What is a system call?  
Define system calls and provide examples.  
Discuss tools like top, ps, vmstat, htop, and strace for diagnosing performance issues.  
Explain how Linux handles file systems.  
Discuss file system types (ext4)  
What is a kernel panic?



* 1. What is the difference between TCP and UDP protocols?  
     TCP: Connection-oriented, reliable, slower, guarantees delivery, suitable for applications requiring error-free data transmission (e.g., HTTP, FTP).  
     UDP: Connectionless, unreliable, faster, doesn't guarantee delivery, suitable for applications where data loss is acceptable and speed is prioritized (e.g., streaming media, DNS).  
       
     2) Explain the concept of IP addressing.  
     IPv4: 32-bit addresses, four octets separated by dots   
     IPv6: 128-bit addresses, hexadecimal format   
     Classes: A, B, C, D, E.  
     Subnetting: Dividing a network into smaller subnets.  
       
     3) What is the role of a gateway in a network?  
     A gateway is a device (router, firewall) that connects two different networks and acts as a bridge between them. It forwards packets between networks based on their destination IP addresses.  
       
     4) What is DNS and its role in networking?  
     DNS (Domain Name System) translates human-readable domain names (e.g., [invalid URL removed]) into machine-readable IP addresses.  
     Linux Networking Commands  
       
     5) How do you configure network interfaces in Linux using ifconfig or ip?  
     Use ifconfig or ip addr to view and modify network interface settings (e.g., IP address, netmask, up/down state).  
       
     6) What is the purpose of the ping command?  
     ping tests network connectivity to a specific host by sending ICMP echo requests and waiting for replies.  
       
     6) How do you view network routes and modify the routing table?  
     Use route to view and manipulate the routing table.  
       
     7) What is the netstat command used for?  
     netstat displays network connections, routing tables, interface statistics, and more.  
       
     8) How do you troubleshoot network connectivity issues in Linux?  
     Check interface configuration: Ensure the interface is up and has the correct IP address and netmask.  
     Ping the gateway  
     Check DNS resolution  
     Inspect firewall rules  
     Analyze network logs
  2. d/f b/w redhat 5,6,7
*  **System Initialization**: **SysVinit** (Red Hat 5/6) vs **systemd** (Red Hat 7).
*  **Kernel Versions**: Red Hat 5 (2.6.18), Red Hat 6 (2.6.32), Red Hat 7 (3.10).
*  **File Systems**: **ext3** in Red Hat 5/6, **XFS** as the default in Red Hat 7.
*  **Virtualization**: Enhanced KVM and Docker support in Red Hat 7.
*  **Package Management**: Improved **YUM** handling across versions, with better repository management in Red Hat 6 and 7.
  1. 9) What is the traceroute command and how is it used?  
     traceroute traces the path packets take to reach a destination, showing intermediate hops and the time it takes for packets to reach each hop.  
     Advanced Topics  
       
     10) What is the difference between TCP and UDP port scanning?  
     TCP port scanning attempts to establish a connection with a specific port, while UDP port scanning sends UDP datagrams to a port and listens for responses. 📌Linux basic commands and questions:  
       
     File and Directory Management  
     ls: List directory contents  
     cd: Change directory  
     mkdir: Create a directory  
     rmdir: Remove an empty directory  
     rm: Remove files  
     cp: Copy files  
     mv: Move or rename files/directories  
     touch: Create an empty file  
     cat: Display file contents  
     head: Display first few lines of a file  
     tail: Display last few lines of a file  
     grep: Search for patterns in files  
     find: Find files and directories  
     locate: Quickly find files (requires updated database)  
     System Information  
     df: Display disk space usage  
     du: Display disk space usage of files and directories  
     free: Display memory usage  
     top: Display system processes  
     ps: Display process information  
     kill: Terminate a process  
     uptime: Display system uptime  
     whoami: Display current user  
     date: Display current date and time  
     User Management  
     useradd: Create a new user  
     userdel: Delete a user  
     passwd: Change password  
     su: Switch user  
     sudo: Execute commands as another user  
     Networking  
     ping: Test network connectivity  
     ifconfig: Configure network interfaces  
     netstat: Display network connections  
     ssh: Secure Shell for remote login  
     Linux Cheat Sheet 2: Advanced Commands  
     File System Management  
     mount: Mount file systems  
     umount: Unmount file systems  
     fsck: Check and repair file systems  
     chown: Change file ownership  
     chmod: Change file permissions  
     Package Management  
     apt-get: Install, update, and remove packages (Debian/Ubuntu)  
     yum: Install, update, and remove packages (Red Hat/CentOS/Fedora)  
     dnf: Install, update, and remove packages (Fedora)  
     Scripting  
     bash: Bourne Again SHell (default shell)  
     sh: Bourne Shell  
     sed: Stream Editor  
     awk: Pattern scanning and text processing language  
     Compression and Archiving  
     tar: Create, extract, and manipulate archive files  
     gzip: Compress and decompress files  
     bzip2: Compress and decompress files (better compression than gzip)  
     Searching and Text Processing  
     grep: Search for patterns in files  
     sed: Stream Editor  
     awk: Pattern scanning and text processing language  
     cut: Extract columns from data  
     sort: Sort data  
     uniq: Report or omit repeated lines  
     Linux Interview Questions  
     What is the difference between hard links and symbolic links?  
     Explain the concept of the Linux kernel.  
     What are the different file system types in Linux?  
     How do you find the PID of a running process?  
     What is the purpose of the cron job?  
     How do you redirect output to a file?  
     What are the different levels of file permissions in Linux?  
     Explain the concept of a shell.  
     How do you find the size of a directory?  
     What is the difference between ls -l and ls -a?  
     How do you create a new user in Linux?  
     What is the purpose of the sudo command?  
     How do you check disk space usage? 1. What are the main networking tools in Linux?  
       
     ifconfig: Used to configure and display network interfaces. Deprecated, but still widely used.  
     ip: More modern tool for networking configuration (replaces ifconfig).  
     netstat: Displays network connections, routing tables, and interface statistics. Deprecated, replaced by ss in most cases.  
     ss: Utility to investigate sockets, often faster and more efficient than netstat.  
     ping: Tests connectivity to another system by sending ICMP packets.  
     traceroute: Traces the route packets take to reach a destination.  
     nslookup: Query DNS servers for information about domain names.  
     dig: A more advanced DNS query tool than nslookup.  
     nmap: Network exploration tool and security scanner.  
     iptables: Tool for managing firewall rules.  
     tcpdump: Network packet analyzer.  
     route: Used to view and manipulate the IP routing table.  
     hostname: Displays or sets the system’s hostname.  
       
       
     2. What are iptables chains and how do they work?  
       
     Answer: iptables uses different chains to process network packets. The three default chains are:  
     INPUT: For incoming network traffic to the system.  
     OUTPUT: For outgoing network traffic from the system.  
     FORWARD: For packets that are routed through the system (not intended for the system itself but forwarded to another destination).  
       
     3. What are the common network troubleshooting commands in Linux?  
       
     ping: Tests the connectivity to a host.  
     traceroute: Traces the route packets take to a network host.  
     netstat or ss: Shows active network connections and listening ports.  
     curl: Tests network services, e.g., HTTP requests.  
     dig or nslookup: Queries DNS records.  
     tcpdump: Captures network packets to analyze network traffic  
       
     4. What is the purpose of systemctl in Linux?  
       
     Start a service: sudo systemctl start <service>  
     Stop a service: sudo systemctl stop <service>  
     Enable a service to start at boot: sudo systemctl enable <service>  
     Check the status of a service: sudo systemctl status <service>  
     Reload systemd configuration: sudo systemctl daemon-reload  
       
     5. How do you configure a static IP on a RedHat-based system (e.g., CentOS)?  
       
     For a static IP configuration, you need to modify the interface configuration file in /etc/sysconfig/network-scripts/ifcfg-<interface>.  
       
     6. What is a network bridge in Linux?  
       
     sudo brctl addbr br0   
     sudo brctl addif br0 eth0   
     sudo ip addr add IP dev br0   
     sudo ip linkset br0 up  
       
     7. How do you restart networking services on a Linux system?  
       
     sudo systemctl restart networking  
     sudo systemctl restart network  
     sudo systemctl restart NetworkManager

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Resume related questions and answers: prepare well this is enough

**1. How do you manage user accounts and groups in a Linux environment?**

I use commands like useradd, usermod, userdel to manage user accounts, and groupadd, groupmod, groupdel for group management. I assign users to groups using usermod -aG. Passwords are managed using passwd. For verification, I refer to /etc/passwd, /etc/shadow, and /etc/group. I also automate account management via shell scripts and cron jobs.

**2. Can you explain the difference between hard links and soft links in Linux?**

Hard links are references to the same inode as the original file, meaning they are indistinguishable from the original file and persist even if the original is deleted. Soft links (symbolic links) are pointers to the filename, not the data; if the original file is deleted, the soft link becomes broken.

**3. What steps do you follow to apply OS patches and kernel upgrades safely?**

I first back up critical data and configurations. Then, I check available updates using yum or dnf (e.g., yum update kernel). After applying updates, I reboot the system and verify with uname -r. I always test patches in a staging environment before applying them to production servers.

**4. Describe how you would troubleshoot a server that fails to boot.**

I would start by accessing the GRUB menu and checking boot parameters. I might boot into rescue or recovery mode. I would inspect logs like /var/log/messages, /var/log/dmesg, and journalctl to identify hardware or service failures. I also check disk status using fsck, and verify if important mount points like /boot are intact.

**5. What tools do you use for monitoring CPU, memory, and disk I/O?**

I use top, htop, vmstat, iostat, free, sar, and uptime for performance monitoring. These help identify resource bottlenecks and abnormal behavior. I’ve also set up Nagios Core for centralized monitoring and alerting.

**6. Can you walk through a shell script you've written to automate a system task?**

Yes. I’ve written a shell script to back up /var/www/html daily:

bash

CopyEdit

#!/bin/bash

BACKUP\_DIR="/backup/$(date +%F)"

mkdir -p "$BACKUP\_DIR"

cp -r /var/www/html "$BACKUP\_DIR"

find /backup -type d -mtime +7 -exec rm -rf {} \;

This script automates backup and deletes backups older than 7 days.

**7. How do you schedule regular scripts using cron? Can you give a sample cron job?**

I use crontab -e to schedule jobs. A sample:

ruby

CopyEdit

0 2 \* \* \* /usr/local/bin/db\_backup.sh

This runs the backup script every day at 2 AM.

**8. What is the difference between at, cron, and systemd timers?**

* cron is for repetitive tasks.
* at runs a command once at a specific time.
* systemd timers are newer, systemd-based scheduled units, offering more flexibility and logging.

**9. How do you create and manage logical volumes using LVM?**

1. Create physical volume: pvcreate /dev/sdb1
2. Create volume group: vgcreate my\_vg /dev/sdb1
3. Create logical volume: lvcreate -L 10G -n my\_lv my\_vg
4. Format it: mkfs.ext4 /dev/my\_vg/my\_lv
5. Mount it and add to /etc/fstab for persistence.

**10. What steps do you take to extend a filesystem on a running server?**

1. Extend logical volume: lvextend -L +5G /dev/my\_vg/my\_lv
2. Resize the file system:
   * For ext4: resize2fs /dev/my\_vg/my\_lv
   * For xfs: xfs\_growfs /mountpoint

What is mounting?

**Mounting in Linux (Interview Answer):**

Mounting is the process of **attaching a filesystem** (like a hard drive, USB, or ISO) to a directory in the Linux directory tree, making its contents accessible to the system.

What is umounting

**Unmounting** is the process of **safely detaching a mounted filesystem** from the Linux directory tree using the umount command.

**What is f.s**

**f.s is used to store data in a structured manner is called f.d**

**d/f types of f.s**

**ext3,ext4.xfs**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**11. How do you check disk usage and inode usage on Linux?**

* Disk usage: df -h
* Inode usage: df -i
* Detailed per-directory usage: du -sh \*

**12. Explain how you configure and secure an SSH server.**

I configure SSH in /etc/ssh/sshd\_config:

* Disable root login: PermitRootLogin no
* Allow only specific users: AllowUsers user1
* Change default port
* Use key-based authentication instead of passwords

I then restart SSH: systemctl restart sshd

**13. What is the purpose of a DNS server, and how do you configure one in Linux?**

A DNS server resolves domain names to IPs. I install bind, configure zone files in /var/named/, and set up /etc/named.conf. I test with dig or nslookup, and ensure the service is running with systemctl start named.

**14. Describe the process of setting up and sharing a directory using Samba and NFS. What are the key differences?**

* **NFS** (Linux to Linux): Install nfs-utils, add export in /etc/exports, run exportfs -a, and start nfs-server.--2049
* **Samba** (Linux to Windows): Install samba, edit /etc/samba/smb.conf, create a share section, add user with smbpasswd, and restart smb.

**Differences:**

* NFS is native to Unix/Linux; Samba is for Windows interoperability.
* Samba uses user authentication; NFS can use IP-based access.

**15. How do you troubleshoot network-related issues on a Linux server?**

I use:

* ping to check reachability
* traceroute for routing path
* netstat, ss, and lsof to inspect open ports and listening services
* tcpdump or wireshark for packet capture
* Check /etc/resolv.conf, /etc/hosts, and firewall rules (iptables/firewalld)

**16. Which commands do you use to monitor network performance and traffic?**

* iftop, iptraf, nload for live network traffic
* netstat or ss for connection status
* sar -n DEV for historical data
* tcpdump for packet-level analysis

**17. How have you used Nagios Core in your previous roles? Can you describe its architecture?**

I installed and configured Nagios Core to monitor servers and services. Its architecture includes:

* **Nagios server**: Central monitoring
* **Plugins**: Check services (e.g., CPU, disk, HTTP)
* **NRPE/NSClient++**: For remote checks
* **Notifications**: Email alerts via SMTP

**18. How do you handle log rotation and management in Linux?**

I use logrotate, which is configured via /etc/logrotate.conf and /etc/logrotate.d/. It handles log size limits, compression, and retention. I also use journalctl for systemd logs and clear old logs with journalctl --vacuum-time=7d.

**19. How do you find and resolve recurring errors from logs?**

I analyze logs using:

* /var/log/messages, dmesg, journalctl
* Search using grep, awk, less
* Monitor log trends via centralized tools or custom scripts

Once I identify a recurring pattern, I correlate it with recent changes and either fix the underlying configuration, tune parameters, or escalate if it’s hardware-related.

**20. How do you manage MySQL databases on a Linux server?**

I manage MySQL using:

* mysql CLI for queries
* mysqldump for backups
* systemctl to control the service
* my.cnf for configuration tuning (e.g., buffer sizes, max connections)  
  I also monitor slow query logs and performance schema to optimize queries.

**21. Can you describe a time when you optimized web server performance or logs?**

I optimized Apache logs by rotating them with logrotate and compressing old logs. I disabled unnecessary modules and enabled caching. I also tuned worker settings in httpd.conf for better concurrency handling under load.

**22. Tell me about a critical issue you resolved during off-hours or under pressure.**

Once, a production web server went down late at night due to disk space exhaustion. I SSHed into the server, identified large log files using du -sh, cleared old backups, and restarted the web service. I then set up a cron job to monitor disk usage and alert before space runs out.

**23. How do you stay updated with changes in Linux administration tools or best practices?**

I follow official documentation (RedHat, CentOS), blogs (like Linuxize), Reddit forums, and GitHub projects. I also test new tools and updates in a virtual lab setup to stay hands-on.

**24. How do you ensure system uptime and meet SLAs?**

I use proactive monitoring with Nagios and automate health checks with scripts. I perform regular patching during planned maintenance windows. I maintain backups and follow a change management process to prevent unplanned downtimes. Critical services are configured with alerts and failover mechanisms when needed.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
user a/c management possible issues:

**: How do you manage user accounts and groups in a Linux environment, and how do you troubleshoot related issues?**

**Answer:**

I manage user accounts and groups in Linux primarily using command-line utilities such as:

* useradd, usermod, userdel – to add, modify, or delete users.
* groupadd, groupmod, groupdel – to manage groups.
* passwd – to set or reset passwords.
* id, groups, who, w, and last – to verify user identity and activity.
* Configuration files like /etc/passwd, /etc/shadow, /etc/group, and /etc/gshadow for manual verification or troubleshooting.

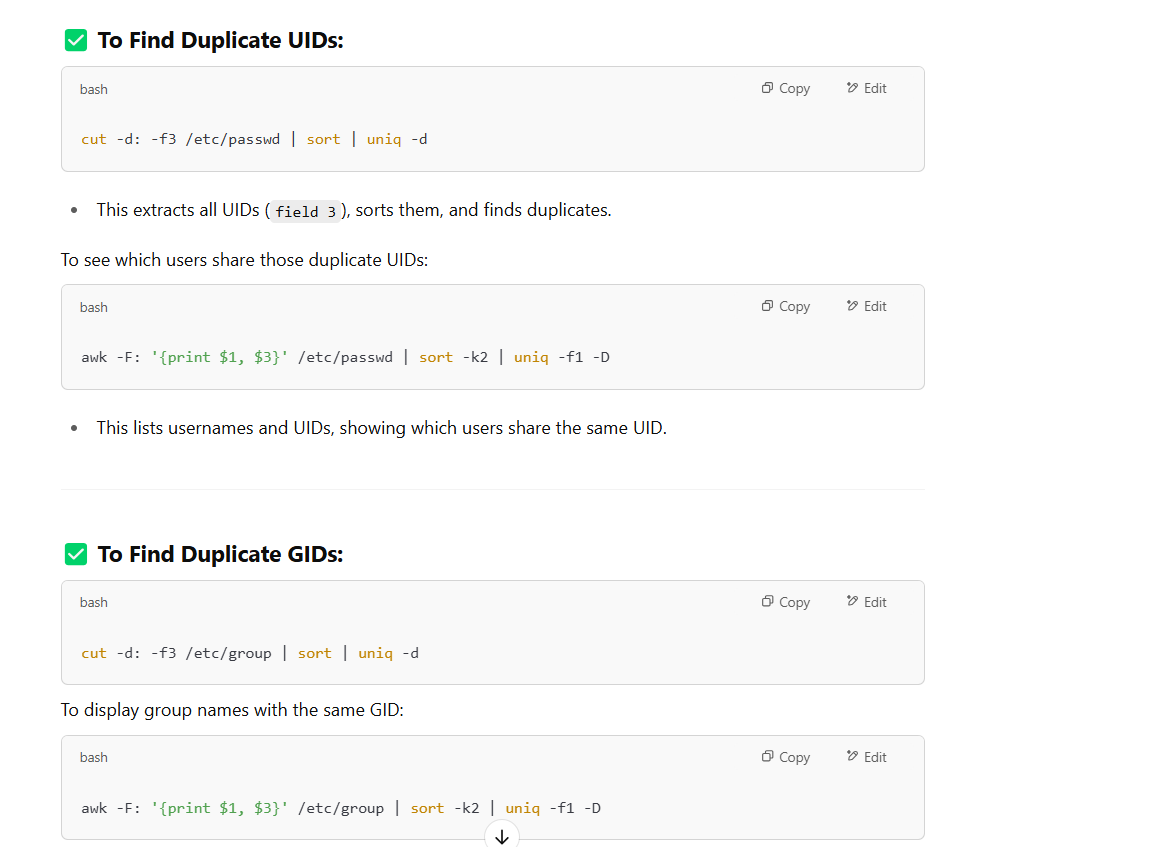
**Common Issues in User/Group Management:**

1. **User unable to log in:**
   * Cause: Shell not defined correctly, account expired, password expired, or user not in proper group.
   * Fix: Check /etc/passwd for shell; use chage -l username for account expiry; reset password with passwd.
2. **Permission denied errors:**
   * Cause: Incorrect group membership or file ownership.
   * Fix: Use ls -l and id username to verify; adjust with chown, chmod, or usermod -aG.
3. **Group changes not reflected:**
   * Cause: User needs to log out and log back in for group membership changes to apply.
   * Fix: Restart the session or use newgrp.
4. **UID or GID conflicts:**
   * Cause: Duplicate UID/GID due to manual edits.
   * Fix: Use getent passwd and getent group to verify; resolve conflicts by adjusting /etc/passwd and /etc/group.
5. **Home directory not created:**
   * Cause: -m option not used with useradd.
   * Fix: Manually create with mkdir /home/username && chown username:username /home/username.

**Troubleshooting Approach:**

1. **Check system logs:**
   * Review /var/log/secure, /var/log/auth.log, and journal logs using journalctl to trace login attempts and failures.
2. **Verify user information:**
   * Use getent passwd username and id username to ensure the account exists and has the correct attributes.
3. **Audit configuration files:**
   * Check for syntax or formatting errors in /etc/passwd, /etc/shadow, /etc/group.
4. **Test permissions and groups:**
   * Use ls -l, groups, and sudo -l to verify access rights.
5. **Use debugging tools:**
   * strace, su - username, or even ssh -vvv can help pinpoint issues during authentication or session initiation.

**Other Operational Issues**

* **Cron jobs or scripts fail due to user not having shell or PATH configured.**
* **User account accidentally deleted or locked.**
* **No sudo privileges set for administrative users.**
* 

**✅ Best Practices to Avoid Issues**

* Always use commands (useradd, usermod, passwd) instead of manually editing system files.
* Regularly audit /etc/passwd, /etc/shadow, /etc/group.
* Use chage, id, getent for validation.
* Apply clear policies for account expiration and password aging.
* Automate user creation/deletion with validated scripts to minimize human error.

Runlevels---**Runlevel** is a preset mode in Linux that defines what services and processes should be running on the system. Each runlevel represents a specific system state, like shutdown, single-user mode, multi-user mode, or graphical interface.344614

0-suhdown

1-single usermode

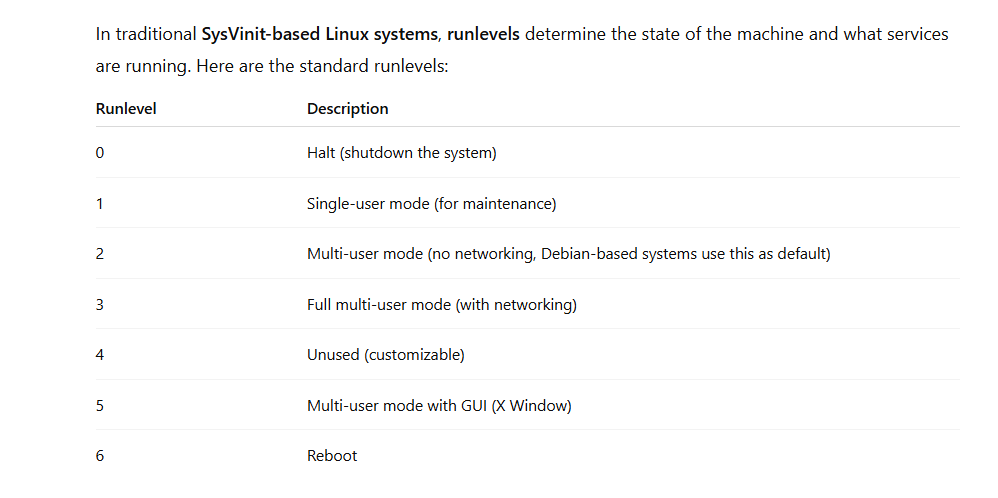
2-multi user mode w.o n/w

3-multi user mode with n/w

4-not used

5-multi usermode with GUI

6.Reboot



\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

How to check is it physical m/c or virtual m/c:

d/f commands to find it

1.lshw

2. systemd-detect-virt

Output for physical machine: none

Output for VM: kvm, vmware, oracle, etc.

3. sudo dmidecode -s system-product-name

**To confirm further (optional):**

**✅ Check with dmesg:**

dmesg | grep -i xen

**✅ Check if you're on AWS:**

curl -s http://169.254.169.254/latest/meta-data/

If you get a list of metadata (like instance-id, ami-id), then you're on **AWS EC2**.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
"VMware is a virtualization technology that allows multiple virtual machines (VMs) to run on a single physical server by sharing the server's hardware resources."

RAID—Raid is an Redunant Array of Independent disk

It is used to combine multiple physical disk drives intoa single unit for data redundancy

Raid0---- Data is split evenly across two or more disks,if one disk fails, all data is lost

Raid1—Data is copies acress 2 or more disks,

If 1 disk fails data , other has an exacat copy of data

Raid5--- If one disk fails, data can be rebuilt using the parity information.

We use **RAID 10 (1+0) – Mirrored Stripes**

A combination of RAID 1 (mirroring) and RAID 0 (striping). Data is mirrored across pairs of disks, then striped across multiple pairs

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

"df -h is a command used to check disk space usage across all mounted file systems or entire partition in a human-readable format."

df -h → shows filesystem space (entire partition)

6feilds

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
"du -sh is a command used to check disk space usage of a particular directory in a human-readable format."

du -sh /home/user/Documents

o/p: 1.4G /home/user/Documents

Inode:

"An inode is a data structure of a file system that stores metadata about a file or directory, such as its size, ownership, permissions, and location on the disk."

"Inum (inode number) is a unique identifier assigned to each file or directory in a Linux file system. It points to the inode that stores that file's metadata."

i-For copying 1location to another location inum is d/f

ii-for moving 1location to another location inum is same

iii.for soft link---ln -s source destination----inum is d/f

iv.for hardlink—ln source destination---inum is same

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
ps command is used to find the active process running on a linux sysytem

 **ps -e or ps -A**: Lists all processes running on the system.

 **ps -f**: Provides full-format listing with more details.

 **ps aux**: Shows all processes for all users, including background processes, with detailed information (including memory and CPU usage).

 **ps -u username**: Displays processes running for a specific user.

UID PID, PPID, C---cpu utilization, stime—start time process,TTY, time, CMD

8feilds:

Pstree -u (username)

Ps -u username ---forest

Pstree -p--------------show in tree format



 **UID**: User ID of the process owner.

 **PID**: Process ID (unique identifier for each process).

 **PPID**: Parent Process ID (the process that started this process).

 **C**: CPU utilization.

 **STIME**: Start time of the process.

 **TTY**: Terminal associated with the process (if any).

 **TIME**: Total CPU time used by the process.

 **CMD**: Command that started the process.

TTY--?---no direct associated terminal

1.Ps -ef----all process with detailed full format

2.Ps -aux---The -a option shows processes of all users, the -u option includes user-oriented information, and the -x option shows processes not attached to a terminal

3-ps -aux | grep <process\_name>--process matching the specified name

**4-ps -eo pid,ppid,cmd,%mem,%cpu**

* **Purpose:** Customizes the output format to display specific columns (e.g., PID, parent PID, command, memory, CPU usage).

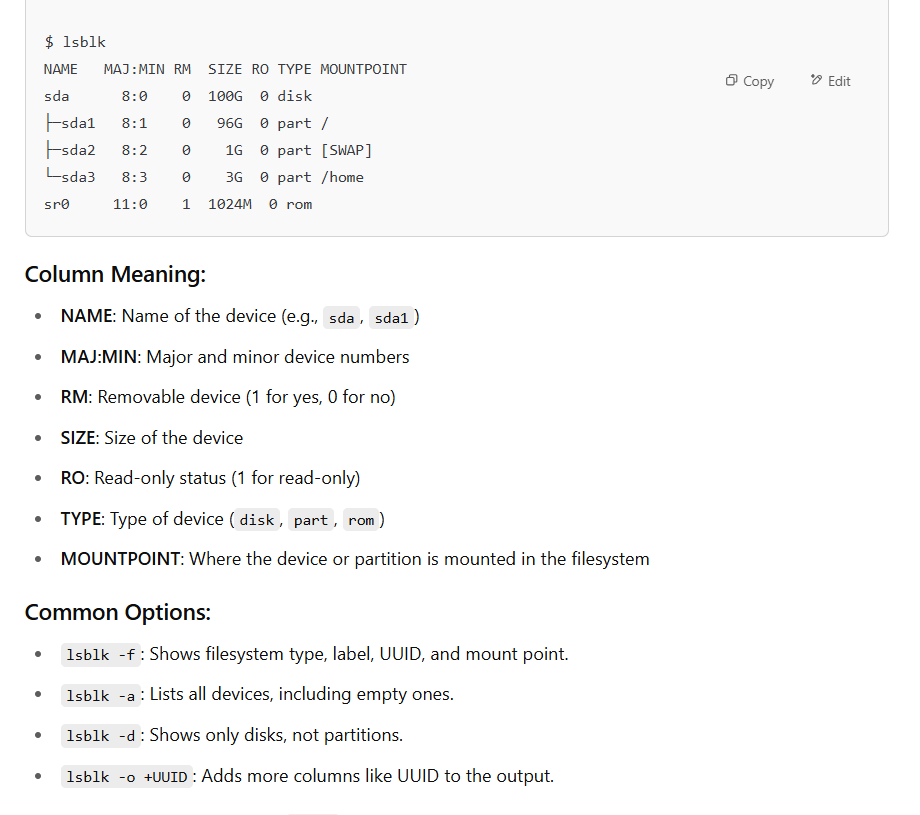
while the -o option allows you to specify the **output format** by selecting particular columns (e.g., PID, CPU, memory, command, etc.).

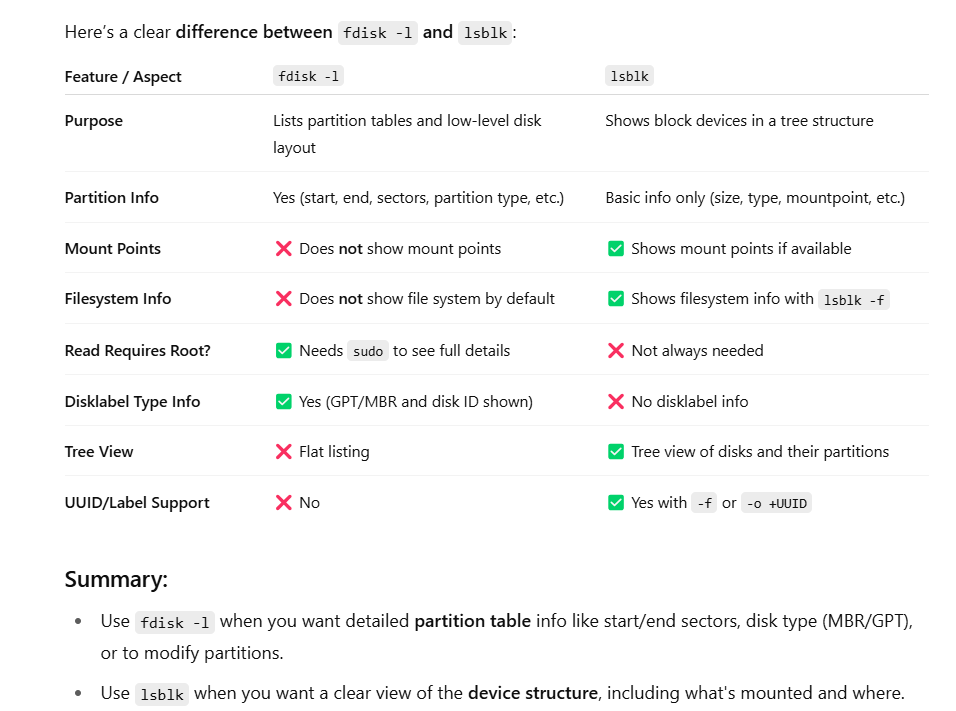
5-**ps --sort=%cpu**

* **Purpose:** Sorts processes by CPU usage.

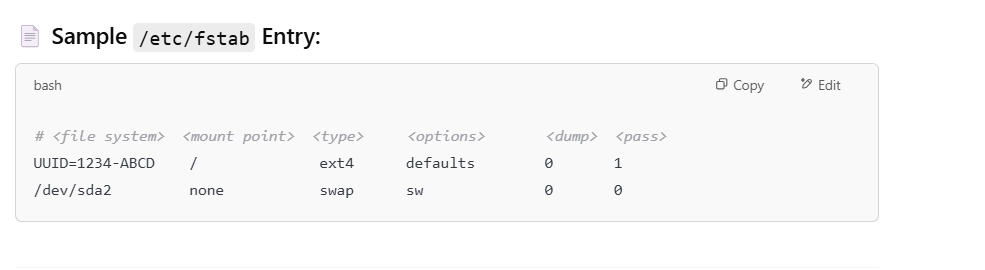
6-ps -l—long listing

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-----





/etc/fstab------filesystem\_\_



What is sudoers

**sudoers** file is a special configuration file in Linux, located at **/etc/sudoers**

It defines the **privileges of users and groups** for executing commands as the **superuser (root)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
check ip address of our system**

**1.ip addr show**

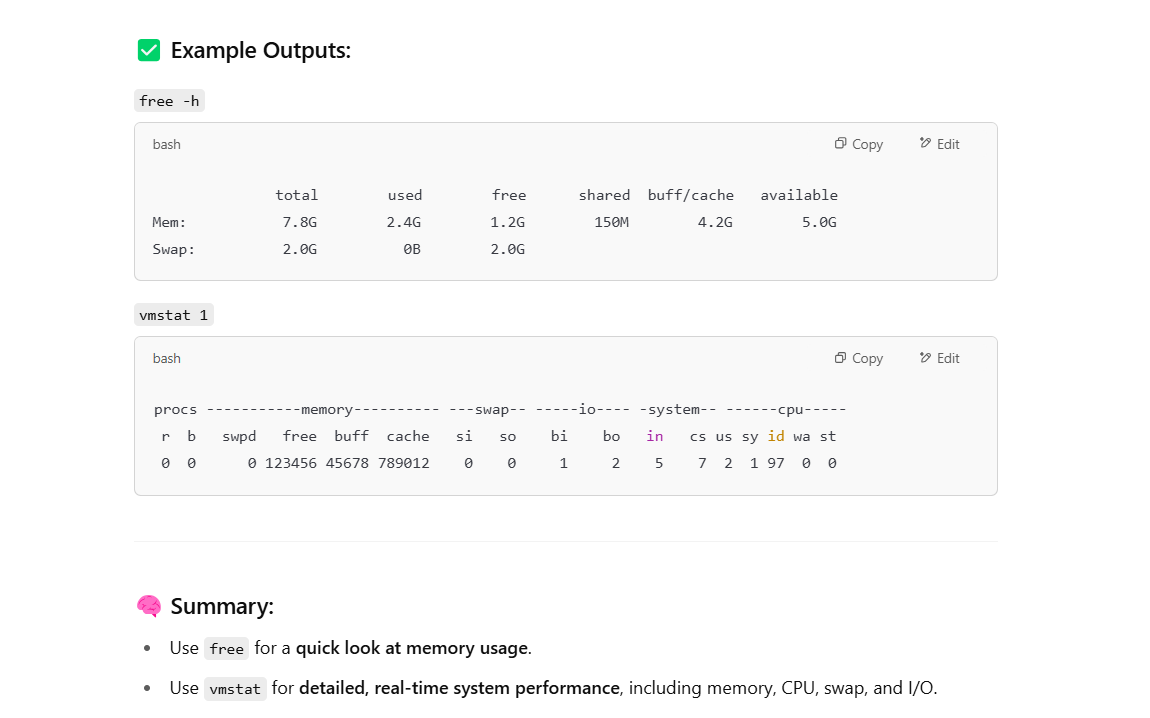
**2.hostname -I**

**Sleep---it is a command used to pause the execution for certain period of time---1s, 2sec etc**

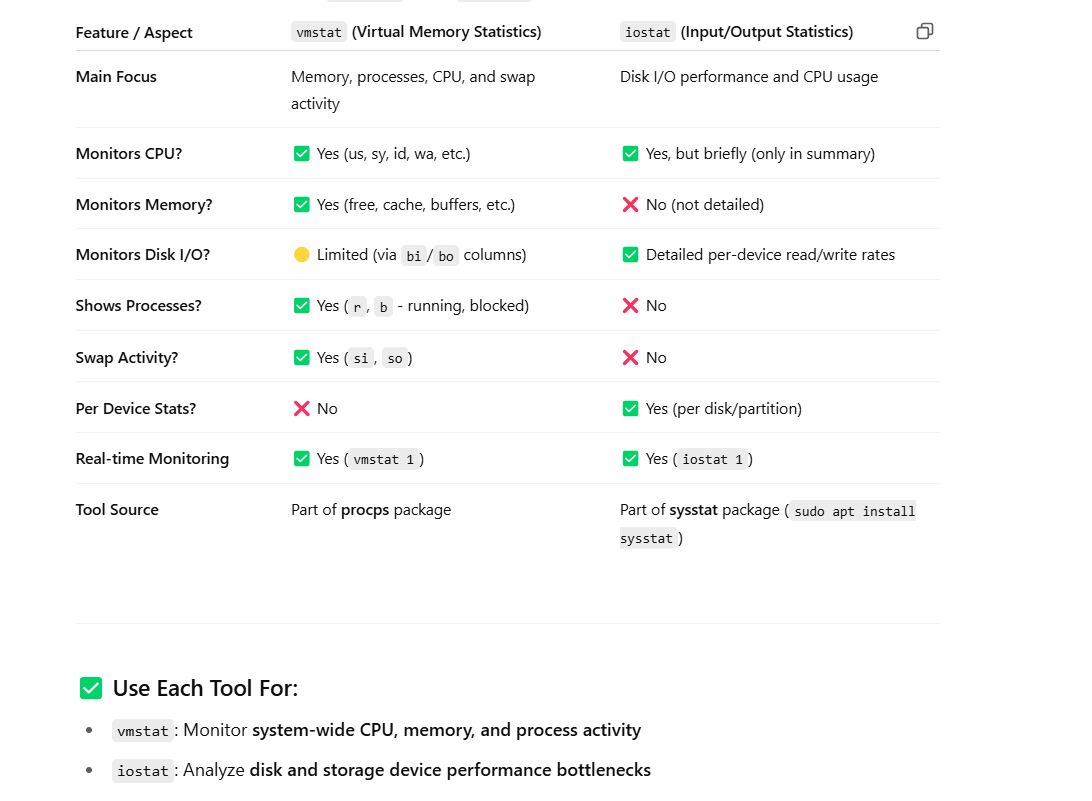
**Watch---It is a command used to excute a command at regular intervals and display the o/p in the terminal**

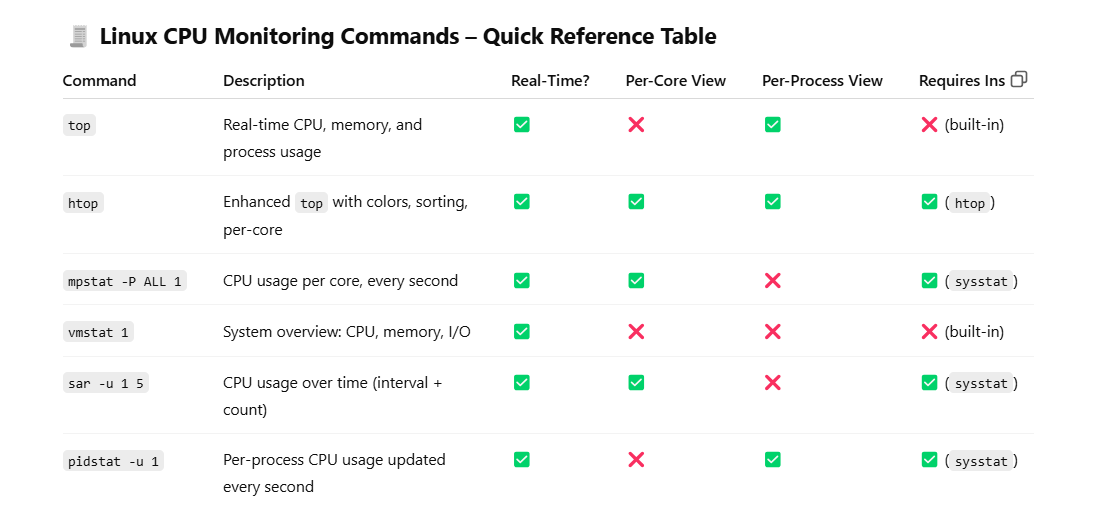
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-----**

**The free command in Linux displays memory usage information, including RAM and swap space.**



-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_--





**Linux Interview Command Cheat Sheet**

**🔹 MEMORY MONITORING**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| free -h | Show memory usage (human-readable) | free -h |
| vmstat 1 | Live memory/swap/cpu overview | vmstat 1 |
| top / htop | Live RAM/CPU usage by process | top |

**🔹 CPU MONITORING**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| top | Live CPU & memory usage per process | top |
| vmstat | CPU idle, wait, system usage | vmstat 1 |
| mpstat -P ALL | CPU usage per core (install sysstat) | mpstat -P ALL |

**🔹 DISK & I/O MONITORING**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| iostat -dx 1 | Per-device disk read/write stats | iostat -dx 1 |
| df -h | Disk space usage of partitions | df -h |
| du -sh /folder | Disk usage of a specific folder | du -sh /var/log |
| lsblk | List block devices | lsblk |

**🔹 SWAP MONITORING**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| free -h | See used/free swap space | free -h |
| vmstat | si/so = swap in/out activity | vmstat 1 |
| swapon -s | Show active swap devices | swapon -s |
| TOP |  |  |

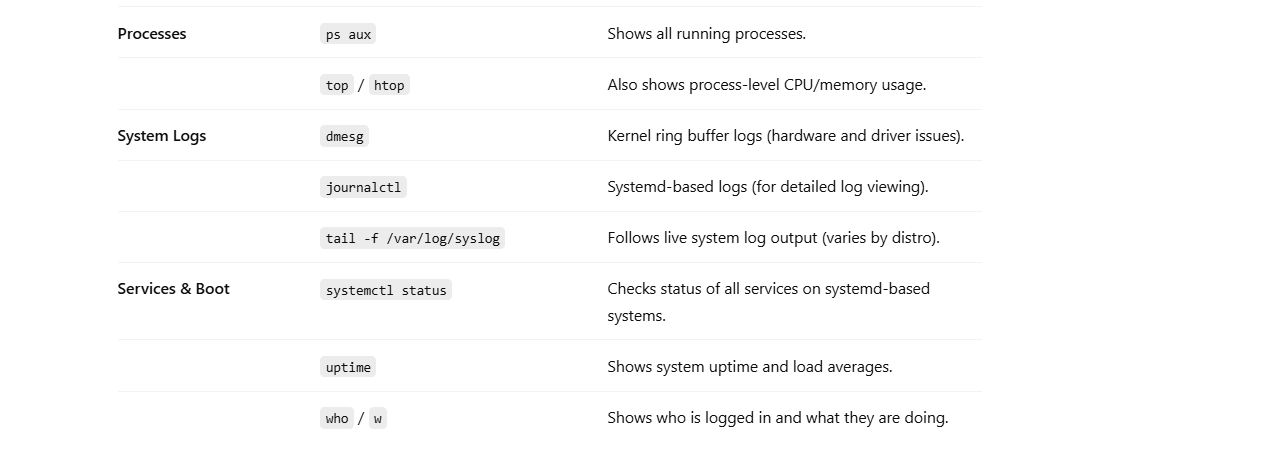
**🔹 NETWORK MONITORING**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| ip a | Show IP address | ip a |
| hostname -I | Get IP quickly | hostname -I |
| ss -tuln | Show open ports | ss -tuln |
| ping <host> | Check connectivity | ping 8.8.8.8 |
| traceroute <host> | Trace path to host | traceroute google.com |

**🔹 SYSTEM INFO & PROCESS**

| **Command** | **Purpose** | **Example** |
| --- | --- | --- |
| ps aux | List all running processes | ps aux |
| top / htop | Monitor processes live | top |
| uptime | Show system load and uptime | uptime |
| uname -a | Kernel and OS info | uname -a |
| dmesg | Kernel ring buffer logs | `dmesg |





\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

| **Area** | **Commands / Tools** | **Why It's Important** |
| --- | --- | --- |
| **System Performance** | sar, dstat, perf | For historical and advanced performance analysis. |
| **Storage & Filesystems** | lsblk, blkid, mount, umount, parted, fdisk, lvm | You'll likely be asked about partitions, LVM, and troubleshooting storage issues. |
| **Disk Health** | smartctl, badblocks | Useful for checking physical disk health — very relevant for datacenter roles. |
| **RAID Monitoring** | mdadm, cat /proc/mdstat | Know how to check and manage software RAID arrays. |
| **Backup & Restore** | rsync, tar, scp, dd, restore, dump | Often comes up in storage and sysadmin interviews. |
| **Security & Access** | chmod, chown, setfacl, getenforce, iptables | Permissions and firewalls are essential for server hardening. |
| **Automation & Logs** | cron, at, logrotate | Scheduled tasks and log management are common real-world responsibilities. |
| **Networking Deep Dive** | tcpdump, ethtool, nmcli, ip route, netplan | For diagnosing network issues beyond basics. |
| **Boot & Recovery** | grub, initramfs, rescue mode, systemctl rescue | Knowing recovery steps can really impress interviewers. |

w---jcpu---all process time

pcpu---particular process time

kill PID-----Gracefully asks the process to terminate. The process can clean up resources (files, memory) before exiting

Kill -9 PID-------Immediately terminates the process. It doesn't allow the process to clean up resources.\

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

Process—

A process is an instance of a running program, it consists of program code, data and various resources.

Socket—It is an end point for communication b/w 2m/c’s over a n/w

TCP===It abrivates Transmission Control protocol

It is secured connection oriented protocol

It establishes connection b/w sender & receiver before transmitting data.

It achieves reliability through error checking, ackg, & returns of lost transmission b/c of this increased lost transmission, overhead & slower performance

UDP--------user datagram protocol:

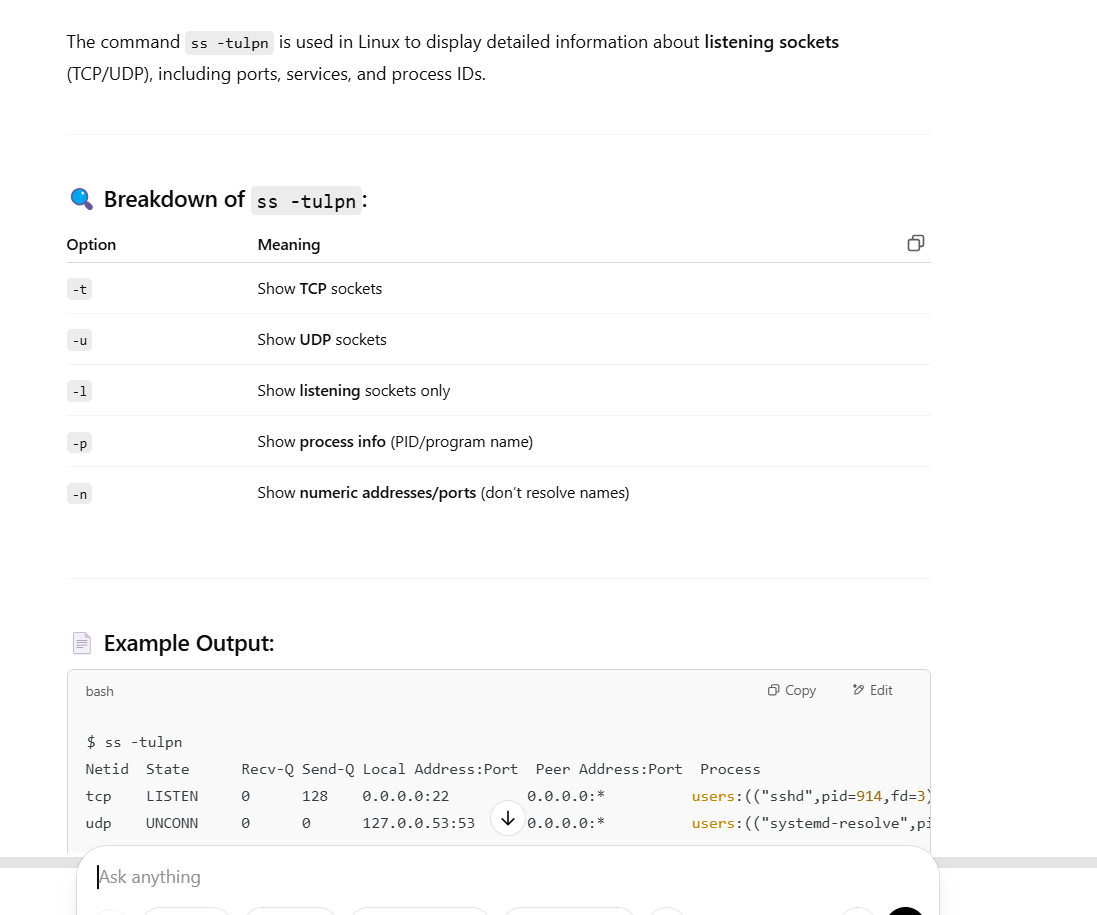
It is unsecured & connectionless protocol that sends data w.o establishing a connection.

It doesn’t guarantee the delivery or order of packets & performance minimal error checking resulting lower overhead & faster speeds.

The **OSI model** consists of **7 layers**, each with specific roles in network communication:

| **Layer #** | **Layer Name** | **Description** |
| --- | --- | --- |
| **7** | **Application Layer** | The layer closest to the user, responsible for end-user interactions (e.g., HTTP, FTP, DNS). |
| **6** | **Presentation Layer** | Translates data between the application and network. It handles data encoding, encryption, and compression. |
| **5** | **Session Layer** | Manages sessions or connections between applications (e.g., establishing, maintaining, and terminating connections). |
| **4** | **Transport Layer** | Provides reliable data transfer (e.g., TCP, UDP). Handles error correction and data flow control. |
| **3** | **Network Layer** | Routes data packets between devices on different networks (e.g., IP, routers). |
| **2** | **Data Link Layer** | Responsible for node-to-node data transfer and error detection. This includes MAC addresses and switches. |
| **1** | **Physical Layer** | The physical medium for transmitting data (e.g., cables, wireless signals, hardware interfaces). |

APST NDP



PATcHING PROCEDURE using Ansible

**Step-by-Step Patching with Ansible**

**Step 1: Set Up Ansible Environment**

* Make sure Ansible is installed on your control node:

bash

CopyEdit

ansible --version

* Install if not present:

bash

CopyEdit

sudo pip install ansible # or use package manager

**Step 2: Create an Inventory File**

Create an inventory file (e.g., hosts.ini) listing your target servers:

ini

CopyEdit

[linux\_servers]

server1.example.com

server2.example.com

[linux\_servers:vars]

ansible\_user=your\_user

ansible\_ssh\_private\_key\_file=~/.ssh/id\_rsa

**Step 3: Create a Patch Playbook**

Here’s a basic playbook (patching.yml) for patching **RHEL/CentOS** systems:

yaml

CopyEdit

---

- name: Patch Linux servers

hosts: linux\_servers

become: true

tasks:

- name: Update all packages to the latest version

yum:

name: '\*'

state: latest

when: ansible\_os\_family == "RedHat"

- name: Reboot if kernel was updated (RedHat)

reboot:

msg: "Rebooting after patching"

pre\_reboot\_delay: 60

post\_reboot\_delay: 60

reboot\_timeout: 600

when: ansible\_facts.packages['kernel'] is defined

For **Ubuntu/Debian**, modify the playbook like this:

yaml

CopyEdit

- name: Update all packages (Debian/Ubuntu)

apt:

upgrade: dist

update\_cache: yes

when: ansible\_os\_family == "Debian"

**Step 4: Run the Playbook**

Run the patching playbook:

bash

CopyEdit

ansible-playbook -i hosts.ini patching.yml

**Optional: Dry Run / Check Mode**

Use --check to simulate patching without making changes:

bash

CopyEdit

ansible-playbook -i hosts.ini patching.yml --check

**Step 5: Schedule Regular Patching (Optional)**

You can use **cron + Ansible Tower/AWX** or even a scheduled cron script like:

bash

CopyEdit

0 3 \* \* 0 /usr/bin/ansible-playbook -i /path/to/hosts.ini /path/to/patching.yml >> /var/log/ansible

Post patching checks:

**1. Verify Package Updates**

Check if all packages were updated successfully:

bash

CopyEdit

# On RHEL/CentOS

yum list updates # Should return nothing if fully patched

rpm -qa --last | head # Show most recently updated packages

# On Ubuntu/Debian

apt list --upgradable # Should return nothing

grep " upgrade " /var/log/dpkg.log | tail

**✅ 2. Verify Kernel Version**

If a kernel update was involved, check if the running kernel is the latest one:

bash

CopyEdit

uname -r # Current running kernel

rpm -q kernel # (RHEL) List installed kernels

If the running kernel does **not match** the latest installed one, a **reboot might be pending**.

**✅ 3. Check System Reboot Status**

Use Ansible or commands to check if a reboot is required:

bash

CopyEdit

# RHEL-based systems (no standard flag, but use uptime or Ansible reboot module in playbook)

uptime

# Debian/Ubuntu

[ -f /var/run/reboot-required ] && echo "Reboot required"

**✅ 4. Use Ansible to Check System Uptime or Last Boot**

bash

CopyEdit

ansible linux\_servers -m shell -a "uptime" -b

ansible linux\_servers -m shell -a "who -b" -b # Show last system boot time

**✅ 5. Log Verification**

Check system logs to verify updates:

bash

CopyEdit

# RHEL/CentOS

cat /var/log/yum.log | tail

# Ubuntu/Debian

cat /var/log/apt/history.log

**✅ 6. Custom Ansible Health Check (Post-patch)**

You can write a follow-up playbook (postcheck.yml) to verify the patch:

yaml

CopyEdit

- name: Verify system health after patching

hosts: linux\_servers

become: true

tasks:

- name: Check uptime

command: uptime

- name: Ensure no pending updates

yum:

list: updates

register: update\_check

when: ansible\_os\_family == "RedHat"

- name: Print if updates are pending

debug:

msg: "Pending updates: {{ update\_check.results }}"

when: update\_check.results | length > 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
If any server **fails to patch during an Ansible run**, you need to **troubleshoot step-by-step** to find out whether it's a **connectivity, authentication, Ansible, or package manager issue**.

**🛠️ Step-by-Step Troubleshooting When a Server Fails to Patch via Ansible**

**✅ 1. Check Ansible Error Output**

First, look at the error message in the Ansible output. Common categories:

* UNREACHABLE: SSH or host issue
* FAILED: Task failed due to command error
* PERMISSION DENIED: sudo/privilege issue
* PACKAGE MANAGER FAILURE: e.g. yum lock, repo error

**✅ 2. Test SSH Connectivity**

Make sure the control node can reach the server:

bash

CopyEdit

ssh user@hostname

If this fails:

* Check SSH keys
* Check firewall or security groups
* Ensure the SSH daemon is running

**✅ 3. Ping the Server with Ansible**

bash

CopyEdit

ansible linux\_servers -i hosts.ini -m ping

A successful response will return:

json

CopyEdit

"ping": "pong"

Otherwise, fix host or inventory settings.

**✅ 4. Check for Sudo Permissions**

Make sure the user has sudo rights (and without password if become: true is used):

bash

CopyEdit

ansible linux\_servers -m shell -a "whoami" -b

If this fails with "permission denied", fix the /etc/sudoers file or use:

ini

CopyEdit

ansible\_become\_password=<your\_password>

**✅ 5. Check Package Manager Lock or Failure**

**RHEL/CentOS:**

bash

CopyEdit

sudo yum list updates # Should not hang

sudo lsof /var/run/yum.pid # Check if another yum process is locking

**Ubuntu/Debian:**

bash

CopyEdit

sudo apt update

sudo fuser /var/lib/dpkg/lock

**Fix for stuck lock (if safe):**

bash

CopyEdit

sudo rm /var/run/yum.pid

sudo dpkg --configure -a

**✅ 6. Manually Run the Failing Ansible Task**

Run the specific task manually to replicate the failure:

bash

CopyEdit

ansible server1 -m yum -a "name='\*' state=latest" -b

You can also increase verbosity for more detail:

bash

CopyEdit

ansible-playbook patching.yml -i hosts.ini -vvv

**✅ 7. Check Disk Space & Network**

bash

CopyEdit

df -h # Make sure there is enough space

ping 8.8.8.8 # Check outbound internet

Package updates may fail if:

* Disk is full (especially /var)
* DNS is not working
* Yum/Apt repositories are unreachable

**✅ 8. Check Logs on the Target Server**

bash

CopyEdit

# RHEL/CentOS

cat /var/log/yum.log

journalctl -xe

# Ubuntu

cat /var/log/apt/history.log

**🧰 Pro Tip: Use Ansible Tags or Limit to Re-run on Failed Servers**

To re-run only on failed servers:

bash

CopyEdit

ansible-playbook patching.yml -i hosts.ini --limit server1

end to end patching report:

**full\_patch\_with\_alerts.yml**

yaml

CopyEdit

---

- name: Patch Linux servers with retry and alerts

hosts: linux\_servers

become: true

vars:

max\_retries: 2

slack\_webhook\_url: "https://hooks.slack.com/services/XXXXXXXXX/YYYYYYYYY/ZZZZZZZZZZZZZZ"

email\_recipient: "sysadmin@yourcompany.com"

smtp\_host: "smtp.yourcompany.com"

smtp\_port: 25

tasks:

- name: Ensure system is reachable

ping:

- name: Patch RHEL/CentOS systems

yum:

name: '\*'

state: latest

when: ansible\_os\_family == "RedHat"

register: patch\_result

retries: "{{ max\_retries }}"

delay: 15

until: patch\_result is succeeded

- name: Patch Debian/Ubuntu systems

apt:

upgrade: dist

update\_cache: yes

when: ansible\_os\_family == "Debian"

register: patch\_result

retries: "{{ max\_retries }}"

delay: 15

until: patch\_result is succeeded

- name: Reboot if kernel updated

reboot:

msg: "Rebooting after patching"

pre\_reboot\_delay: 60

post\_reboot\_delay: 60

reboot\_timeout: 600

when: "'kernel' in ansible\_facts.packages"

- name: Confirm patch success

debug:

msg: "✅ Patching succeeded on {{ inventory\_hostname }}"

when: patch\_result is succeeded

# --- Slack Alert on Failure ---

- name: Send Slack alert if patching failed

uri:

url: "{{ slack\_webhook\_url }}"

method: POST

headers:

Content-Type: "application/json"

body: |

{

"text": ":rotating\_light: \*Patching FAILED on {{ inventory\_hostname }}\*",

"username": "ansible-bot",

"icon\_emoji": ":exclamation:"

}

body\_format: json

when: patch\_result is failed

# --- Email Alert on Failure ---

- name: Send email alert if patching failed

mail:

host: "{{ smtp\_host }}"

port: "{{ smtp\_port }}"

to: "{{ email\_recipient }}"

subject: "❌ Ansible Patch FAILED on {{ inventory\_hostname }}"

body: |

Patching failed on {{ inventory\_hostname }} after {{ max\_retries }} attempts.

Please check logs or rerun manually.

when: patch\_result is failed

**✅ How to Run**

bash

CopyEdit

ansible-playbook -i hosts.ini full\_patch\_with\_alerts.yml

**🧪 Tips for Testing**

* To simulate failure, temporarily disable a host or give a fake hostname in the inventory.
* Monitor the Slack channel or check your email inbox for failure alerts.

\_\_\_\_\_\_  
kernel patching:

Performing **kernel patching** with Ansible is mostly the **same process** as regular patching—but with a few **additional steps and considerations**, because kernel updates **require a reboot** for changes to take effect.

**✅ Is Kernel Patching Different?**

* **Same mechanism**: You still use yum or apt to update the kernel package.
* **Different consideration**: You **must verify the kernel update** and **ensure reboot happens**.
* Optionally, **remove old kernels** after reboot (on RHEL-based systems).

**🛠️ Steps to Perform Kernel Patching via Ansible**

**Step 1: Patching the Kernel Package**

For **RHEL/CentOS**:

yaml

CopyEdit

- name: Ensure latest kernel is installed

yum:

name: kernel

state: latest

register: kernel\_update

become: true

For **Ubuntu/Debian**:

yaml

CopyEdit

- name: Install latest kernel (Ubuntu)

apt:

name: linux-image-generic

state: latest

update\_cache: yes

register: kernel\_update

become: true

**Step 2: Reboot If Kernel Was Updated**

yaml

CopyEdit

- name: Reboot if kernel was updated

reboot:

msg: "Rebooting to apply new kernel"

pre\_reboot\_delay: 60

post\_reboot\_delay: 60

reboot\_timeout: 600

when: kernel\_update.changed

**Step 3: Verify Running Kernel**

After reboot, you can check if the new kernel is active:

yaml

CopyEdit

- name: Display current running kernel

shell: uname -r

register: current\_kernel

- name: Display installed kernel versions

shell: rpm -q kernel # for RHEL

register: installed\_kernels

You can write a conditional to check if current\_kernel is the latest one.

**Step 4 (Optional): Remove Old Kernels**

Only after verifying the new kernel is stable!

yaml

CopyEdit

- name: Remove old kernels, keep latest 2

package:

name: "{{ item }}"

state: absent

with\_items: "{{ ansible\_facts.packages.kernel | map(attribute='name') | list }}"

when: item != ansible\_kernel

Or use the package-cleanup command (RHEL only):

yaml

CopyEdit

- name: Remove old kernels

command: package-cleanup --oldkernels --count=2 -y

when: ansible\_os\_family == "RedHat"

**✅ Final Note**

If you're in a **live production** environment:

* Schedule kernel patching during maintenance windows.
* Use --limit to apply it in batches (e.g., patch 2 servers at a time).
* Consider kernel live patching (like kpatch or Canonical Livepatch) if rebooting is not acceptable.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
how to connect to servers through cyber arc-

**Method 2: Using CyberArk Central Credential Provider (CCP) (for DevOps / Ansible)**

If you are integrating with automation (e.g., Ansible):

* Fetch credentials with a REST API:

bash

CopyEdit

curl -k -H "Authorization: Bearer <token>" "https://<ccp-url>/AIMWebService/api/Accounts?AppId=MyApp&Safe=LinuxSafe&Object=LinuxRoot@10.0.0.5"

* Use the retrieved password in automation (temporarily).

Requires configuration of **AppID**, permissions, and safe access.

**✅ Tips & Best Practices**

| **Action** | **Tool** |
| --- | --- |
| View session logs | PVWA > Monitoring |
| Manually checkout password | PVWA > Account > Show |
| Automatically inject creds | PSM (session proxy) |
| Rotate password | PVWA or via scheduled CPM |
| Integrate with scripts | Central Credential Provider |

**🔐 No Direct SSH/RDP?**

If CyberArk is enforced in your org, **you may not be allowed to connect directly using SSH with a password or key.** You must use the **PSM session** to avoid credential exposure.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-----

**1. What is the role of an SLA in managing complex technical environments?**

**Answer:**

* **SLA (Service Level Agreement)** is a contract between service providers and users defining the level of service expected. In the context of complex technical environments, SLAs set expectations for **uptime**, **incident resolution time**, **performance**, and **response times**.
* For instance, if an environment has an SLA requiring 99.9% uptime, it means that the system should not be down for more than ~8.76 hours a year.

**2. How do you manage incidents in Linux environments?**

**Answer:**

* Incident management involves identifying, responding to, and resolving issues that disrupt normal operations. The process typically follows these steps:
  1. **Detection**: Using monitoring tools like Nagios, Zabbix, or Prometheus to detect anomalies such as high CPU usage or disk space running out.
  2. **Prioritization**: Assessing the severity of the incident and categorizing it according to predefined SLAs.
  3. **Diagnosis**: Investigating logs (/var/log/syslog, /var/log/messages, journalctl).
  4. **Resolution**: Implementing a fix, such as restarting a service, applying patches, or increasing resource allocation.
  5. **Recovery**: Verifying the system is back to normal.
  6. **Documentation**: Updating incident reports and root cause analysis.

**3. What KPIs would you monitor in a complex technical environment?**

**Answer:**

* Key Performance Indicators (KPIs) are metrics used to evaluate the success of IT services:
  + **Uptime/Availability**: Percentage of time a system is operational.
  + **Incident Response Time**: Time taken to acknowledge and respond to an incident.
  + **Mean Time to Repair (MTTR)**: Average time taken to fix an issue.
  + **CPU/Memory Usage**: Helps in identifying resource bottlenecks.
  + **Disk Space Utilization**: Monitoring usage to prevent running out of space.
  + **Service Requests vs. Incidents**: To understand workload distribution.
  + **Automated Tasks Success Rate**: Measures the success of automated tasks like backups or patching.

**4. What are proactive actions you would take to optimize a Linux environment?**

**Answer:**

* **Resource Utilization**: Continuously monitor system resources (CPU, memory, disk, and network) and adjust them as needed. For example, configuring swap space for memory overflow or optimizing the number of processes using nice and renice commands.
* **Automation**: Automate repetitive tasks using **cron jobs** or **Ansible** to patch systems, back up data, or rotate logs.
* **Log Management**: Use centralized logging tools like **ELK stack** (Elasticsearch, Logstash, Kibana) to analyze logs efficiently and trigger alerts based on thresholds.
* **Capacity Planning**: Based on historical data, proactively scale up resources (e.g., adding storage to avoid disk space issues).

**5. How would you make recommendations for hardware or technology updates?**

**Answer:**

* **Monitor performance metrics** (CPU load, memory usage, disk I/O) to identify bottlenecks.
* Analyze current hardware limitations (e.g., old hard drives, insufficient RAM) and suggest **upgrades** or **hardware replacements**.
* Recommend updates on **software versions** that provide new features or security patches, such as kernel updates or application updates.
* **Virtualization**: Recommend switching to virtualized environments (e.g., using **VMware** or **KVM**) for better resource management and scaling.
* Stay updated on new technologies and hardware that offer **better performance**, **energy efficiency**, and **cost savings**.

**6. How would you contribute to an implementation project from planning to operational support?**

**Answer:**

* **Planning Phase**: Participate in requirements gathering, defining the architecture, and understanding project goals. Consider **scalability**, **security**, and **reliability**.
* **Implementation Phase**: Help in configuring and installing systems. For example, setting up servers, configuring networking, and installing necessary applications. Tools like **Terraform** or **Ansible** can automate many aspects.
* **Testing Phase**: Ensure that systems meet performance benchmarks and security requirements.
* **Operational Support**: After deployment, provide operational support such as **monitoring**, **incident management**, and **optimization**. Ensure documentation is up-to-date for system maintenance.

**7. How do you organize and participate in technical committees?**

**Answer:**

* **Preparation**: Review documentation, understand the agenda, and be ready to discuss potential improvements or concerns.
* **Participation**: Share insights on system performance, recommend solutions based on **root cause analysis**, and discuss new technologies or tools that can improve efficiency.
* **Follow-up**: Document key takeaways and action items from the committee meeting. Follow through with tasks assigned and ensure that decisions made are implemented.

**8. What is the importance of keeping technical documentation up to date?**

**Answer:**

* **Clarity**: Ensures that new team members or contractors can quickly understand system configurations, processes, and workflows.
* **Troubleshooting**: Helps to quickly locate common problems and solutions, such as how to recover from a specific error.
* **Compliance**: For audits and regulatory compliance, having accurate documentation ensures that all systems and processes are properly accounted for.
* **Efficiency**: Well-maintained documentation helps reduce downtime and speeds up incident response times.

**9. What would you include in training systems administrators?**

**Answer:**

* **Basic Linux Administration**: File systems, process management, networking, and user management.
* **Advanced Troubleshooting**: Debugging logs, using strace, gdb, systemctl, etc.
* **Automation Tools**: Teaching tools like **Ansible**, **Terraform**, and **Puppet** to automate deployments and configurations.
* **Security Best Practices**: Hardening Linux servers, firewalls, SELinux/AppArmor, and managing permissions and access control.
* **Performance Monitoring**: Using tools like top, htop, vmstat, iostat, and monitoring systems like **Nagios** or **Zabbix**.

**10. How would you ensure compliance and security during technical platform optimizations?**

**Answer:**

* **Follow Security Best Practices**: Regularly patch systems, disable unnecessary services, and use encryption for sensitive data.
* **Auditing**: Use tools like **auditd** or **OSSEC** to monitor changes to critical files or configurations.
* **Backup**: Ensure that systems are properly backed up before making any changes, and have a recovery plan in place.
* **Testing**: Perform non-disruptive tests and review security logs before implementing changes.

**Conclusion**

These questions focus on your ability to **manage, optimize, and secure** complex environments while meeting SLAs, ensuring system availability, and responding to incidents effectively. The answers emphasize your **hands-on experience** with monitoring, troubleshooting, automation, and providing technical leadership.

Let me know if you'd like more details on any specific question!

4o mini

**You said:**

give me for this jd Experience on RedHat 5,6 &7 system administration, installation and configuration. (Must) ð Linux administration and trouble shooting. ð Knowledge on Kernel Patching, OS upgrade, hardware drivers and firmware upgrades. ð Working knowledge of HA / Clusters. ð Experience on HP / CISCO Hardware ð Shell scripting ð Knowledge on Volume Manager, native multipathing, FC connectivity. ð Should have strong fundamental understanding of VMware infrastructure. ð Kickstart installation/Trouble shooting. ð Knowledge on Satellite , Puppet, Ansible , etc. will be an add-on. ð Network and network services (NFS, NIS, etc.) administration and troubleshooting. ð Strong concepts of networking like TCP / IP . ð Red hat Certification ð SAN (EMC/HP/IBM) ð Automation expert. ð Linux Certification ð RedHat Cluster

**ChatGPT said:**

Updated saved memory

Here are **Linux-related questions and answers** relevant to your **JD (Job Description)**, with a focus on RedHat systems, HA clusters, virtualization, and automation tools.

**1. What are the key differences between Red Hat 5, 6, and 7 in terms of system administration?**

**Answer:**

* **Red Hat 5**:
  + Older version with traditional init system.
  + **Ext3 filesystem** was the default.
  + RPM package manager for updates and installations.
  + Minimal support for modern hardware.
* **Red Hat 6**:
  + Introduced **systemd** as an option for service management (but still uses init by default).
  + **Ext4** filesystem became the default.
  + Enhanced support for virtual environments (e.g., **KVM**).
  + **NetworkManager** became part of the default installation for easier network configuration.
* **Red Hat 7**:
  + Fully switched to **systemd** as the default init system.
  + **XFS** is the default filesystem.
  + Improved **security** features with **SELinux** and **firewallD**.
  + Introduced support for **Docker** containers and better integration with **virtualization** platforms like **VMware**.

**2. What steps would you follow for kernel patching and OS upgrades on RedHat systems?**

**Answer:**

1. **Kernel Patching**:
   * **Check current kernel version**: uname -r
   * **Update package repositories**: yum update
   * **Install the latest kernel**: yum install kernel
   * **Reboot the system** to apply the new kernel: reboot
   * **Verify the updated kernel**: uname -r
2. **OS Upgrade**:
   * **Backup data** before performing an upgrade.
   * **Update all packages**: yum update
   * Use **Red Hat Subscription Manager** to ensure your system is subscribed.
   * **Upgrade OS** using yum or **Red Hat Satellite** for version upgrades (e.g., from RHEL 6 to 7).
   * **Post-upgrade checks**: Check logs, verify services, and test applications.

**3. How do you manage and troubleshoot a high availability (HA) cluster in RedHat?**

**Answer:**

* **Configure HA Clusters** using **Red Hat Cluster Suite (RHCS)** or **Pacemaker**.
  + **Install required packages**: yum install pcs pacemaker corosync
  + **Configure the cluster**:
    - Set up the **corosync.conf** and **pcs cluster**.
    - Use **pcs** to create and manage resources (e.g., virtual IP, services).
  + **Test failover** by simulating failures on one node and ensuring services fail over to another node.
  + **Troubleshoot** by checking logs (/var/log/messages), cluster status (pcs status), and node health (pcs node status).

**4. How would you handle kernel module updates or firmware upgrades on RedHat systems?**

**Answer:**

* **Kernel Module Updates**:
  + Ensure the module is compatible with the current kernel.
  + Use modprobe <module\_name> to load or update a kernel module.
  + To remove a module: modprobe -r <module\_name>
  + Verify module status: lsmod | grep <module\_name>
* **Firmware Upgrades**:
  + Use **fwupd** or vendor-specific tools to update firmware.
  + For **HP** servers, use tools like **HPONCFG** or **Smart Array Configuration**.
  + For **Cisco** hardware, use their tools or **Cisco UCS Manager**.

**5. What is the role of Volume Manager and multipathing in RedHat?**

**Answer:**

* **Volume Manager (LVM)**:
  + LVM allows logical volume management and provides flexibility in managing storage.
  + You can create volume groups, logical volumes, and resize volumes without disrupting services.
  + Commands:
    - pvcreate, vgcreate, lvcreate to create volumes.
    - lvextend, lvreduce to resize volumes.
* **Multipathing**:
  + **Native Multipathing (DM-Multipath)** allows the system to use multiple paths for redundancy and load balancing in SAN (Storage Area Network) environments.
  + Configuration involves editing /etc/multipath.conf and enabling the multipath service: systemctl enable multipathd.
  + Command to check paths: multipath -ll.

**6. How would you configure and troubleshoot networking services like NFS and NIS?**

**Answer:**

* **NFS (Network File System)**:
  1. **Install NFS**:

bash

CopyEdit

yum install nfs-utils

* 1. **Configure NFS server** by editing /etc/exports:

bash

CopyEdit

/data \*(rw,sync,no\_root\_squash)

* 1. **Start NFS service**:

bash

CopyEdit

systemctl start nfs-server

systemctl enable nfs-server

* 1. **Troubleshooting**:
     + Check rpcbind and nfs-server logs: journalctl -u rpcbind -u nfs-server.
     + Use showmount -e <server> to verify shared directories.
* **NIS (Network Information Service)**:
  1. **Install NIS**:

bash

CopyEdit

yum install ypserv

* 1. **Configure NIS** on the server by editing /etc/yp.conf.
  2. **Start NIS service**:

bash

CopyEdit

systemctl start ypserv

systemctl enable ypserv

* 1. **Troubleshooting**:
     + Use ypwhich to check NIS server status.
     + Check logs for errors: journalctl -u ypserv.

**7. How do you use Puppet, Ansible, and Satellite for automation in a RedHat environment?**

**Answer:**

* **Puppet**:
  + **Install Puppet Agent** on client machines.
  + Use **Puppet master** to define desired system states in manifests (e.g., ensuring specific packages are installed or services are running).
  + Automate software installation, configuration, and updates.
* **Ansible**:
  + **Install Ansible**: yum install ansible.
  + Use **Playbooks** to automate tasks like configuring users, installing packages, and managing services.
  + Ansible is agentless, so it works over SSH, simplifying configuration and automation.
* **Red Hat Satellite**:
  + **Satellite** is used for **patch management**, **subscription management**, and **configuration management**.
  + It automates updates, provisioning, and configuration management for large environments.

**8. What is Kickstart installation and how do you troubleshoot it?**

**Answer:**

* **Kickstart** allows for automated, unattended installations of RedHat systems.
  + **Create Kickstart file**: Define system configurations, partitions, package selections, network settings, etc.
  + Run the installer with the Kickstart file:

bash

CopyEdit

anaconda --kickstart=/path/to/ks.cfg

* **Troubleshooting**:
  + Check logs in /var/log/anaconda for errors during installation.
  + Ensure that the Kickstart file is correctly formatted, particularly regarding network and storage configuration.

**9. How do you manage VMware infrastructure in a RedHat environment?**

**Answer:**

* **VMware Tools**: Install **VMware Tools** to improve guest OS performance and enable features like seamless mouse integration, time synchronization, and enhanced networking.
* **VMware ESXi**: You can manage RedHat VMs via **vSphere Client** or use **VMware CLI** tools (esxcli, vmkfstools) for advanced management.
* **Automation**: Use **Ansible** or **vSphere Automation SDK** to automate provisioning, configuration, and management tasks for RedHat VMs.

**10. What is Red Hat Cluster and how is it configured?**

**Answer:**

* **Red Hat Cluster** provides high availability for applications by ensuring that if one node fails, the other takes over seamlessly.
  + **Install Pacemaker** and **Corosync** for clustering.
  + Configure **cluster nodes** and **resources** using pcs command:

bash

CopyEdit

pcs cluster auth <node1> <node2>

pcs cluster setup --name <cluster\_name> <node1> <node2>

pcs cluster start --all

pcs resource create <resource\_name> <resource\_type> op start interval=0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
kIckstart:

**Basic to Intermediate Kickstart Interview Questions**

**1. What is Kickstart in Linux?**

**Answer:**  
Kickstart is a method to automate the installation of Red Hat-based Linux distributions like RHEL and CentOS. It uses a configuration file (ks.cfg) that provides all the required inputs (like disk partitioning, package selection, network settings) to perform an unattended installation.

**2. Where is the Kickstart file usually located during a PXE boot installation?**

**Answer:**  
The Kickstart file can be located on a web server (HTTP/HTTPS), NFS, or FTP server. The path is specified in the PXE boot configuration using:

nginx

CopyEdit

linux ks=http://<server>/ks.cfg

**3. What are the main sections in a Kickstart file?**

**Answer:**

* lang – Language selection
* keyboard – Keyboard layout
* timezone – Timezone setting
* network – Network configuration
* rootpw – Root password
* bootloader – Bootloader options
* clearpart, part, volgroup, logvol – Disk partitioning
* %packages – Packages to install
* %post – Post-installation scripts

**4. How do you test a Kickstart configuration file before using it in production?**

**Answer:**

* Use ksvalidator /path/to/ks.cfg to validate syntax.
* Perform a test installation in a **virtual environment** (like VirtualBox or KVM) with the Kickstart file.
* Use logging (%post --log=/root/postinstall.log) to verify %post script execution.

**5. What is the use of %post section in Kickstart?**

**Answer:**  
The %post section contains shell commands that are executed **after** the OS installation is completed but before the first reboot. It’s commonly used to:

* Install custom packages
* Configure services
* Download scripts
* Set file permissions

**6. What’s the difference between part and logvol in Kickstart?**

**Answer:**

* part: Creates a **standard partition** on the disk.
* logvol: Creates a **logical volume** using LVM inside a volume group.

**7. Can you create a user during installation with Kickstart?**

**Answer:**  
Yes, you can use the user directive in the Kickstart file:

bash

CopyEdit

user --name=john --password=encrypted\_password --groups=wheel --shell=/bin/bash

**8. How do you automate Kickstart installations over the network?**

**Answer:**

1. Set up a **PXE boot** server with TFTP and DHCP.
2. Place the Kickstart file on a web/NFS/FTP server.
3. Add the ks= parameter in the bootloader entry:

go

CopyEdit

append initrd=initrd.img ks=http://<server>/ks.cfg

**9. What should you do if Kickstart installation fails?**

**Answer:**

* Switch to a different terminal (e.g., Ctrl+Alt+F3 or F4) and check logs:
  + /tmp/anaconda.log
  + /tmp/syslog
  + /tmp/ks-script-<n>.log
* Validate Kickstart syntax with ksvalidator.
* Check that the Kickstart file path is reachable and permissions are correct.

**10. Can you include external scripts in a Kickstart file?**

**Answer:**  
Yes, in the %post section you can use wget, curl, or scp to fetch and execute external scripts:

bash

CopyEdit

%post

curl -O http://server/script.sh

sh script.sh

%end

Would you like me to provide a sample ks.cfg file for practice or more advanced questions (e.g., for integrating with Ansible or Satellite)?

4o

**You said:**

what is anacoda, isit related to kickstart

**ChatGPT said:**

Yes, **Anaconda** is **directly related to Kickstart**.

**🐍 What is Anaconda (in Linux context)?**

**Anaconda** is the **installer** used by Red Hat-based Linux distributions (RHEL, CentOS, Fedora, Rocky Linux, etc.) to perform operating system installations.

**🧩 Key Facts About Anaconda:**

| **Feature** | **Description** |
| --- | --- |
| **Role** | It's the **installation program** that handles partitioning, package installation, and system configuration. |
| **Text & GUI** | Works in **graphical** and **text** modes. |
| **Kickstart usage** | When a Kickstart file is provided, **Anaconda reads and executes** it to automate the installation. |
| **Logs location** | During installation, Anaconda logs to /tmp/anaconda.log, /tmp/syslog, etc. |

**🔗 How Anaconda and Kickstart Are Related:**

| **Aspect** | **Kickstart** | **Anaconda** |
| --- | --- | --- |
| Type | Configuration File | Installer Software |
| Purpose | Defines how the OS should be installed | Executes the OS installation |
| Relationship | Kickstart is **input** to Anaconda | Anaconda **reads** Kickstart to perform actions |

✅ Think of **Kickstart as the recipe**, and **Anaconda as the chef** that follows it.

**Example:**

If you boot a system with:

nginx

CopyEdit

linux ks=http://server/ks.cfg

* **Anaconda** starts automatically,
* It reads the ks.cfg file,
* Performs everything (disk partitioning, network setup, package installation) **without manual input**.

**Scenario: PXE Boot + Kickstart + Anaconda Automated RHEL Installation**

**🧱 1. Infrastructure Setup**

You need:

* A **PXE server** (with DHCP, TFTP)
* A **web server** (HTTP or FTP or NFS) to host the Kickstart file
* RHEL ISO files extracted to a shared location

**📁 2. Prepare Kickstart File (ks.cfg)**

Save this file on your HTTP server, e.g., /var/www/html/ks.cfg:

bash

CopyEdit

# ks.cfg

lang en\_US.UTF-8

keyboard us

timezone Asia/Kolkata

rootpw --iscrypted $6$abcdef... # Encrypted password

bootloader --location=mbr

clearpart --all --initlabel

autopart

network --bootproto=dhcp --device=eth0 --onboot=on

firewall --disabled

selinux --disabled

%packages

@core

wget

curl

%end

%post

echo "Post-install script running..." >> /root/postinstall.log

%end

**⚙️ 3. PXE Boot Menu Configuration**

Edit your PXE boot config file, usually at /var/lib/tftpboot/pxelinux.cfg/default or similar:

bash

CopyEdit

label linux

menu label ^Install RHEL 8 (Automated)

kernel vmlinuz

append initrd=initrd.img inst.ks=http://192.168.0.10/ks.cfg

Here:

* inst.ks=http://192.168.0.10/ks.cfg is the **Kickstart file**
* vmlinuz and initrd.img are from the RHEL ISO /images/pxeboot/

**🐍 4. Installation Flow**

1. Client boots via PXE.
2. PXE server loads vmlinuz and initrd.img.
3. **Anaconda** starts and reads inst.ks=... (your Kickstart file).
4. Anaconda performs installation using the instructions in ks.cfg.
5. %post scripts are executed.
6. The system reboots fully installed and configured.

**📝 Logs You Can Check**

After installation:

* /root/anaconda-ks.cfg: Kickstart file used (auto-generated if interactive)
* /var/log/anaconda.log: Full Anaconda installation log
* /root/postinstall.log: Your custom post script output

This setup is ideal for **mass deployments**, **consistent server builds**, or **cloning environments**.

Would you like a downloadable sample ks.cfg file or a visual architecture diagram of this setup?

1. To find the script subhadra.sh if you forgot where it's located, you can use the find, locate, or which command depending on how and where it might have been saved.

**✅ 1. Using find Command (Best for precision)**

Search from root (this may take time):

bash

CopyEdit

sudo find / -type f -name "subhadra.sh" 2>/dev/null

* / — root directory (searches entire system)
* -type f — only files
* -name — match exact name
* 2>/dev/null — suppress permission errors

If you want to search only in your home:

bash

CopyEdit

find ~ -type f -name "subhadra.sh"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
2. In a **Linux environment**, **NAS volume persistent volumes** refer to **network-attached storage (NAS)** that is mounted on a Linux server and made **persistently available** across reboots or container restarts (e.g., in Kubernetes, Docker, or virtual environments).

**🔍 What Is NAS?**

**NAS (Network Attached Storage)** is:

* A **file-level storage device** connected over a **network**.
* Typically accessed via **NFS (Network File System)** or **SMB/CIFS**.
* Used for **shared access** to files among multiple servers or clients.

**📌 What Are Persistent Volumes?**

* A **persistent volume** is a storage location that **retains data** even when the application, service, or container using it is stopped or restarted.
* In **Linux**, it usually means mounting a NAS share to a directory and ensuring it is **automatically mounted** on boot.

**🛠 Example: Mounting an NFS (NAS) Volume as a Persistent Volume in Linux**

**Step 1: Install NFS utilities (if not installed)**

bash

CopyEdit

sudo apt install nfs-common # Debian/Ubuntu

sudo yum install nfs-utils # RHEL/CentOS

**Step 2: Create a Mount Point**

bash

CopyEdit

sudo mkdir -p /mnt/nasdata

**Step 3: Mount Temporarily**

bash

CopyEdit

sudo mount -t nfs 192.168.1.100:/export/nasdata /mnt/nasdata

**Step 4: Make It Persistent (Survives Reboot)**

Edit /etc/fstab:

bash

CopyEdit

192.168.1.100:/export/nasdata /mnt/nasdata nfs defaults 0 0

Now run:

bash

CopyEdit

sudo mount -a

**📦 In Kubernetes (Optional Advanced Use)**

In Kubernetes, NAS-backed **Persistent Volumes (PVs)** are used to provide external storage:

yaml

CopyEdit

apiVersion: v1

kind: PersistentVolume

metadata:

name: nfs-pv

spec:

capacity:

storage: 5Gi

accessModes:

- ReadWriteMany

nfs:

server: 192.168.1.100

path: "/export/nasdata"

**✅ Benefits**

* **Centralized storage** accessible from multiple nodes.
* **Data persistence** even after service/app restarts.
* **Easier backup and replication**.
* **Ideal for shared resources**, e.g., logs, configs, or databases.

**Summary**

In Linux, a **NAS volume persistent volume** means mounting a **network-shared directory (via NFS or SMB)** and configuring it to **persist across reboots**. It’s commonly used in both traditional server setups and container environments like Docker or Kubernetes.

3.d/f b/w find & locate

 **find** searches the file system in real-time, offering more flexibility but can be slower.

 **locate** uses a pre-built database for faster searches but may show outdated results if the database isn't updated.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

4. how to check cpu utilisation process that are running with your userid

**ChatGPT said:**

To check CPU utilization for processes running under your user ID, you can use the top or ps commands:

1. **Using top**:
   * Run top and press u to filter by your username.

bash

CopyEdit

top

Then press u and enter your username.

1. **Using ps**:
   * You can use the ps command with -u to filter processes by your user ID.

bash

CopyEdit

ps -u <your-username> -o pid,%cpu,command

This shows the PID, CPU usage, and the command for processes running under your user ID.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_----

5.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**1. Server is running slow — how do you troubleshoot?**

**Answer:**

First, I check CPU, memory, and load average using top, htop, or uptime. If load is high, I identify top-consuming processes. Then I check memory with free -h and swap usage. I also look at disk I/O using iostat or vmstat, and ensure there’s no high wait time (wa%). For disk space, I run df -h and du -sh /\*. If the system has recently slowed down, I check logs using journalctl -xe or dmesg for errors like hardware or OOM kills.

**💽 2. A disk is full — what steps do you take?**

**Answer:**

I use df -h to identify which partition is full. Then I run du -sh \* inside the large directory to find what’s consuming space. Often, it’s log files under /var/log or app cache files. I clean up old logs or rotate them using logrotate. If necessary, I check for hidden files (like deleted files still held by processes) using lsof | grep deleted. If logs are rapidly filling space, I investigate the service causing them.

**🧠 3. Swap usage is high — what could be wrong and how do you fix it?**

**Answer:**

High swap usage usually means the system is running low on RAM. I check with free -h and top to see which processes use most memory. If it's expected, I may increase RAM or tune the application. If not, I look for memory leaks or misconfigured services. I may restart memory-heavy services temporarily. I also check the swappiness value (cat /proc/sys/vm/swappiness) — reducing it can delay swap usage.

**🔌 4. Network is down or slow — what do you check?**

**Answer:**

First, I check interface status using ip a or ifconfig. Then, I ping the default gateway to verify local connectivity. If it's a DNS issue, I try nslookup or dig. I check routes using ip route and firewall rules using iptables -L or firewalld. For performance issues, I use iftop, nload, or netstat -s to analyze traffic and errors. Logs (/var/log/messages, dmesg) may also show NIC errors.

**🔐 5. A user cannot SSH into the server — how do you fix it?**

**Answer:**

I verify SSH is running with systemctl status sshd. I check /var/log/secure or journalctl -u sshd for authentication errors. I confirm the user exists and has correct permissions on their home and .ssh directories (chmod 700 ~/.ssh, chmod 600 ~/.ssh/authorized\_keys). I also check if the firewall allows SSH (port 22) and verify no security policies like SELinux are blocking access.

**🧱 6. How do you check RAID health?**

**Answer:**

For software RAID, I use cat /proc/mdstat or mdadm --detail /dev/mdX. It shows sync status, failed disks, etc. For hardware RAID, I use vendor tools like MegaCli, hpacucli, or storcli. I also monitor SMART data using smartctl -a /dev/sdX.

How many groups user exits

groups <username>

or

groups <username> | awk '{print NF - 1}'