Linux Fundamentals

Core Commands 🛠️

- ls: Lists directory contents.
  - ls -l: Long format (permissions, owner, size, date).
  - ls -a: Shows hidden files (starting with '.').
- grep: Searches for patterns in files.
  - grep "pattern" filename: Finds "pattern" in the specified file.
  - grep -r "pattern" directory/: Searches recursively.
- find: Searches for files in a directory hierarchy.
  - find . -name "*.txt": Finds all .txt files in the current directory and subdirectories.
  - find /home -user johndoe: Finds files owned by user "johndoe" in /home.
- cp: Copies files and directories.
  - cp source_file destination_file
  - cp -r source_directory destination_directory: Copies directories recursively.
- scp: Securely copies files between hosts.
  - scp user@remote_host:/path/to/remote_file /path/to/local_destination (remote to local)
  - scp /path/to/local_file user@remote_host:/path/to/remote_destination (local to remote)
- ssh: Securely connects to a remote host.
  - ssh user@remote_host
- rsync: Efficiently synchronizes files/directories, locally or remotely.
  - rsync -avz /source_directory/ user@remote_host:/destination_directory/ (local to remote, archive, verbose, compress)
- ps: Displays current processes.
  - ps aux: Shows all processes for all users in a detailed format.

    - ps -ef: Another way to show all processes.

- kill: Sends signals to processes (commonly to terminate them).
    - kill PID: Sends TERM signal (graceful shutdown).
    - kill -9 PID or kill -SIGKILL PID: Sends KILL signal (forceful shutdown).

Linux Runlevels (System V init) / Targets (systemd)

- **Runlevels** (older init systems like SysVinit) define the state of the machine and what services are running.
    - **0**: Halt
    - **1**: Single-user mode (maintenance)
    - **3**: Multi-user mode with networking (text-based)
    - **5**: Multi-user mode with networking and GUI
    - **6**: Reboot
- **systemd targets** have largely replaced runlevels. Common targets include:
    - poweroff.target (0)
    - rescue.target (1)
    - multi-user.target (2, 3, 4)
    - graphical.target (5)
    - reboot.target (6)
    - systemctl get-default: Shows the default target.
    - systemctl set-default multi-user.target: Sets the default target.

Users and Group Permissions 👥

- **Users**: Individuals who can log in and use the system.
- **Groups**: Collections of users. Permissions can be assigned to groups.
- **Permissions**: Read (r), Write (w), Execute (x).
    - Displayed as rwxrwxrwx (User, Group, Others).

  - Example: -rw-r--r-- means the owner has read/write, group has read, others have read.
  - chmod: Changes permissions (e.g., chmod u+x file adds execute for user).
  - chown: Changes ownership (e.g., chown user:group file).

6 Stages of Linux Boot Process 🚀

1. **BIOS/UEFI**: Initializes hardware, performs POST (Power-On Self-Test), and loads the bootloader.
2. **Bootloader (GRUB/LILO)**: Loads the kernel into memory and (optionally) an initramfs. Presents a boot menu.
3. **Kernel Initialization**: The kernel sets up memory, loads drivers, and mounts the root filesystem (often initially via initramfs).
4. **Init Process (SysVinit/systemd)**: The kernel starts the first user-space process, init (PID 1).
   - **SysVinit**: Executes startup scripts based on runlevels (/etc/inittab and scripts in /etc/rc.d/).
   - **systemd**: Manages services and targets using "units" defined in service files.
5. **Runlevel Scripts / systemd Targets**: Services are started according to the defined runlevel or target.
6. **Login Prompt/Display Manager**: The system is ready for user login, either via a text-based console or a graphical display manager.

Linux Signals 🚦

- Signals are asynchronous notifications sent to processes to inform them of events.
- Common signals:
  - SIGHUP (1): Hangup. Often used to tell a daemon to reload its configuration.
  - SIGINT (2): Interrupt (Ctrl+C). Requests termination.
  - SIGQUIT (3): Quit (Ctrl+\). Requests termination and core dump.
  - SIGKILL (9): Kill. Unconditional termination (cannot be caught or ignored).
  - SIGTERM (15): Terminate. Requests termination (can be caught and handled gracefully). Default for kill command.

  - SIGSTOP (19): Stop. Pauses the process (cannot be caught or ignored).
  - SIGTSTP (20): Terminal Stop (Ctrl+Z). Pauses the process, can be resumed.
- Use man 7 signal for a full list and details.

Web Server & Networking

Nginx Server Basics 🌐

- High-performance, event-driven web server, reverse proxy, load balancer, and HTTP cache.
- **Key features**: Scalability, efficiency, rich feature set.
- **Configuration**: Primarily through nginx.conf and included files.
  - **Directives**: Instructions (e.g., listen, server_name, location).
  - **Contexts/Blocks**: Sections like http, server, location that group directives.
- **Basic commands**:
  - sudo systemctl start nginx
  - sudo systemctl stop nginx
  - sudo systemctl restart nginx
  - sudo systemctl reload nginx (reloads config without dropping connections)
  - sudo nginx -t (tests configuration)

Named-Based vs. IP-Based Virtual Hosts

- **Virtual Hosts**: Allow one server to host multiple websites.
- **IP-Based**: Each website has its own unique IP address. Simpler to configure but requires more IP addresses.
- **Name-Based**: Multiple websites share a single IP address. The server determines which site to show based on the Host header sent by the client's browser. More common and efficient with IP address usage.

Proxy vs. Reverse Proxy

- **Proxy (Forward Proxy)**:

    - Acts on behalf of **clients**.
    - Clients send requests to the proxy, which then forwards them to the internet/destination server.
    - **Use cases**: Bypass filtering, caching, anonymity, access control for client outbound traffic.
    - **How it works**: Client ➡️ Proxy ➡️ Internet Server.
- **Reverse Proxy**:
    - Acts on behalf of **servers**.
    - Clients send requests to the reverse proxy (thinking it's the actual server). The reverse proxy then forwards the request to one or more backend servers.
    - **Use cases**: Load balancing, SSL termination, caching, security (hiding backend server IPs), serving static content.
    - **How it works**: Client ➡️ Reverse Proxy ➡️ Backend Server(s).

Basic Networking

- **OSI Model (Open Systems Interconnection)**: A 7-layer conceptual framework for network communication.
    1. **Physical**: Bits, cables, hardware.
    2. **Data Link**: MAC addresses, Ethernet, switching.
    3. **Network**: IP addresses, routing, ICMP.
    4. **Transport**: TCP (reliable, connection-oriented), UDP (unreliable, connectionless), ports.
    5. **Session**: Manages connections, session establishment.
    6. **Presentation**: Data formatting, encryption, compression.
    7. **Application**: HTTP, FTP, SMTP, DNS - user-facing protocols.
- **HTTP (HyperText Transfer Protocol)**:
    - Application layer protocol for transmitting hypermedia documents (e.g., HTML).

    - **How it works**: Client sends an HTTP request (e.g., GET, POST) to a server. Server processes it and sends back an HTTP response (e.g., status code 200 OK, content).
    - **Plain text**: Data is not encrypted.
- **HTTPS (HyperText Transfer Protocol Secure)**:
    - HTTP over SSL/TLS (Secure Sockets Layer/Transport Layer Security).
    - **How it works**: Same as HTTP, but communication is encrypted.
        1. **Handshake**: Client and server establish a secure connection using SSL/TLS certificates to verify identity and agree on encryption keys.
        2. **Encrypted Data Transfer**: Subsequent HTTP requests and responses are encrypted.
    - **Difference**: **Security**. HTTPS encrypts data, protecting it from eavesdropping and tampering. HTTP does not. HTTPS uses port 443; HTTP uses port 80.

Linux Ports 🔢

- Logical connection points for network communication.
- **Well-known ports (0-1023)**: Reserved for common services (e.g., 22 for SSH, 80 for HTTP, 443 for HTTPS). Require root privileges to bind to.
- **Registered ports (1024-49151)**: Can be registered by software vendors.
- **Dynamic/Private ports (49152-65535)**: For temporary or private connections.
- Key command: netstat -tulnp or ss -tulnp (shows listening TCP/UDP ports and associated programs).

Working with Linux Systems

Working Remotely 💻

- **SSH (Secure Shell)**: Primary tool for secure remote login and command execution.
    - ssh username@hostname_or_ip
- **SCP (Secure Copy)**: For secure file transfer.
    - scp source_file user@remote_host:/destination_path

- **SFTP (SSH File Transfer Protocol)**: Provides file system access over SSH, more interactive than scp.
- **Rsync**: For efficient file synchronization over SSH.
  - rsync -avz local_dir/ user@remote_host:/remote_dir/
- **Terminal Multiplexers (Screen, Tmux)**: Allow detaching and reattaching sessions, keeping processes running even if the connection drops.

FHS - Linux File System Hierarchy 📂

- A standardized directory structure for Linux.
- Key directories:
  - /: Root directory.
  - /bin: Essential user command binaries (for all users).
  - /sbin: Essential system binaries (for root).
  - /etc: Configuration files.
  - /dev: Device files.
  - /proc: Virtual filesystem providing process and kernel information.
  - /var: Variable files (logs, spool files, etc.).
  - /tmp: Temporary files.
  - /usr: User utilities and applications.
    - /usr/bin: Non-essential command binaries.
    - /usr/sbin: Non-essential system binaries.
    - /usr/local: Locally installed software.
  - /home: User home directories.
  - /boot: Boot loader files, kernel.
  - /lib: Essential shared libraries and kernel modules.
  - /opt: Optional add-on application software packages.
  - /mnt: Temporary mount point for filesystems.

    - /media: Mount points for removable media.
    - /srv: Site-specific data served by this system.

Remotely Copying Files

- scp:
    - **Remote to Local**: scp username@remote_host:/path/to/remote_file /path/to/local_destination
    - **Local to Remote**: scp /path/to/local_file username@remote_host:/path/to/remote_destination
- rsync: More robust, can resume transfers, good for large files/directories.
    - **Remote to Local**: rsync -avz username@remote_host:/path/to/remote_source /path/to/local_destination
    - **Local to Remote**: rsync -avz /path/to/local_source username@remote_host:/path/to/remote_destination

Environment Variables ⚙️

- Dynamic named values that can affect the way running processes behave.
- **Setting**:
    - Temporary (current shell): VARIABLE_NAME="value"
    - Export to child processes: export VARIABLE_NAME="value"
- **Listing**:
    - env: Lists all environment variables.
    - printenv: Lists all or specific environment variables.
    - echo $VARIABLE_NAME: Prints the value of a specific variable.
- **Common Variables**: PATH, HOME, USER, SHELL, LANG.

.bashrc **vs.** .bash_profile **vs.** .environment

- .bash_profile **(or** ~/.profile **for some shells/distros,** ~/.bash_login**)**:
    - Read by **login shells** (e.g., when you SSH in or log in on the console).
    - Used for commands that should run once per session (e.g., setting PATH).

- .bashrc:
  - Read by **interactive non-login shells** (e.g., when you open a new terminal window).
  - Often sourced by .bash_profile to ensure consistency.
  - Used for aliases, shell functions, prompt settings.
- /etc/environment **(or** ~/.pam_environment **for user-specific PAM settings)**:
  - Not a shell script, but a file read by PAM (Pluggable Authentication Modules) usually at login.
  - Sets **system-wide** (for /etc/environment) or **user-specific** (for ~/.pam_environment) environment variables available to all processes started after login, regardless of shell.
  - Format: VARIABLE_NAME="value" (no export).
  - **Note**: Not all systems use /etc/environment in the same way; behavior can vary. ~/.profile or shell-specific files are often more reliable for user environment.

Java & Tomcat

Java Memory Increase & JVM Problems ☕

- **JVM Heap Memory**: Where Java objects live.
  - -Xms<size>: Sets initial heap size (e.g., -Xms512m).
  - -Xmx<size>: Sets maximum heap size (e.g., -Xmx1024m).
  - These are set as JVM arguments when starting a Java application.
- **Common JVM Problems & Solutions**:
  - OutOfMemoryError: Java heap space: Increase -Xmx. Analyze heap dumps (using tools like jmap, Eclipse MAT) to find memory leaks.
  - OutOfMemoryError: PermGen space **/** Metaspace: (Older Java versions used PermGen; Java 8+ uses Metaspace). Stores class metadata.
    - PermGen: -XX:PermSize=<size>, -XX:MaxPermSize=<size>

        - Metaspace: -XX:MetaspaceSize=<size>, -XX:MaxMetaspaceSize=<size>
    - **High CPU Usage**: Profile the application (e.g., using jstack for thread dumps, JProfiler, YourKit) to identify hot spots.
    - **Slow Performance**: Could be due to insufficient memory (causing excessive Garbage Collection), inefficient code, or external factors. Monitor GC logs (-verbose:gc, -Xloggc:<file>).

Tomcat Server Basics 🐘

- Open-source Java servlet container and web server. Executes Java Servlets and JSPs.
- **Directory Structure (key dirs in** $CATALINA_HOME**)**:
    - bin: Startup/shutdown scripts (startup.sh, shutdown.sh, catalina.sh).
    - conf: Configuration files (server.xml, web.xml, context.xml).
    - webapps: Deploy your web applications (WAR files) here.
    - logs: Tomcat logs (catalina.out, access logs).
    - lib: Tomcat's JAR files and common libraries for webapps.
- server.xml: Main configuration file. Defines:
    - **Connectors**: Handle client connections (e.g., HTTP on port 8080).
    - **Engine, Host, Context**: Elements for request processing and webapp deployment.
- **Deployment**: Drop WAR files into the webapps directory. Tomcat can auto-deploy.
- **Memory for Tomcat**: Set JVM options (like -Xms, -Xmx) in setenv.sh (in bin directory - you might need to create it) via the CATALINA_OPTS or JAVA_OPTS environment variables.

Security & Scripting

Security Basics 🛡️

- **Principle of Least Privilege**: Users/processes should only have the permissions necessary to perform their tasks.

- **Keep Systems Updated**: Apply security patches regularly (sudo apt update && sudo apt upgrade or sudo yum update).
- **Strong Passwords & Password Policies**: Enforce complexity, rotation.
- **Firewall (e.g.,** ufw**,** firewalld**,** iptables**)**: Control incoming/outgoing network traffic.
- **Disable Unnecessary Services**: Reduce attack surface.
- **Regular Backups**: Essential for recovery.
- **Monitor Logs**: Check /var/log/auth.log (login attempts), /var/log/syslog or /var/log/messages for suspicious activity.
- **Use SSH Keys**: More secure than passwords for SSH access. Disable password authentication if possible.
- **File Integrity Monitoring (e.g., AIDE, Tripwire):** Detect unauthorized file changes.
- **Intrusion Detection Systems (IDS)**: (e.g., Snort, Suricata) can detect malicious activity.

Shell Scripting Basics 📜

- Automating tasks using shell commands in a script file.
- **Shebang**: First line specifies the interpreter (e.g., #!/bin/bash).
- **Variables**: name="value"; access with $name.
- **Command Substitution**: output=$(command) or ` command ` .
- **Conditional Statements**:
  - if [condition]; then ... elif [condition]; then ... else ... fi
- **Loops**:
  - for item in list; do ... done
  - while [condition]; do ... done
- **Functions**:

Bash

```
my_function() {
  echo "Hello from function"
```

```
 return 0
}
my_function # Call function
```

- **Input/Output**: echo (output), read (input).
- **Exit Status**: $? holds the exit status of the last command (0 for success).
- Make scripts executable: chmod +x script_name.sh.

Real-Life Problem Solved (Example Outline)

You'll need to tailor this with a genuine experience.

- **Problem**: Describe a specific issue. (e.g., "A critical Java application on a production server was experiencing frequent OutOfMemoryError: Java heap space errors, leading to service disruptions.")
- **Environment**: Linux distribution, application server (e.g., Tomcat), Java version.
- **Troubleshooting Steps**:
  1. **Initial checks**: ps aux | grep java (check process), free -m (system memory), top (CPU/memory usage).
  2. **Log analysis**: Reviewed application logs, Tomcat logs (catalina.out), and GC logs (if enabled).
  3. **JVM arguments**: Checked current -Xms and -Xmx settings in setenv.sh or the Tomcat startup script.
  4. **Heap Dump Analysis**: Used jmap -dump:live,format=b,file=heap.bin <pid> to get a heap dump. Analyzed it with Eclipse MAT (Memory Analyzer Tool).
- **Root Cause**: (e.g., "MAT revealed a memory leak caused by a specific class holding onto large collections of objects that were no longer needed.") or (e.g., "The -Xmx value was set too low for the application's workload, especially during peak hours.")
- **Solution**: (e.g., "Worked with developers to fix the code leak." or "Increased the -Xmx to an appropriate value (e.g., from 1GB to 2GB) after monitoring resource usage and ensuring the server had sufficient physical memory. Monitored GC activity post-change to confirm improvement.")

- **Outcome**: (e.g., "Reduced OutOfMemoryError occurrences significantly, improving application stability and user experience.")