Quick Reference Summary

- **Linux Fundamentals**
  - Everything is a file, hierarchical file system, multi-user environment.
- **Bash & Commands**
  - Basic commands, environment variables, scripting.
- **Text Editing**
  - Vim (modal), Nano (user-friendly), or GUI editors for code/config.
- **File/Directory Operations**
  - `ls`, `cp`, `mv`, `mkdir`, `touch`, wildcards, absolute vs. relative paths.
- **Permissions & Users**
  - `rwx` structure, `umask`, special bits (sticky, setGID), `su`, `sudo`.
- **Processes & Services**
  - `ps`, `pstree`, `top`, `nice`, `kill`, background/foreground jobs, `systemctl`.
- **Libraries & Packages**
  - Static vs. dynamic libraries, `ldconfig`, `yum/dnf` for installing/updating/removing packages.

Overview

- **Operating System (OS):** Manages computer resources; provides interfaces (CLI/GUI) for users and applications.
- **Linux:** An open-source OS known for its modularity, security, and scalability (from personal devices to supercomputers).
- **Advantages of Linux:**
  - **Open source** (no license fees, community support)
  - **Modular & Extensible** (use only needed components; can modify/extend code)
  - **Resource-efficient** (lower overhead, ideal for servers)
  - **Low vulnerability** to malware and viruses

Components of Linux

- **Kernel:** Core OS component; manages hardware (memory, CPU, devices)
- **User Space:** Where user applications and utilities run
- **File System:** Hierarchical structure (root `/`, subdirectories `/bin`, `/etc`, `/home`, etc.)
- **Daemons:** Background services that provide various OS or app functions (e.g., `sshd`, `syslogd`)

Linux Distributions (Distros)

- **Common Distros:** Ubuntu, Debian, Fedora, CentOS, Amazon Linux, Red Hat Enterprise Linux (RHEL), SUSE, etc
- **Amazon Linux 2 & 2023:** Optimized for AWS EC2 instances, includes AWS integration tools, security/hardening features

- **Bottlerocket:** Minimal OS specialized for container clusters

---

Bash Shell & Command-Line Basics

Bash (Bourne-Again Shell)

- **Default shell** on most Linux distributions
- **CLI & Scripting:** Type commands at the prompt (`$`) or run shell scripts
- **Command Syntax:**
    - ***CommandOption Argument***
    - Example: `man -i whoami` (command: `man`, option: `-i`, argument: `whoami`)

Common Commands

- **Basic Informational Commands:**
    - `date` (current date/time), `uptime` (server runtime/load), `hostname` (system name), `cal` (calendar), `bc` (calculator), `history` (previous commands), `id` (user identity)
- **Help & Info:**
    - `man <command>` (manual pages), `whatis <command>` (short description), `which <command>` (executable path)

Shell Variables & Environment Variables

- **Declaring Variables:** `name=value`, e.g. `myvar="Hello World"`
- **Displaying Variables:** `echo $myvar`
- **Environment Variables:**
    - `$USER`, `$HOME`, `$SHELL`, `$PATH`
- **Aliases:**
    - Create: `alias ll='ls -l'`
    - Remove: `unalias ll`

Linux Editors

Command-Line Editors

- **Vim:** Powerful, modal editing (modes: normal, insert, visual, command-line). Steep learning curve but highly efficient
- **Nano:** User-friendly, simpler interface, ideal for beginners

GUI Editors

- **Gedit, Atom, Sublime Text, etc.** provide a graphical interface with menus and toolbars

Essential Editor Commands

- **Vim:**
  - Open file: `vim filename`
  - Modes: `i` (insert), `Esc` to return to normal, `:` for command-line mode
  - Save/Quit: `:w`, `:q`, `:wq`
- **Nano:**
  - Open file: `nano filename`
  - Bottom bar shows shortcuts (e.g., `Ctrl+O` to save, `Ctrl+X` to exit)

Navigating & Managing Files/Directories

Key File Concepts

- **Everything is a file** in Linux (commands, hardware, directories).
- **Listing Files/Dirs:** `ls`
  - Important options: `-l` (long format), `-a` (show hidden), `-h` (human-readable), `-R` (recursive).
- **File Naming Conventions:** Case-sensitive, avoid special characters, no duplicates in the same directory.

Navigation Commands

- `pwd` (print working directory)
- `cd` (change directory)
  - `cd /absolute/path`
  - `cd relative/path`
  - `.` (current dir), `..` (parent dir)
- `tree` (shows directory structure in tree format)
- `ls` (list contents)

Creating & Managing Files/Dirs

- `touch file.txt` (create empty file)
- `cp src dest` (copy)
- `mv src dest` (move/rename)
- `mkdir newdir` (make directory)
- `rmdir emptydir` or `rm -r dir` (remove directory)

Wildcards

- `*` (any number of characters), `?` (exactly one char), `[ ]` (match characters in brackets)

Managing File Permissions & Users

Default Permissions & umask

- New files often default to `rw-r--r--` (644)
- New directories often default to `rwxr-xr-x` (755)
- **umask** sets default permissions. Example: `umask 002` → directories get `775`, files get `664`

Special Permissions

- **Sticky bit (t):** Prevents users from deleting files they don't own in a directory (e.g., `/tmp`)
- **SetGID (s):** Inherit group ownership. For files, process runs with file's group ID. For directories, new files inherit parent's group.

User Permission Levels

- **Standard User:** Limited rights
- **Root User:** Full system privileges; use caution
- **su <user>:** Switch user (requires that user's password)
- **sudo <command>:** Run a command with elevated privileges (requires own password, if configured in `/etc/sudoers`)

Managing Users & Groups

- **`/etc/passwd`:** Contains user account info
- **`/etc/group`:** Contains group info
- **useradd / usermod / userdel:** Commands for adding/modifying/deleting user accounts
- **gpasswd / usermod:** Add a user to a group

Programs, Processes, Jobs, and Daemons

Programs & Processes

- **Program:** Executable file/software.
- **Process:** A running instance of a program (PID = unique process ID).
- **ps:** Snapshot of processes.
- **pstree:** Shows processes in a tree format.

- **nice / renice:** Adjust process priority (`-20` highest, `19` lowest).

Jobs & Job Control

- **Foreground/Background:**
  - `&` to run a command in the background.
  - `fg %<jobID>` bring job to foreground; `bg %<jobID>` send job to background.
  - `jobs` lists active jobs, their IDs, and states.
- **nohup command & disown:** Keep processes running after logout.

Daemons

- **Long-running background processes** (e.g., `sshd`, `crond`).
- Often started at boot, run until stopped/rebooted.
- *System daemons* (systemd, sshd), *Network daemons* (httpd), *File system daemons* (nfsd), etc.

Process Monitoring & Signals

- **top:** Real-time system usage; press keys like `M` (sort by memory), `P` (sort by CPU).
- **kill [OPTIONS] <PID>:** Send signals (e.g., `kill -9 <PID>` force terminate).
- **pidof / pgrep:** Find process IDs.

Scheduling Tasks

- **at:** One-time scheduling (`at TIME`).
- **cron:** Regular scheduling (reads from `crontab` or `/etc/cron.*` directories).

Managing Services (systemd)

- **systemd:** Manages services (units).
- **systemctl:** Start, stop, enable, disable services.
  - `systemctl start <service>`
  - `systemctl status <service>`
  - `systemctl enable <service>`

Shared Libraries & Package Management

Shared Libraries

- **Static Libraries (.a):**
  - Compiled into executable at build time.
  - No external dependency at runtime but larger file size, more memory usage.

- **Dynamic Libraries (.so):**
    - Linked at runtime.
    - Smaller executables, shared memory usage.
    - Must be present on the system; missing/corrupt libraries break apps.

Library Configuration

- **ld.so.conf & /etc/ld.so.conf.d/*.conf:** Directories where the linker searches for libraries.
- **ldconfig:** Updates cache and config for shared libraries. `ldconfig -v` lists libraries, `ldconfig -p` shows library cache.

Package Managers

- **RPM:** Base for Red Hat, CentOS, Amazon Linux.
- **YUM & DNF:** High-level wrappers around RPM that resolve dependencies automatically. Common commands:
    - `yum/dnf install <pkg>`
    - `yum/dnf remove <pkg>`
    - `yum/dnf update [<pkg>]`
    - `yum/dnf search <keyword>`
    - `yum/dnf list installed`
    - `yum/dnf repolist`
    - `yum/dnf history`
- **Zypper:** Used by SUSE-based distros.

Sample Workflows

- **Install a Package:** `yum install htop`
- **Update a Package:** `yum update htop`
- **Remove a Package:** `yum remove htop`
- **Check Installed Packages:**`yum list installed`

Exploration of the Concepts in Linux

Investigating Why Linux Is a Good Choice

When a solutions architect investigates why Linux might be an excellent operating system for developing a new application, **two notable features** stand out. First, **Linux's modular design** allows installations to be tailored precisely to the workload, enabling teams to include only the components they need. Second, **certain Linux distributions target specific workloads** such as servers, edge devices, or containers, ensuring an optimal setup for a particular development environment. Together, these features give developers a customized and efficient platform ideally suited for various cloud-based solutions.

Key Advantages of the Linux OS

Exploration of the Concepts in Linux

A support engineer preparing a presentation on Linux's benefits should emphasize two primary strengths. First, **Linux is highly portable**, meaning it can run on everything from small, **embedded processors** to **large-scale enterprise servers**. This **versatility** simplifies migrating and scaling applications in the cloud. Second, **Linux can support multiple users and applications concurrently**, making it an excellent choice for **multi-tenant** or collaborative development environments where **many processes run in parallel**.

Choosing the Right Linux Distribution for Containers

A development team that wants to **run containerized applications** on Amazon EC2 can benefit greatly from using **Bottlerocket**. Bottlerocket is **purpose-built for containers**, containing only the **components essential for running them**. By focusing on **minimalism**, Bottlerocket reduces both update overhead and resource usage, **freeing up system resources** for container workloads and **increasing reliability** in a cloud setting.

Why Amazon Linux 2 Works Well on EC2

When explaining why **Amazon Linux 2** is suitable for deployments on Amazon EC2, two features stand out. First, **its kernel is optimized for EC2 instances**, leveraging Amazon's insights about the virtual hardware environment to enhance performance. Second, **it supports kernel live patching**, allowing **critical updates without requiring a reboot**. These features help reduce downtime and streamline operations for cloud-based development teams

Finding the Right AMI for Legacy Linux Distributions

A developer needing to migrate an application that runs on an older Red Hat Enterprise Linux version can turn to the **AWS Marketplace**. This service provides a **broad selection of Amazon Machine Images** (**AMIs**) from both commercial and community sources. It's designed to help cloud teams quickly find the exact distribution and version they need, ensuring compatibility with legacy applications without compromising on modern cloud infrastructure.

Understanding `ls -la /home/ec2-user/project`

The command `ls -la /home/ec2-user/project` **lists all files and directories** (including **hidden** ones) under `/home/ec2-user/project`, **displaying their permissions and other details** in a **long format**. This is a fundamental Linux file system operation, critical for understanding which items exist in a directory and who has access to them—key information for managing cloud-based systems.

Changing File Permissions

The **`chmod`** command is essential for a Linux user or developer who needs to **adjust file or directory permissions**. Modifying permissions ensures that **owners**, **groups**, or **all system users** can **read**, **write**, or **execute** files as required. This is particularly useful for maintaining security in a cloud environment where multiple services and users might share resources.

Redirecting Output to Another Program

Exploration of the Concepts in Linux

The **pipe operator (`|`)** in Linux is a powerful tool for **redirecting the output of one program to another**. This **chaining of commands** is at the heart of effective shell scripting and command-line work. For instance, it allows you to **pass data from a listing command directly into a search function** without manually creating temporary files—a critical skill for **streamlining workflows** in cloud or DevOps tasks.

Navigating the Linux File System

To **move around** the directory structure in Linux, developers rely on the **`cd`** (**change directory**) command. This straightforward command is vital for accessing code repositories, configuration files, and logs in a well-structured environment, whether the developer is working locally or remotely on a cloud instance.

Creating a New Directory

The **`mkdir`** (**make directory**) command does exactly what its name implies: **creates a new folder**. Having well-organized directories is crucial in any software environment, particularly when managing multiple projects and deployments in the cloud.

Quitting Vim

In the popular Linux text editor Vim, typing **`:q`** in normal mode is the quickest way to quit. Vim's modal interface can initially seem complex, but basic commands like this are essential. A developer working in a cloud environment might often update configuration files or code directly on remote servers, so mastering Vim commands can significantly speed up their workflow.

Quick Text Replacement with `sed`

When a Linux administrator needs to **replace occurrences of one word with another in a text file**, **`sed`** (**stream editor**) provides an efficient, automated solution. Instead of opening a file interactively, `sed` **processes files or streams of data directly**. This capability is especially advantageous in DevOps pipelines or cloud scripts where large-scale or repetitive text manipulation tasks are common.

Applying SGID Permissions to a File

Using **`chmod g+s file.py`** grants SGID (**Set Group ID**) permissions to `file.py`. This means that when the file runs, **it runs with the permissions of its group**, a setup often used for shared development environments. SGID helps ensure collaborative tasks can be **executed with consistent permissions**, supporting productivity in multi-developer cloud projects.

Managing Group Membership with `gpasswd -M`

The **`-M`** option in `gpasswd` lets you **set the entire list of group members** at once, replacing existing members with a new list. This can be especially useful when onboarding or offboarding multiple developers at the same time in a cloud project, ensuring that only authorized users remain in critical groups.

Exploration of the Concepts in Linux

Interpreting `umask` Output

An output like **`0033`** from the `$ umask` command defines the **default file and directory permissions** for **newly created items**. By default, umask sets **which permissions are turned off** when a new file or directory is created. Developers and system administrators use this to maintain secure defaults, which is particularly important when managing code repositories or configuration files in cloud environments with many users.

Checking Traceroute Results

When using **`traceroute`** to assess network paths, two vital indicators of acceptable network performance are **the elapsed time of each hop** and **the total number of hops**. Shorter times and fewer hops generally indicate better throughput and lower latency, which is crucial for cloud application developers aiming to optimize user experience and troubleshoot connectivity issues.

Typical Network Interface for Application Access

On some servers, **`eth0`** serves as the default or primary **network interface** for external communication. Understanding which interface handles traffic helps developers and operations teams configure firewalls, routing, and monitoring tools correctly in a cloud infrastructure, ensuring applications remain accessible. Note: on RHEL-based systems like Amazon Linux, the default network interface is **ensX**, where X is a digit.

Changing Process Priority

The **`top`** utility and the **`renice`** command are both effective for **adjusting the priority** of **running processes** in Linux. `top` offers a real-time view of system usage, allowing quick, interactive priority changes, while `renice` directly **sets a new priority** using the process ID. Note: **nice** allows you to **start a new process** with a particular priority, **renice** updates the priority of a running process.

Searching Files for Specific Content

Combining **`find`** and **`grep`** is a common method for searching specific terms within certain files. For example, using `find /usr/share -name "*.conf" -size -100k -exec grep email {} \;` locates all `.conf` files **under 100 KB in size** and looks for lines containing "*email*" in each. These commands are powerful in cloud environments where configurations and logs may be scattered across multiple directories or servers.

Removing a Package with RPM

To **uninstall software** via the RPM Package Manager, **`rpm -ev BadPackage`** removes (**erases**) the package and provides **verbose output**. Package management is a foundational skill for maintaining clean and efficient application servers. Regularly removing unneeded packages is an important practice to minimize attack surfaces and resource overhead in a production cloud environment.