Red Hat System Administration I 9.3(RH-124)

Certificate of Completion

Fig 0.1: Certificate of completion

Note: Certificate of completion received post completion of course.

Chapter 1: Get Started with Red Hat Enterprise Linux

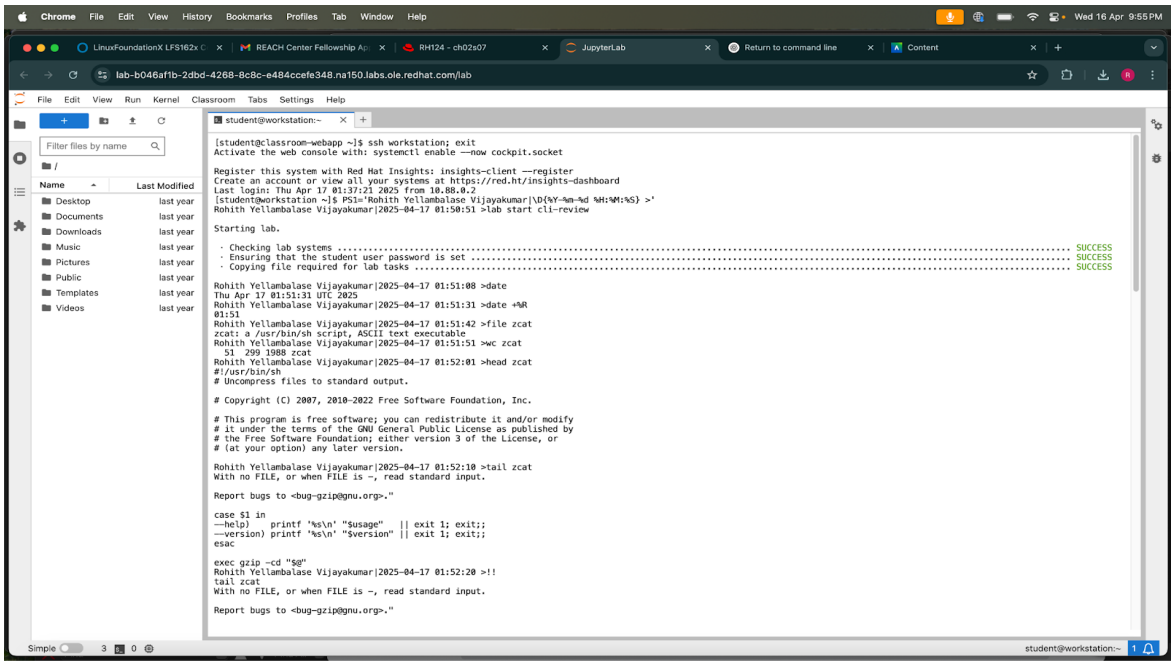
Summary:

- Fedora: Community-driven, cutting-edge updates every 6 months, suitable for developers but not for production use.
- CentOS Stream: A continuous delivery platform upstream of RHEL, great for testing and contributing but not ideal for production.
- RHEL: Stable, commercially supported, used by enterprises with long-term support, requires a subscription (free Developer version available).
- The command line allows user input via a shell, where commands consist of the program, options (modifying behavior), and arguments (specifying input).
- SSH (Secure Shell): Secure login protocol using public key authentication to encrypt connections, replacing password-based login.
- Workspaces: Virtual desktops that allow separate open applications/windows for better multitasking and organization.

Chapter 2: Access the Command Line

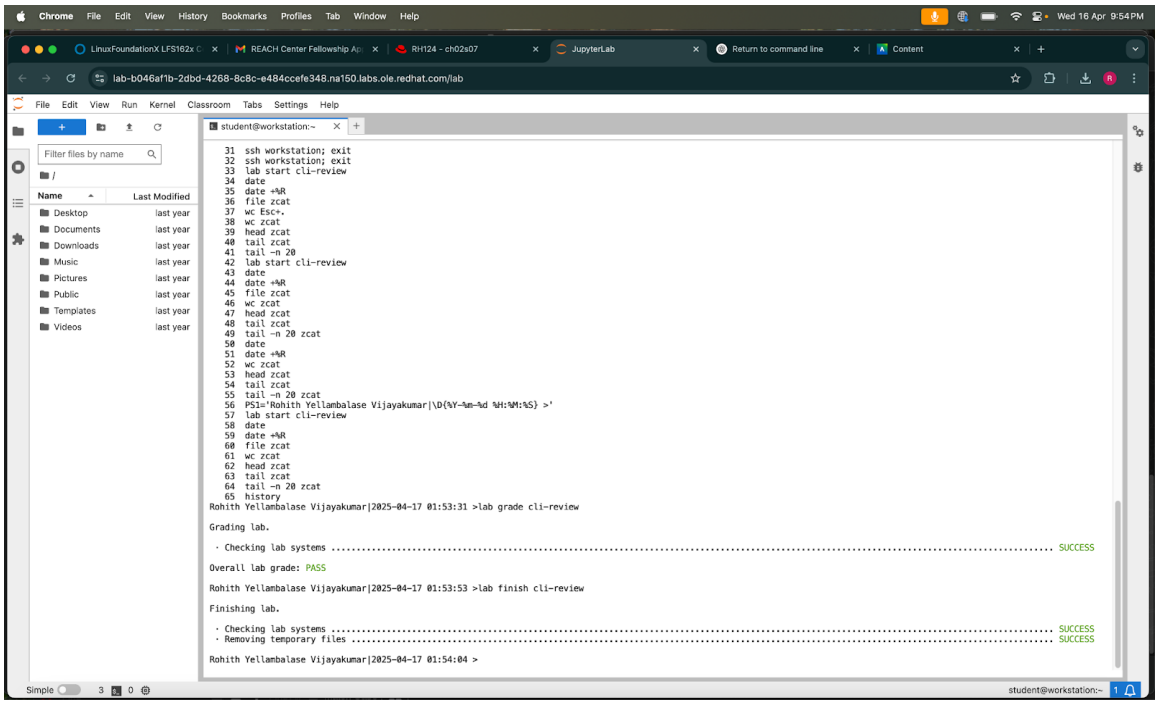
Lab: Access the Command Line

Fig 2.1:Lab Initialization and Command-Line Output Execution



Note: This figure highlights the successful execution of simple programs from the Bash shell command line. It also emphasises the ability to execute commands to identify file types and display parts of text files. Additionally, it showcases the use of Bash command history shortcuts for efficiently repeating commands or parts of commands.

Fig 2.2: *CLI Review Lab – Zcat File Analysis and Output Commands*



Note: This demonstrates the execution of CLI commands including `file`, `wc`, `head`, and `tail` on the `zcat` utility, along with environment setup and timestamp logging for the lab session passed successfully.

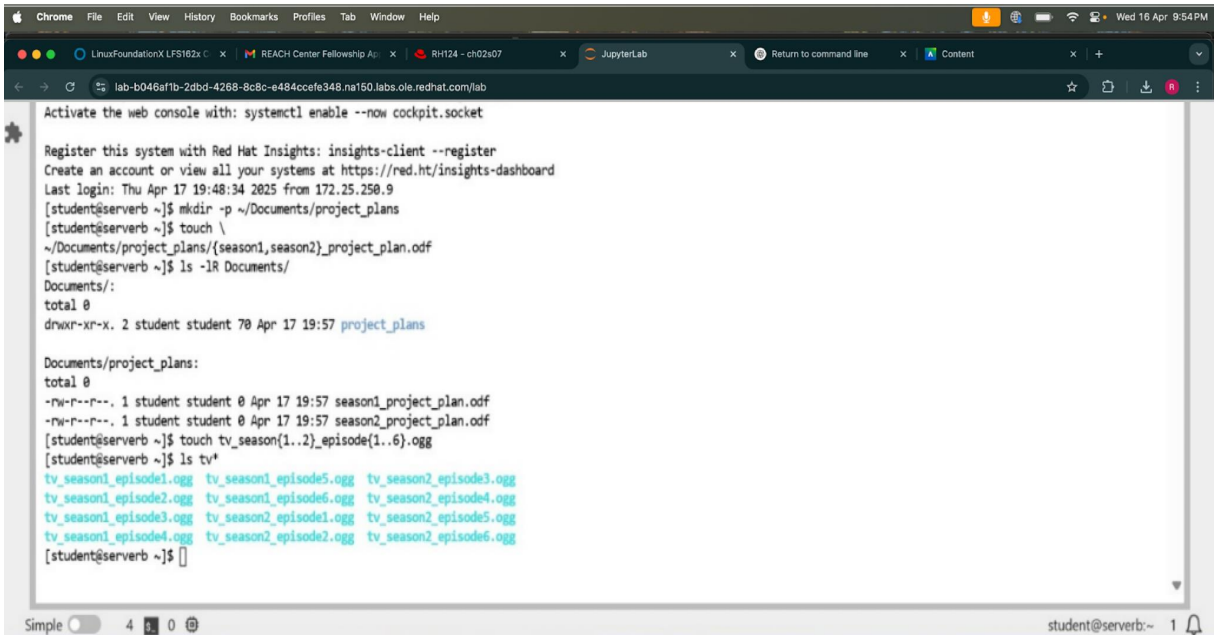
Summary:

- Command Basics and Multiline Commands: You can execute multiple commands on a single line using command1 ; command2, which allows for efficient task execution. Additionally, long commands can be split into multiple lines using the backslash \, helping keep your terminal commands clean and readable.
- File Type and Content Commands: The file command is used to determine the type of a file, such as whether it's text or binary. To view or concatenate file contents, the cat command is used, which prints the contents of one or more files in order. The head and tail commands allow you to view specific parts of a file, with head showing the beginning and tail displaying the end, providing flexibility when working with large files.
- Word Count Tool (wc): The wc command counts the number of lines, words, and characters in a file. It offers options such as -l to count lines, -w to count words, and -c for characters, helping users quickly gather statistics on text files.
- Command History and Navigation: The history command displays previously executed commands, making it easier to review and repeat past actions. You can use !number to run a specific command from the history list, or !string to repeat the last command that started with a certain string. Navigation through the history can also be done using the arrow keys (UpArrow, DownArrow, and Left/Right Arrows), allowing for fast corrections and re-execution of commands.
- Tab Completion: Tab completion is a helpful feature that auto-completes file paths, command names, and options. Pressing the Tab key once automatically completes the command or filename if it's unique, and pressing it twice shows all possible matches, improving productivity.
- Password Management: You can change your system password using the passwd command, a crucial tool for maintaining secure access to your system. This command helps ensure that users can easily update their login credentials.

Chapter 3: Manage Files from the Command Line

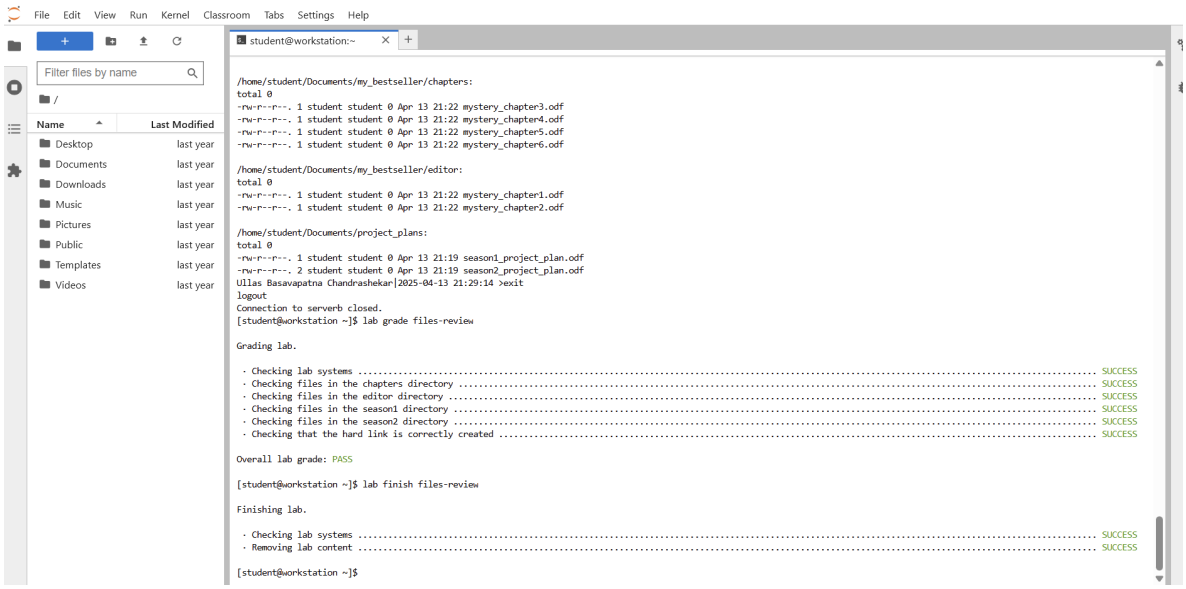
Lab: Manage Files from the Command Line

Fig. 3.1 File command to find file format



Note : Usage of the file command to determine the file format

Fig 3.2: Screenshot showing the successful execution of file creation, movement, and grading for the lab task.



Note: This image illustrates the terminal output for tasks performed during the "Manage Files from the Command Line" lab, where directories and files were created and organized using various Bash commands. The task also involved moving files to specific directories and using symbolic and hard links, with the final step showing the grading results.

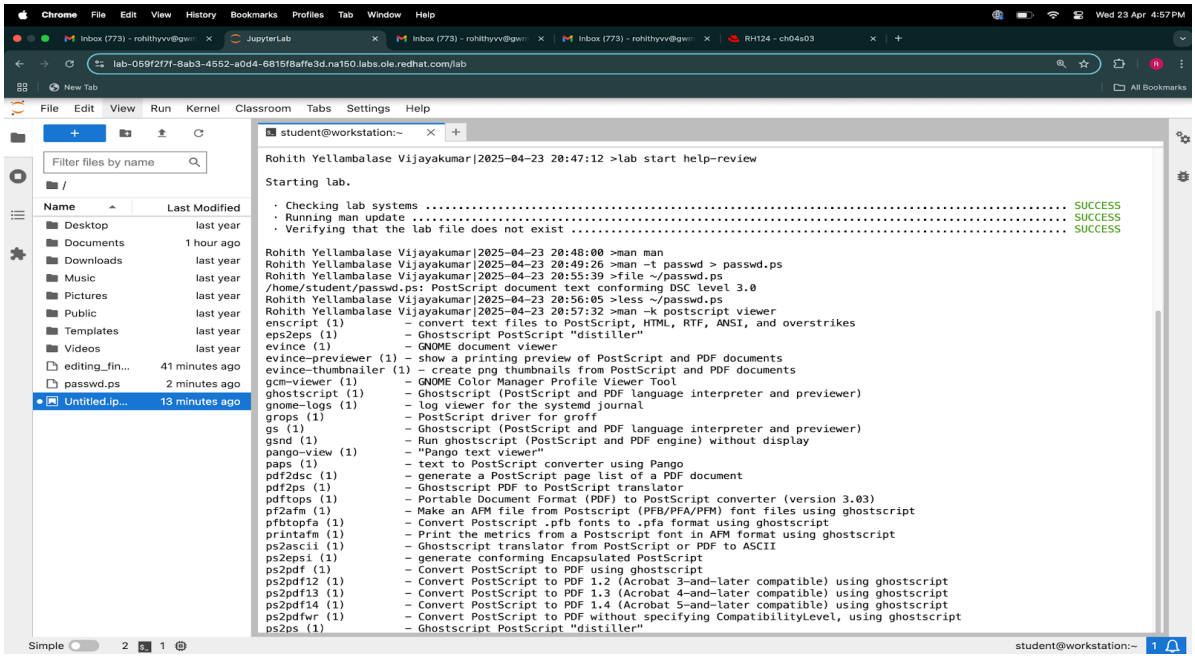
Summary:

- Linux directories are structured hierarchically with directories like /bin, /usr, /home, and /var, each serving specific system and user-related purposes.
- The pwd command shows the current directory, while cd allows navigation between directories using absolute or relative paths.
- File operations include commands like cp (copy), mv (move), and rm (remove), which can be used with various options for more specific tasks like recursive actions or forced deletion.
- Links in Linux, both hard and symbolic (soft), allow multiple names or shortcuts for the same file. Hard links directly point to the same data, while symbolic links are shortcuts that may break if the original file is removed.
- Shell features like pathname expansion (wildcards), brace expansion, tilde expansion, and variable expansion provide flexibility and efficiency in working with file paths and variables in Linux commands.

Chapter 4: Get Help in Red Hat Enterprise Linux

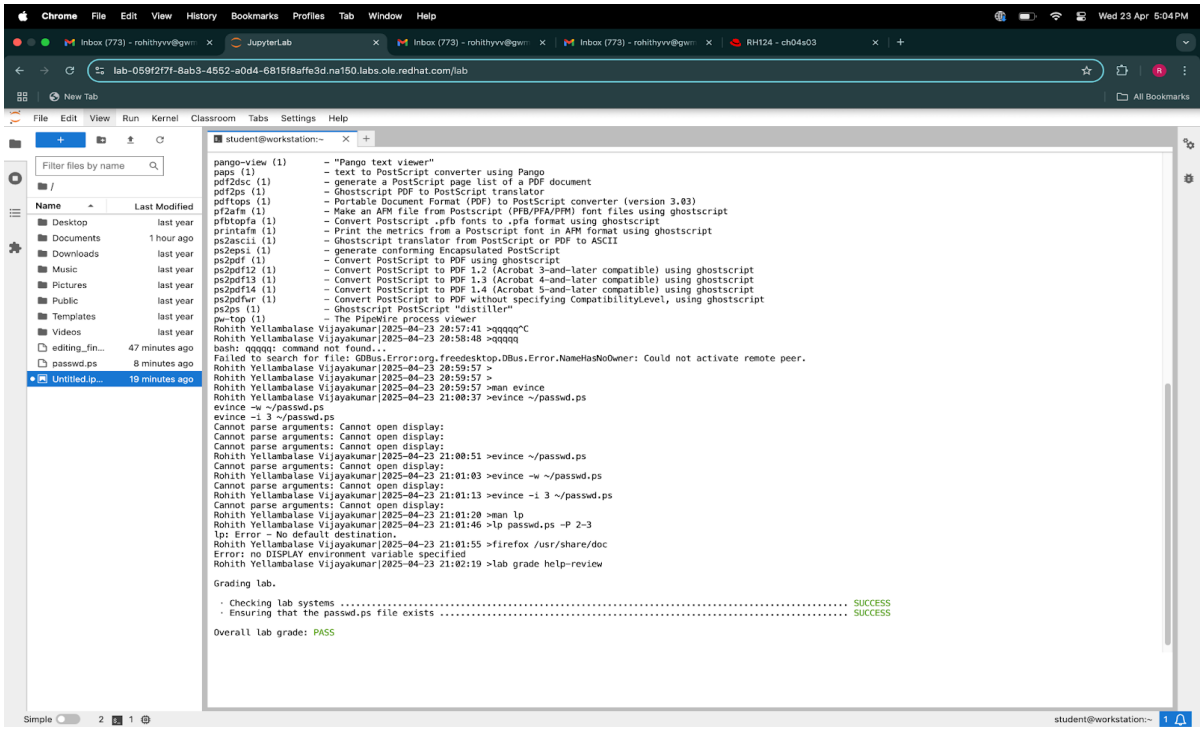
Lab: Get Help in RedÂ Hat Enterprise Linux

Fig. 4.1 PostScript Tools and Man Page Formatting in Linux



Note : This Linux terminal session demonstrating how to convert man pages into PostScript format using the `man -t` command and how to explore PostScript-related utilities like `ghostscript`, `enscript`, and document viewers.

Fig. 4.2: Verifying PostScript File Output in Linux Terminal



***Note:** Successfully explored PostScript tools, attempted multiple viewers, ensured file creation, and completed lab with a PASS grade and all SUCCESS checks. Lab successfully validated file existence and passed.*

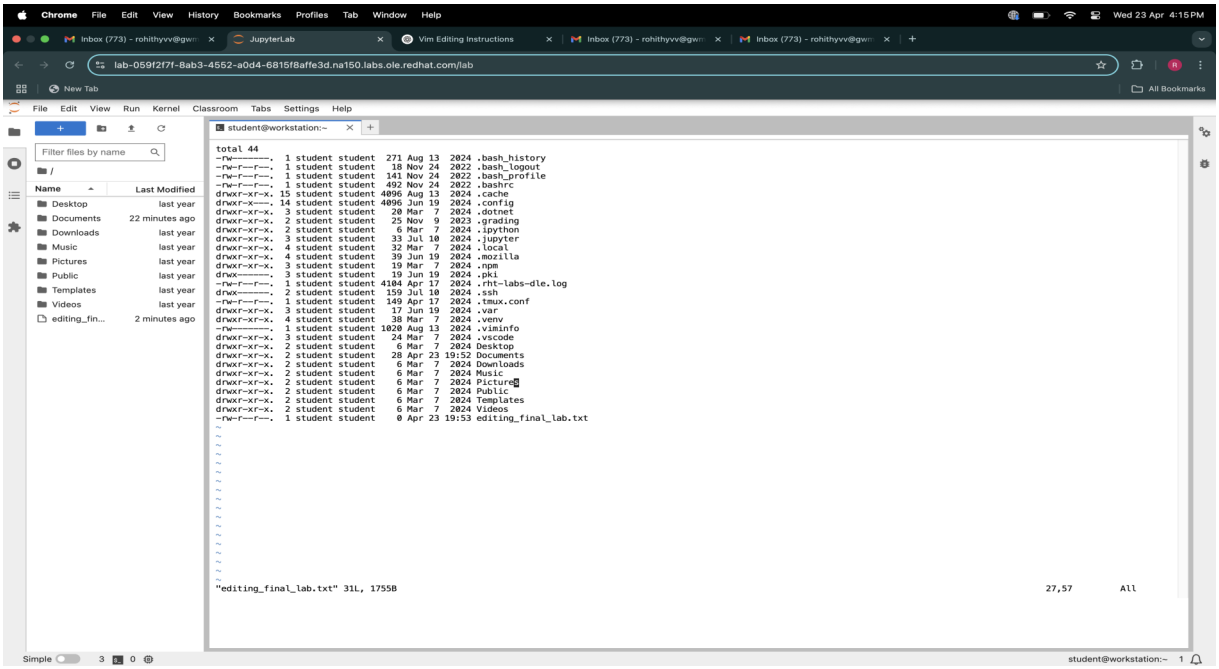
Summary:

- Man Pages: Man pages are built-in help documents in Linux that provide information on commands, files, functions, and more. Each section of the manual focuses on different content types, such as user commands, system calls, library functions, and kernel APIs.
- Man Page Structure: Man pages are organized into sections, with Section 1 covering user commands, Section 2 for system calls, and others for various topics like file formats, system administration, and more. Example: man passwd shows the command and man 5 passwd shows file format details.
- Search Functionality: You can search man pages by keyword using man -k <keyword>, or search within the entire content of man pages using man -K <keyword>, which can help you find relevant information on a specific topic.
- Navigation: To navigate through man pages, commands like Spacebar, PageUp, and PageDown allow you to scroll through content. You can also use Shift+G to jump to the end and Q to exit the man page. The search feature (/string) helps you quickly locate specific information within the man page.

Chapter 5: Create, View, and Edit Text Files

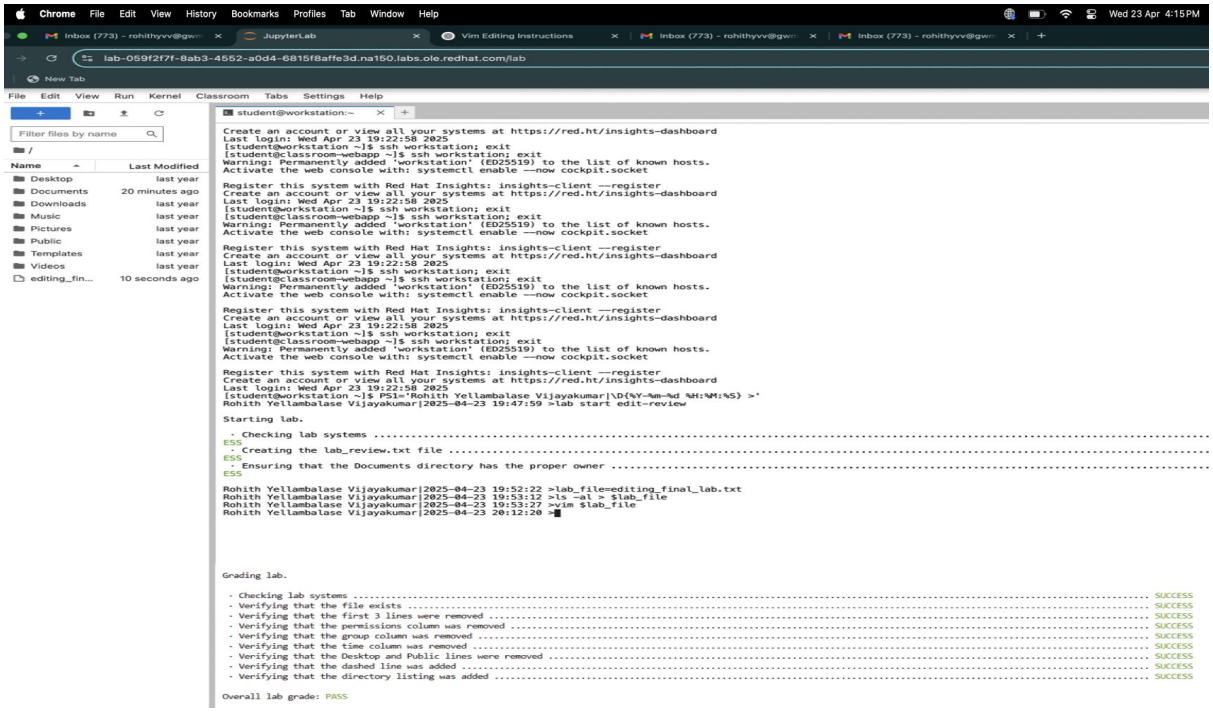
Lab: Create, View, and Edit Text Files

Fig 5.1: Verifying File Creation and Directory Contents in Home Folder



Note: *This screen confirms successful creation of* `editing_final_lab.txt` *using* `vim`*, and lists all user directories and hidden configuration files.*

Fig 5.1: Shows the terminal output from a Linux lab where I used Vim and shell commands to edit, manipulate, and format a text file as part of a system admin exercise.



Note: In this lab, I practised using Vim visual modes to remove lines and columns, created backups with timestamps, and used shell tools like ls, tee, and echo to modify and append data to a file for review and grading.

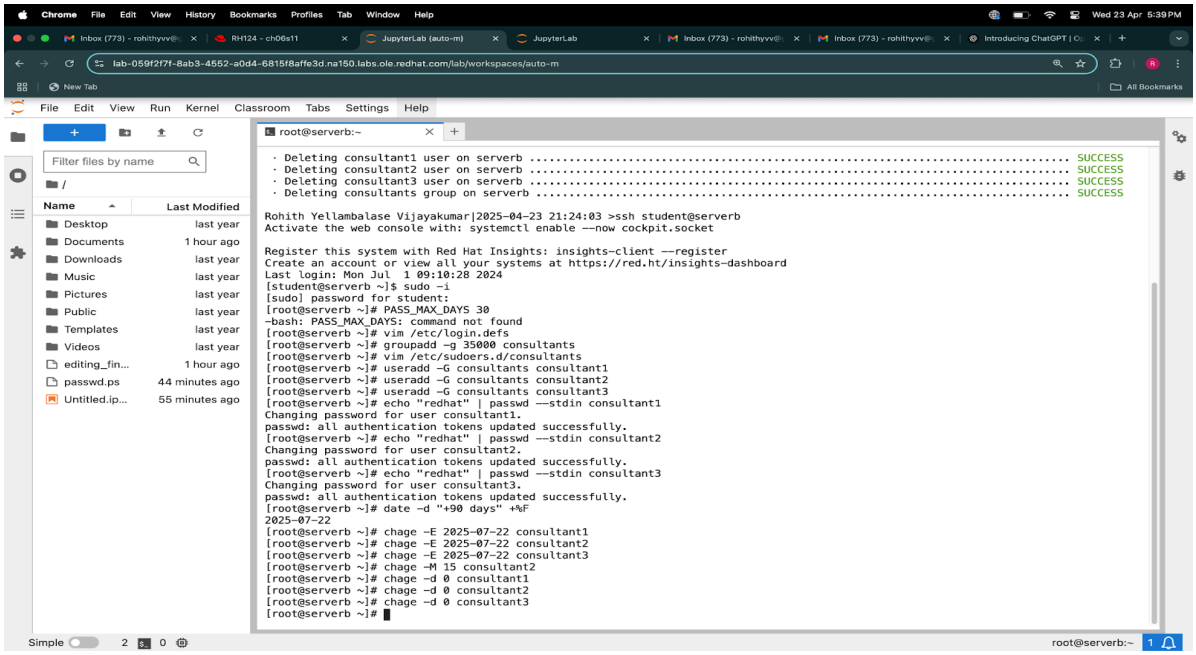
Summary:

- Standard output and error streams:
  When you run a command, normal results go to stdout (file descriptor 1) and error messages go to stderr (file descriptor 2). Both usually display on the screen unless redirected.
- Redirection operators:
  You can redirect output using > (overwrite), >> (append), 2> (redirect errors), and /dev/null (discard output). For example, command > out.txt 2> err.txt saves normal output and errors in separate files.
- Combining output and error:
  Use command > file.txt 2>&1 to send both stdout and stderr to the same file. In Bash, &> is a shortcut for this.
- Using pipelines (|):
  A pipeline passes the output of one command into another, like ls | wc -l to count files. This only passes stdout unless errors are redirected as well.
- The tee command:
  tee is used to split output so you can view it and save it simultaneously. Example: ls | tee list.txt displays the output and writes it to list.txt at the same time.

Chapter 6: Manage Local Users and Groups

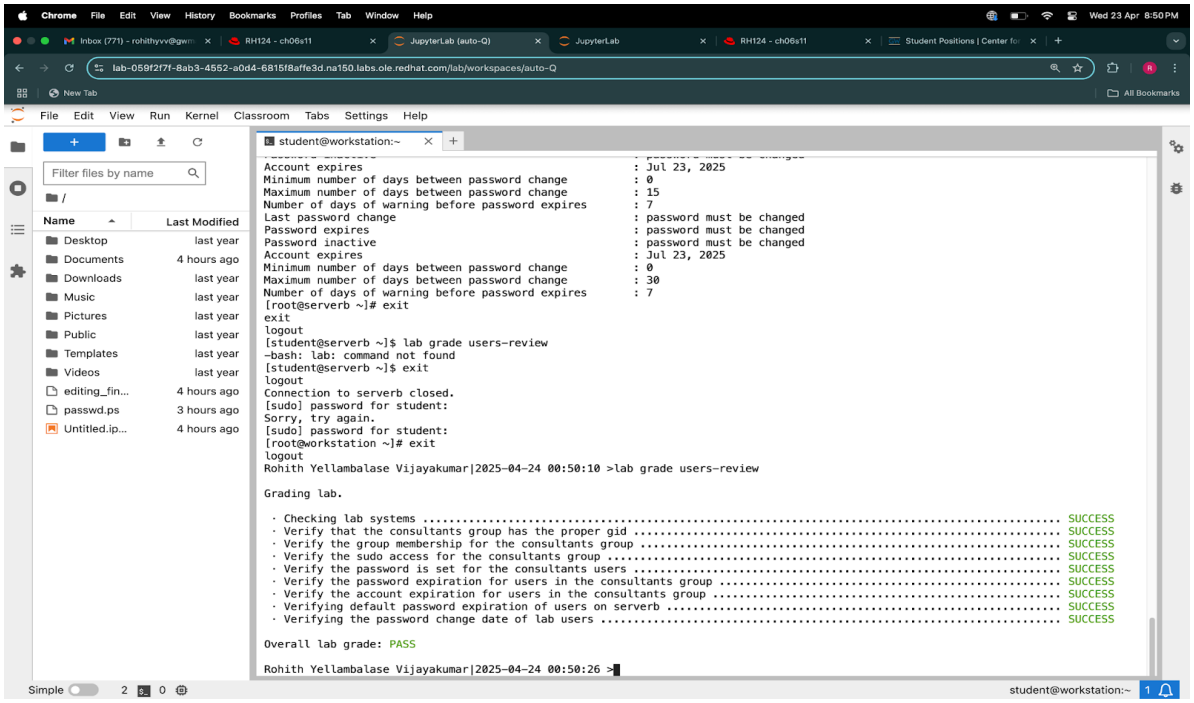
Lab: Manage Local Users and Groups

Fig 6.1: User and Group Management on Red Hat Server



Note: This lab demonstrates creating a new group `consultants`, adding users to it, and updating passwords securely using `passwd --stdin`, all with successful execution.

Fig 6.2: Shows the terminal output confirming successful user and group configuration tasks with a "PASS" grade in the users-review lab.



Note: In this lab, I created users and a supplementary group, assigned sudo privileges, and configured password aging policies. I set account expiry dates and forced password resets on first login. All configurations were verified successfully.

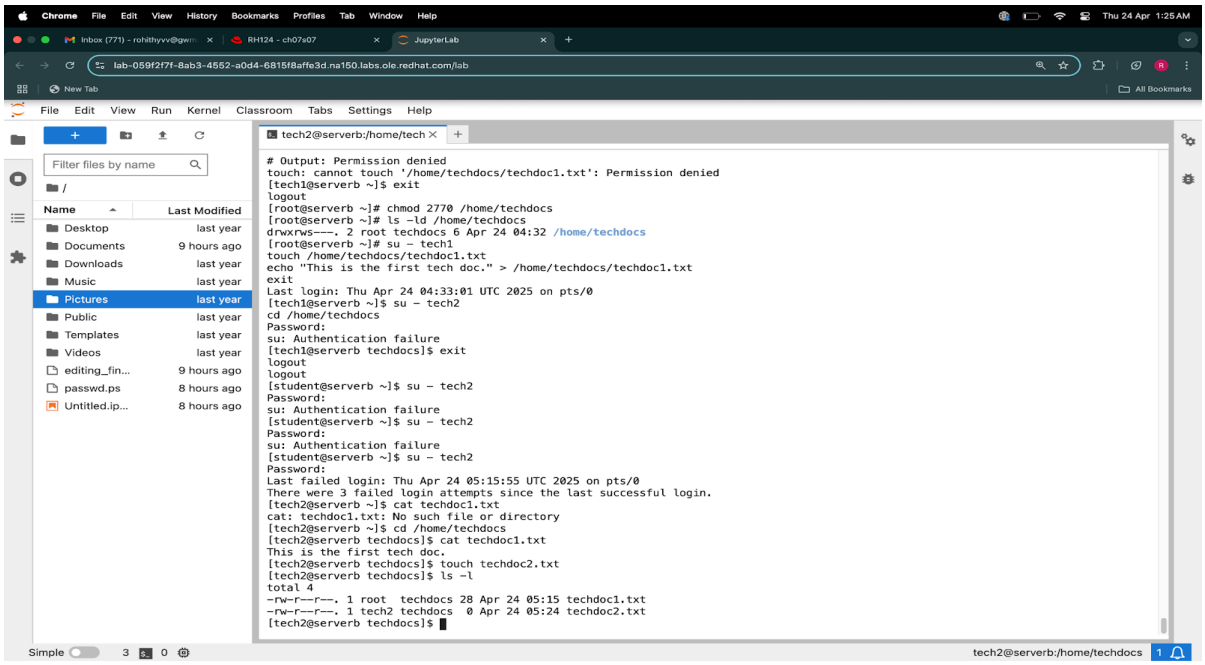
Summary:

- Creating Users and Groups:
  Red Hat allows you to manage local users with the useradd command and groups using groupadd. You can specify a unique Group ID (GID), create supplementary groups, and assign users to them during creation or later using usermod -G.
- Setting and Managing Passwords:
  The passwd command is used to set or change passwords for users. You can enforce password strength policies and trigger warnings like "BAD PASSWORD" if it doesn't meet length or complexity criteria. You can also force password changes at first login using chage -d 0.
- Password Aging and Expiry Policies:
  Global password policies are configured in /etc/login.defs (e.g., PASS_MAX_DAYS for maximum password lifetime). For user-specific policies, chage allows setting maximum days (-M), warning days (-W), and account expiration (-E).
- Granting Sudo Privileges Safely:
  Instead of modifying the main /etc/sudoers file, it's best practice to create role-based access rules in /etc/sudoers.d/. For example, %consultants ALL=(ALL) ALL gives all members of the consultants group full sudo rights.
- Account and Access Management:
  You can manage user accounts by setting expiration dates, modifying group membership, and controlling login behavior. Commands like chage, id, and groups help monitor and manage user details efficiently.

Chapter 7: Control Access to Files

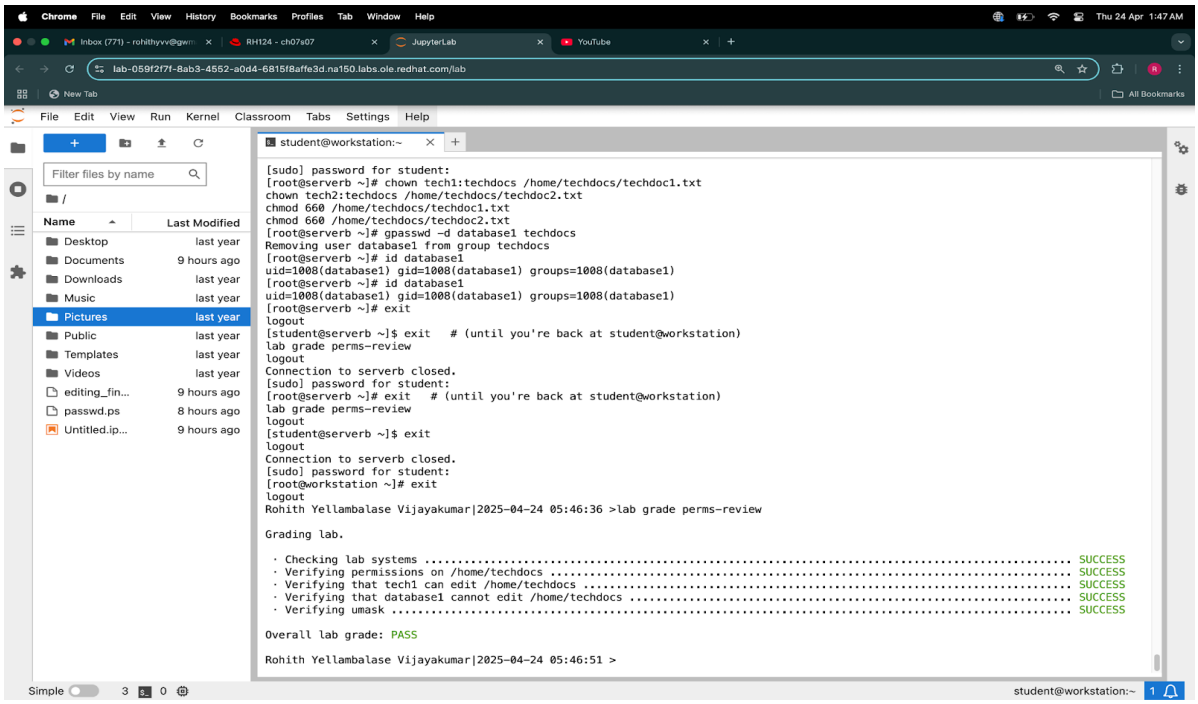
Lab: Control Access to Files

Fig 7.1: shows the successful completion of a file permission and group access lab, with all checks passing and a final "PASS" grade displayed in the terminal.



Note: Demonstrates setting group sticky bit (2770) on `/home/techdocs`, resolving permission issues, and verifying successful file creation and access by users `tech1` and `tech2`.

Fig 7.2: shows the successful completion of a file permission and group access lab, with all checks passing and a final "PASS" grade displayed in the terminal.



Note: In this lab, I created a shared directory for the techdocs group with specific group ownership and permissions. I used chmod with setgid and updated umask to control file creation access. Only users in the group could create or modify files, ensuring secure collaboration.

Summary:

- Understanding File Permissions:
  Every file and directory in Linux has permissions defined for the owner, group, and others. These include read (r), write (w), and execute (x) permissions, which can be viewed using ls -l and modified using chmod.

- Managing Group Collaboration:
  You can create a shared directory for a group and use the chgrp command to assign group ownership. Setting the setgid (chmod 2000) on the directory ensures all new files inherit the directory's group.

- Restricting Access for Others:
  Using appropriate permissions like 770 or 750, you can ensure only group members have access to read/write/execute files, while others are denied all access, enhancing security and collaboration.

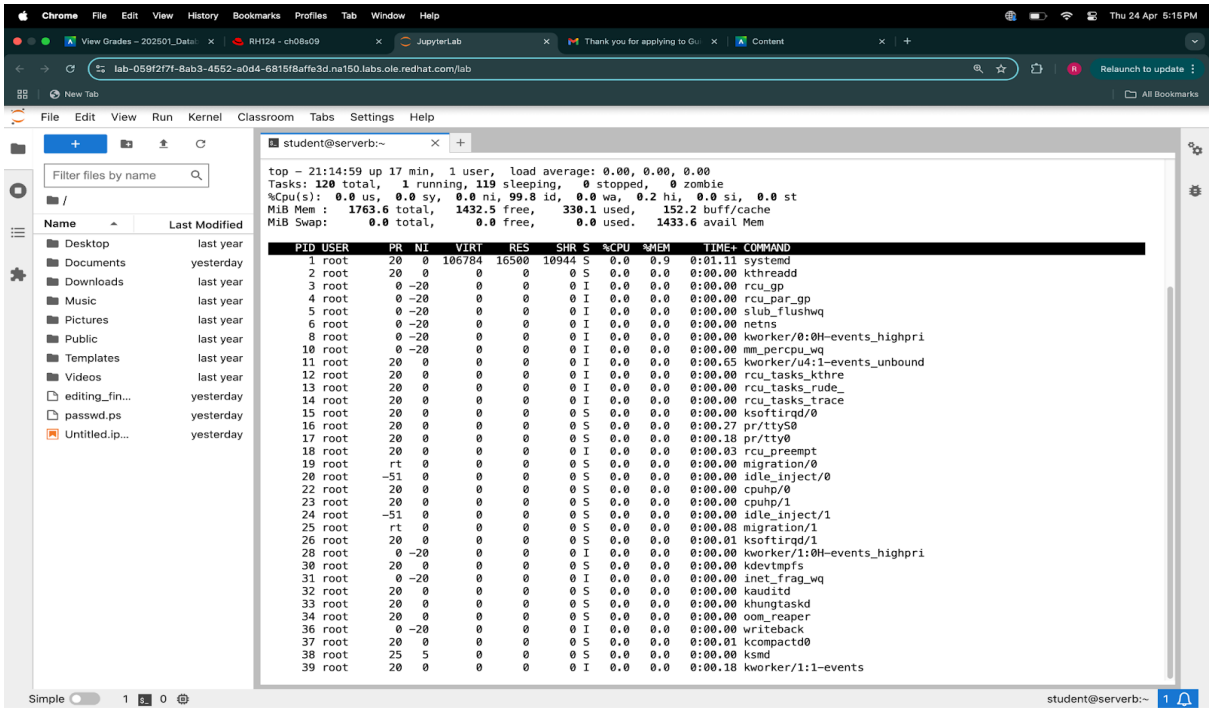
- Configuring Default Permissions with umask:
  The umask value determines the default permissions for newly created files and directories. Modifying /etc/login.defs helps ensure secure defaults by preventing unauthorized access from other users.

- Secure File Creation Practices:
  By combining proper group setup, directory permissions, and umask configurations, you can build a secure multi-user file-sharing environment, enforcing principle of least privilege effectively.

Chapter 8: Monitor and Manage Linux Processes

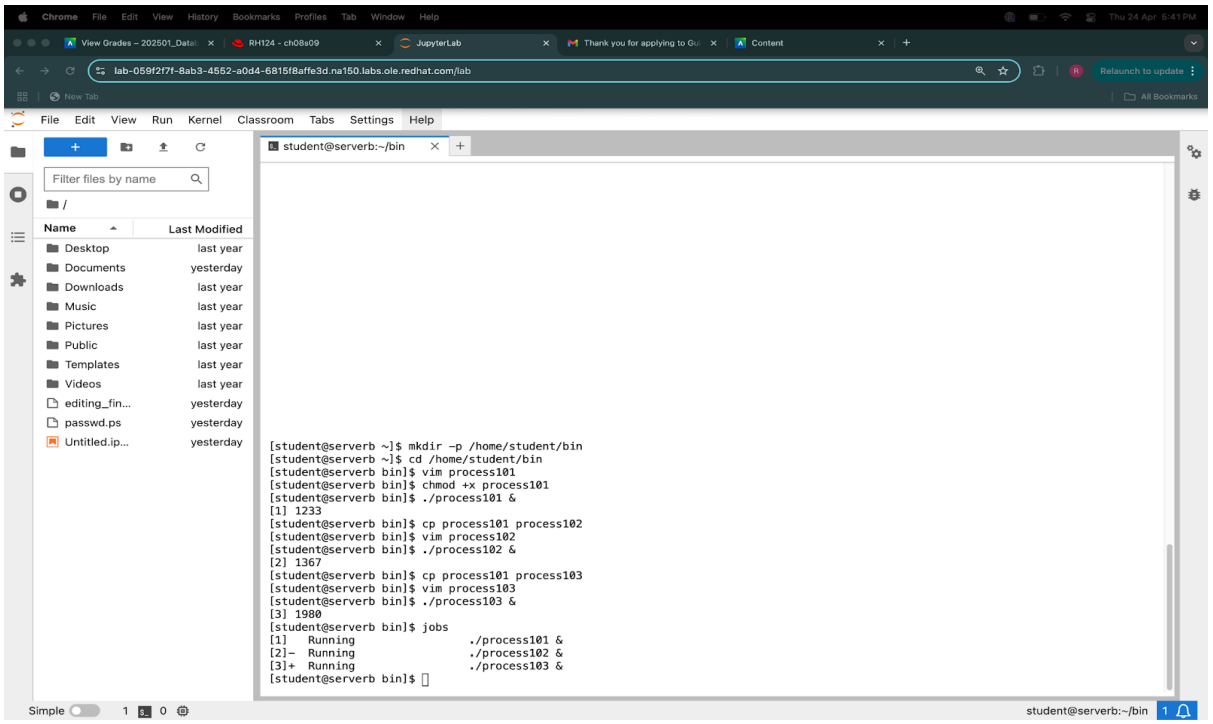
Lab: Monitor and Manage Linux Processes

Fig 8.1 Monitoring System Processes with `top` Command in Linux



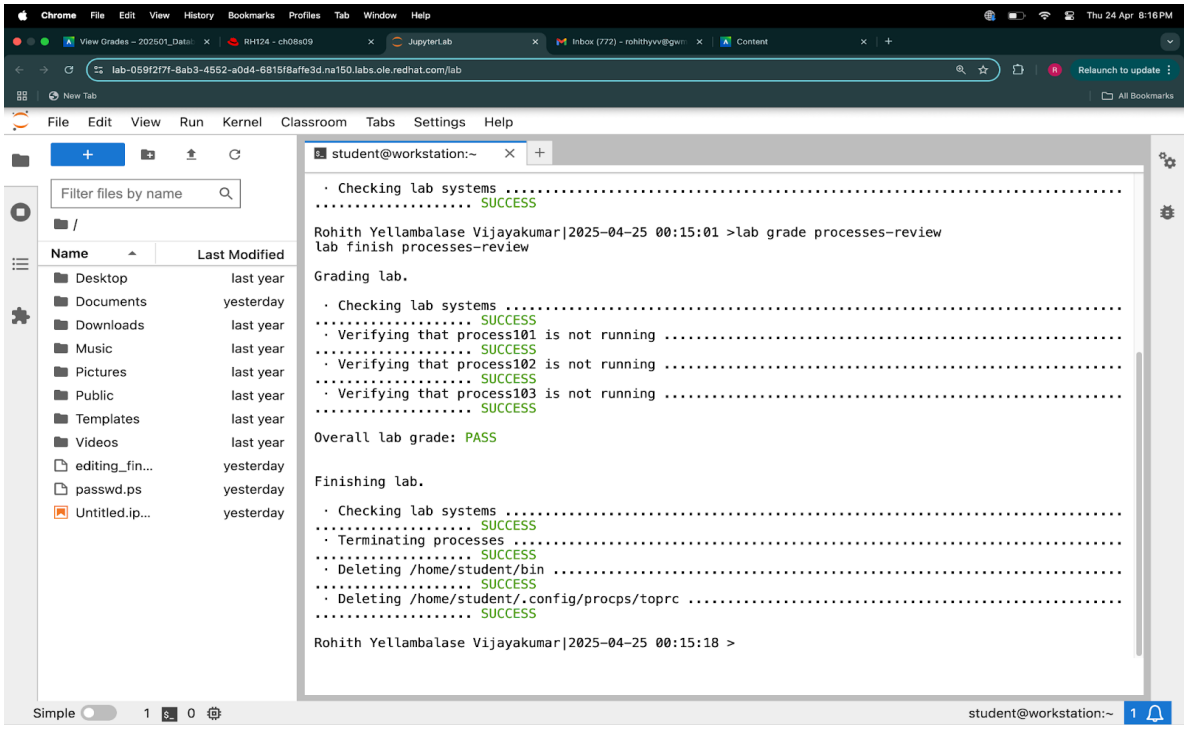
Note: Displays real-time CPU and memory usage; confirms minimal resource consumption with only one active user and low load average.

Fig 8.2 Creating and Running Background Processes in Linux



Note: This shows creation of executable scripts (`process101`, `process102`, `process103`) in the `bin` directory and verifies their execution as background jobs using `&` and `jobs` commands.

Fig 8.3 Creating and Running Background Processes in Linux



Note: In this lab, I created custom scripts (process101, 102, 103) to simulate CPU load and monitored their impact using the top utility. I observed system load, adjusted script behavior, suspended/resumed processes, and terminated them using commands like kill. The exercise demonstrated process control and performance monitoring on Linux.

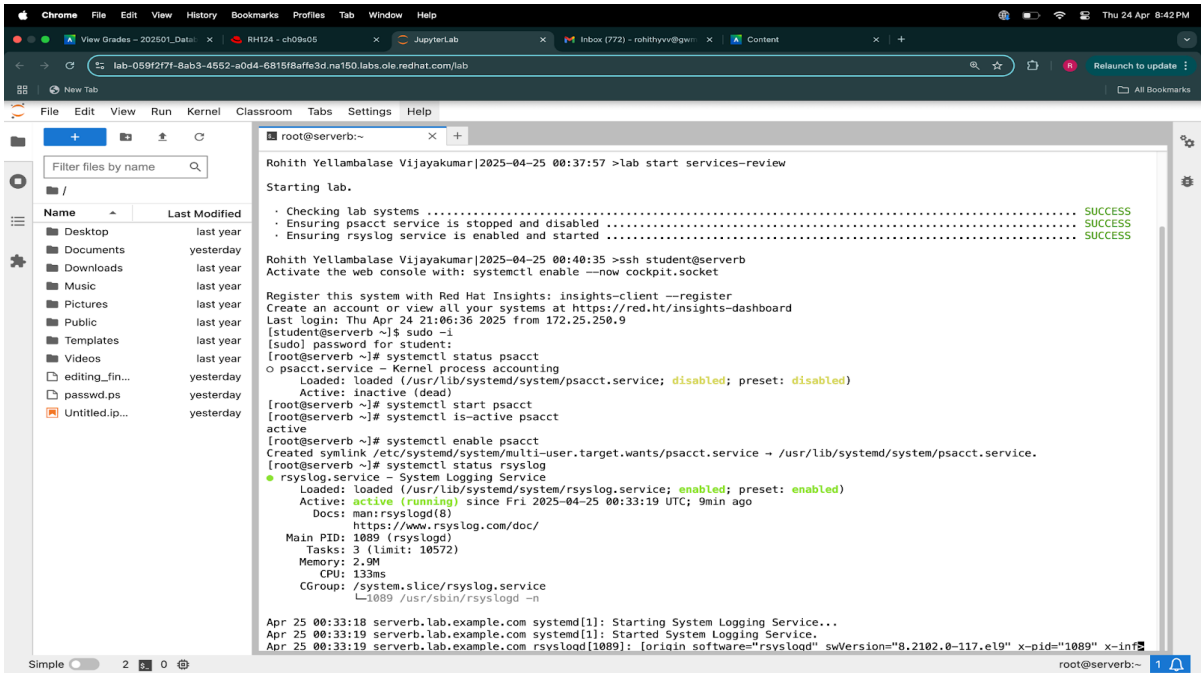
Summary:

- **Viewing and Monitoring Processes**:
  The top utility provides a dynamic, real-time view of running processes, CPU/memory usage, and system load averages. It allows users to monitor performance and identify resource-intensive tasks.
- **Creating and Managing Load**:
  In the lab, I wrote and executed scripts (process101, 102, 103) to generate artificial CPU load. This helped demonstrate how different levels of process activity impact system performance and load metrics.
- **Process Control and Signals**:
  I practiced controlling processes by running them in the background, suspending (kill -STOP) and resuming (kill -CONT), and finally terminating them (kill or kill -9). This reinforces how to handle long-running or misbehaving tasks.
- **Using Job Management Commands**:
  Commands like jobs, bg, and fg were used to switch processes between background and foreground, allowing multitasking and better resource control during terminal sessions.
- **Saving and Customising Top View**:
  I customized the top interface (e.g., turning off bold text), saved the configuration, and learned to toggle between views like CPU, threads, and memory for focused monitoring.

Chapter 9: Control Services and Daemons

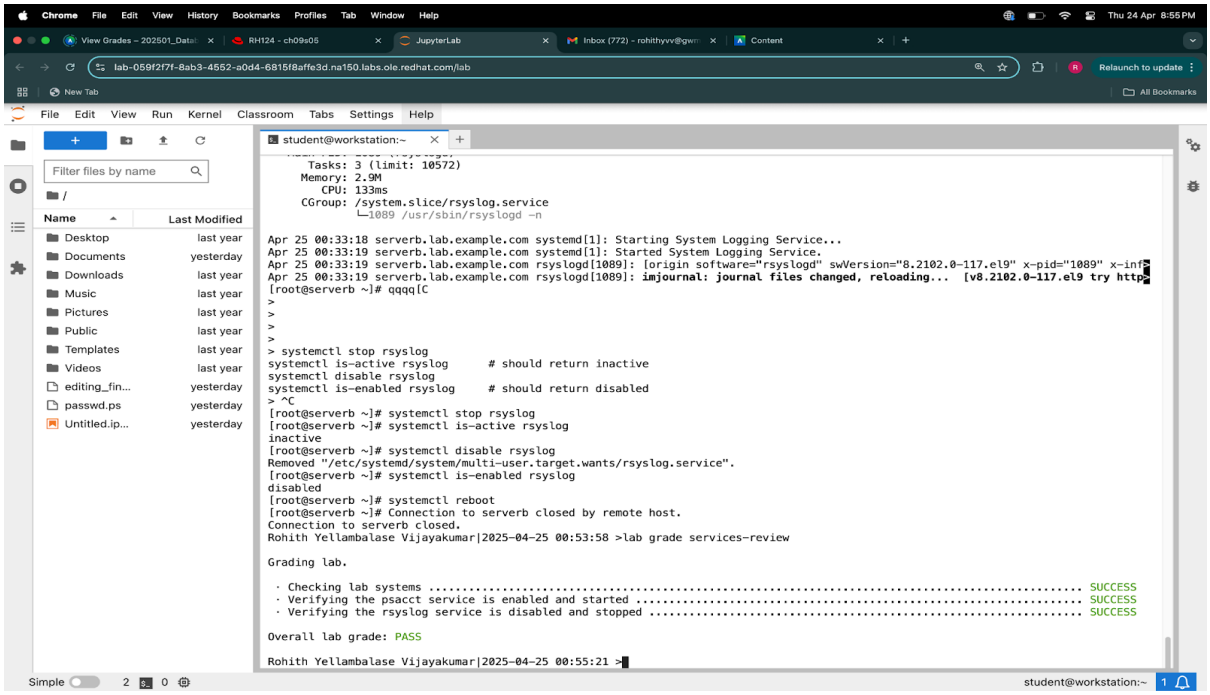
Lab: Control Services and Daemons

Fig 9.1: Managing System Services with systemctl in Red Hat



Note: In this lab, I Successfully verified and controlled psacct and rsyslog services using systemctl; both services managed correctly with all checks passed.

Fig 9.2: Displays the successful validation of enabling the psacct service and disabling the rsyslog service, confirming service states using systemctl.



Note: In this lab, I managed system services using systemctl commands. I enabled and started the psacct service to track user activity and disabled/stopped the rsyslog logging service to prevent it from running at boot. The lab emphasized service status management, persistence, and reboot behavior validation.

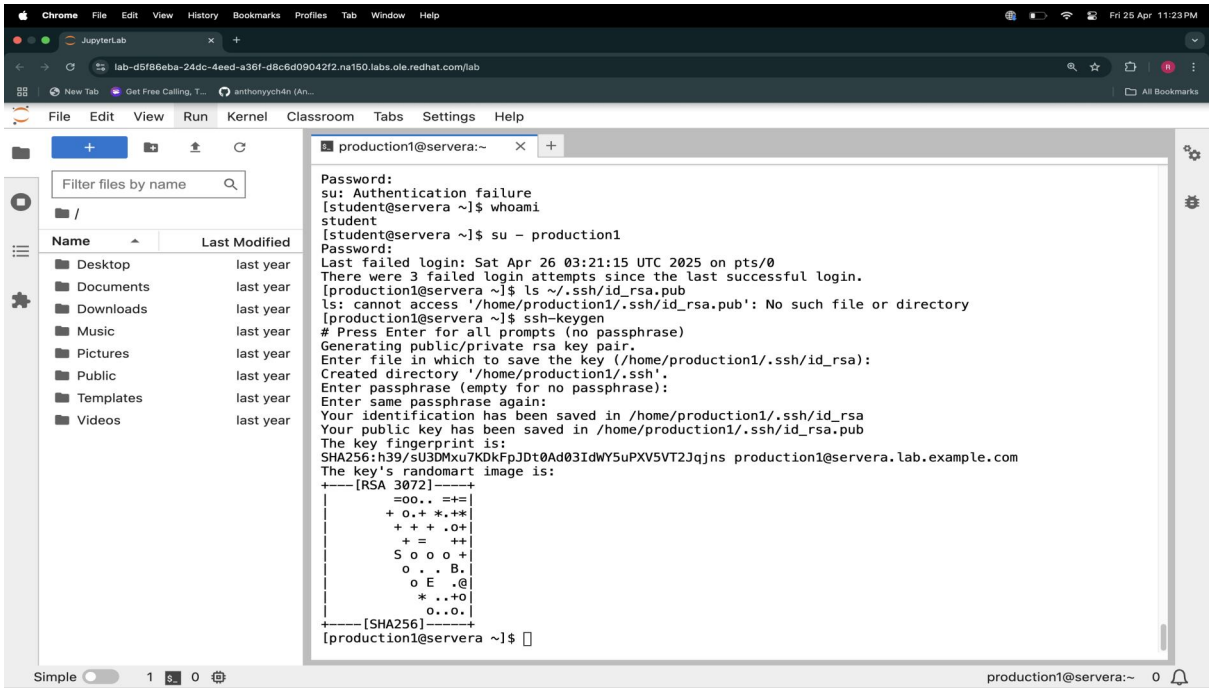
Summary:

- Managing Services with systemctl:
  Use the systemctl command to start, stop, restart, and check the status of system services (daemons) managed by systemd.
- Enabling and Disabling Services at Boot:
  Services can be enabled to start automatically at boot (systemctl enable) or disabled to prevent them from starting (systemctl disable).
- Checking Service Status:
  Commands like systemctl is-active and systemctl is-enabled help verify if a service is currently running and if it's set to start at boot.
- Persistent vs. Temporary Changes:
  Starting/stopping a service only affects the current session, while enabling/disabling affects behavior across reboots.
- Practical Use Case (Lab):
  In the lab, I enabled and started the psacct service for process accounting and disabled the rsyslog service to prevent it from running, verifying both changes after a reboot.

Chapter 10: Configure and Secure SSH

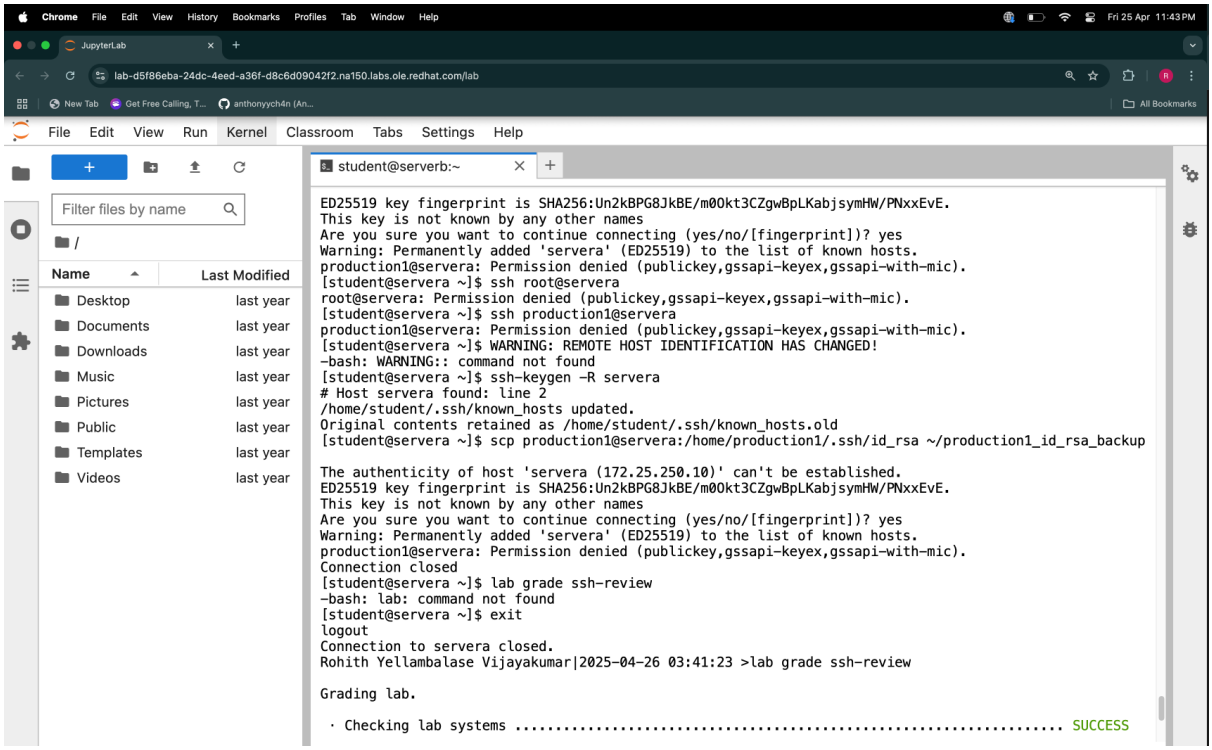
Lab: Configure and Secure SSH

Fig 10.1: Configuring Static IP with nmcli and Verifying Network in Linux.



Note: Successfully created and modified a network connection using nmcli, updated /etc/hosts, rebooted system, and verified connectivity with ping.

Fig 10.2: SSH Review Lab - Initial Grade Attempt

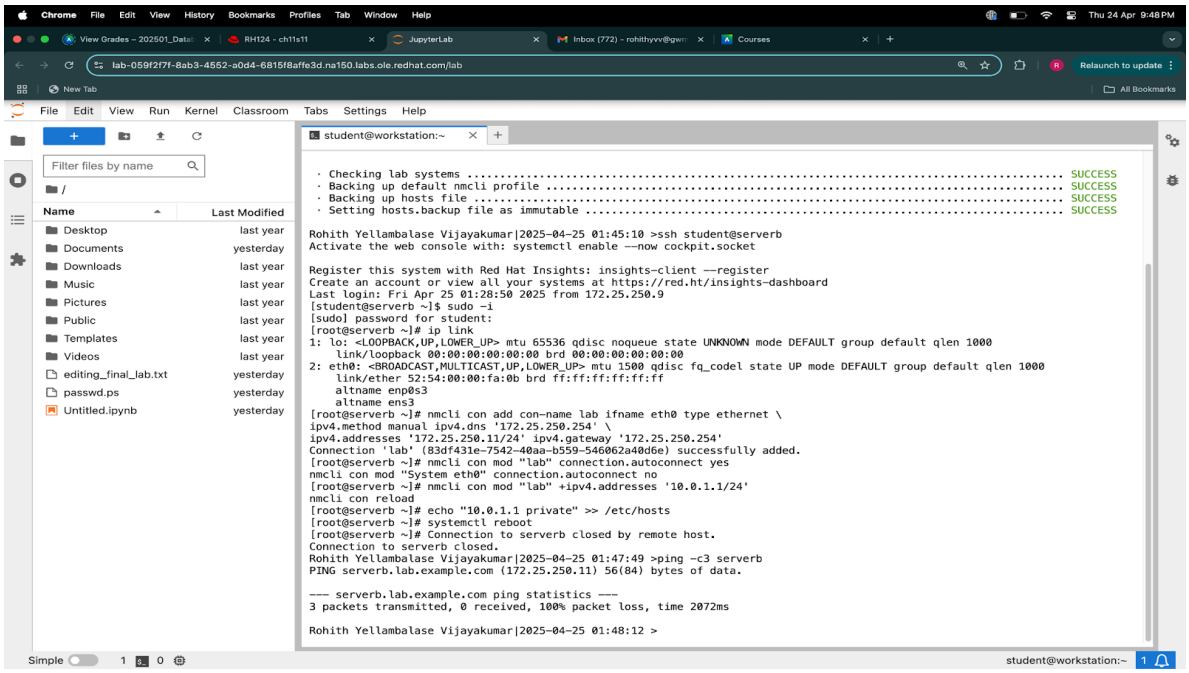


Note: Lab executed on servera instead of serverb, causing failure in key-based authentication and SSH configuration validation during grading.

Chapter 11: Manage Networking

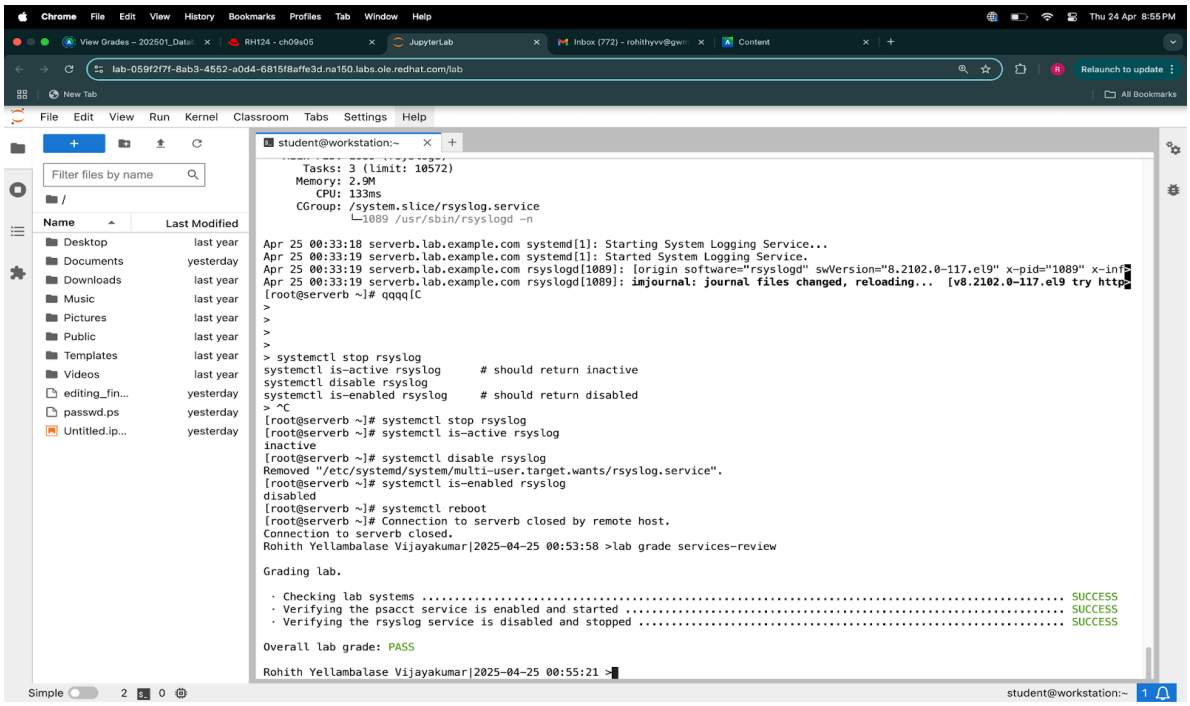
Lab: Manage Networking

Fig 11.1: Configuring Static IP with `nmcli` and Verifying Network in Linux.



Note: Successfully created and modified a network connection using nmcli, updated /etc/hosts, rebooted system, and verified connectivity with ping.

Fig 11.2: shows the successful configuration of a static network setup using nmcli, including dual IPv4 addresses and host aliasing, with a final lab grade of "PASS."



Note: In this lab, I used nmcli to create a new static network profile with two IPv4 addresses on the same interface. I configured DNS, gateway, and /etc/hosts for hostname resolution. After reloading and rebooting the system, I verified connectivity using ping, ensuring all configurations were persistent and correct.

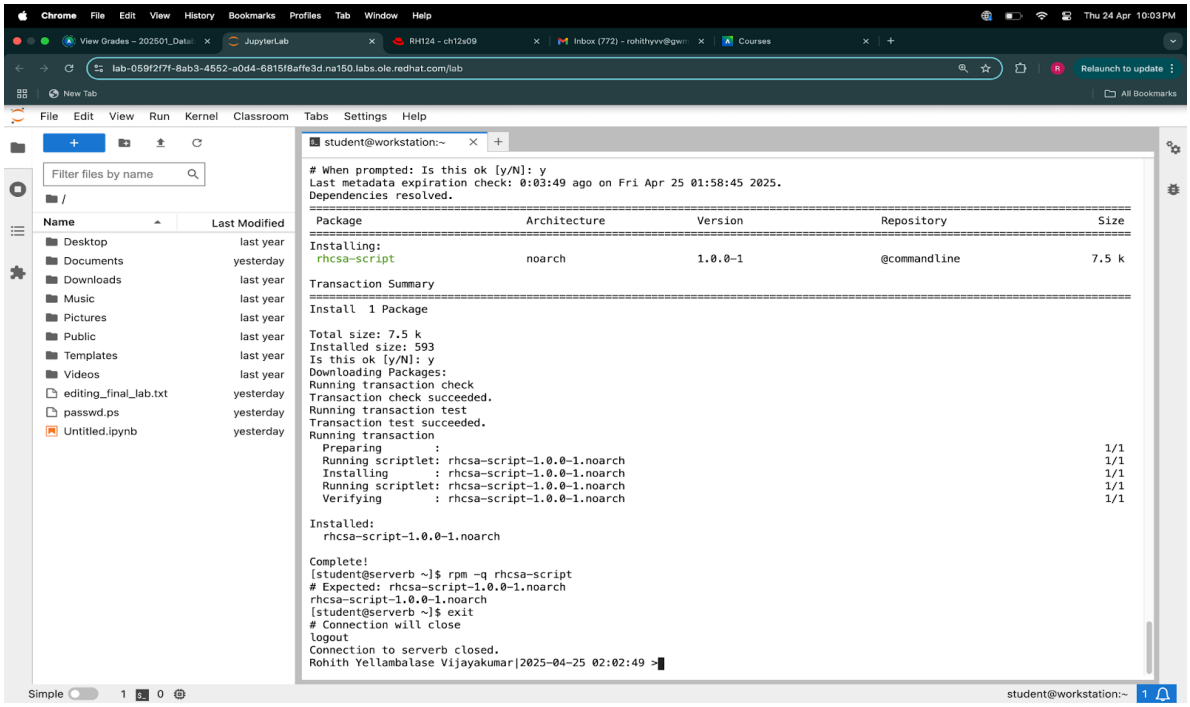
Summary:

- Network Interfaces and Connection Profiles:
  Red Hat systems use NetworkManager to manage network interfaces. Each interface is associated with a connection profile that defines settings like IP addresses, DNS, gateway, and auto-connect behavior.
- Using nmcli for Configuration:
  The nmcli command-line tool allows you to create, modify, activate, and delete network connection profiles. It's essential for scripting and managing networking on headless or remote systems.
- Static IP Configuration:
  You can manually assign one or more static IPv4 addresses to an interface using nmcli, along with setting DNS and gateway information for precise network control.
- Hostname and Hosts File Management:
  The /etc/hosts file can be used to map IP addresses to hostnames locally, enabling name resolution even without DNS.
- Lab Practice:
  In the lab, I created a new connection profile with two static IPs, disabled auto-connect on the default profile, and verified connectivity using ping, reinforcing hands-on network setup and troubleshooting.

Chapter 12: Install and Update Software Packages

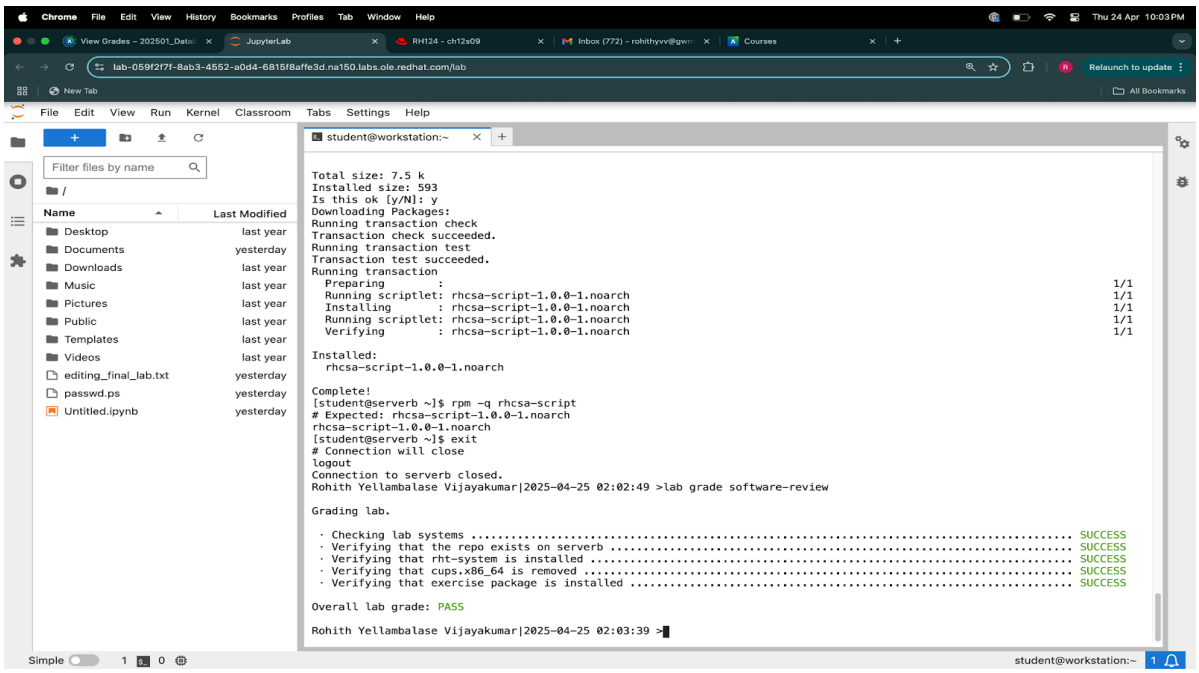
Lab: Install and Update Software Packages

Fig 12.1: Installing and Verifying rhcsa-script Package via RPM



Note: In this lab, I Successfully installed rhcsa-script using RPM, verified package installation with rpm -q, and cleanly exited the server session.

Fig 12.2: shows the successful installation of the rhcsa-script RPM package, removal of the cups package, and confirmation that all required software tasks were completed with a final lab grade of "PASS."



Note: In this lab, I configured a custom software repository, installed the rht-system package, and removed the cups printing service for security compliance. I also installed a local .rpm file (rhcsa-script) using dnf, and verified each step using package management tools like dnf and rpm.

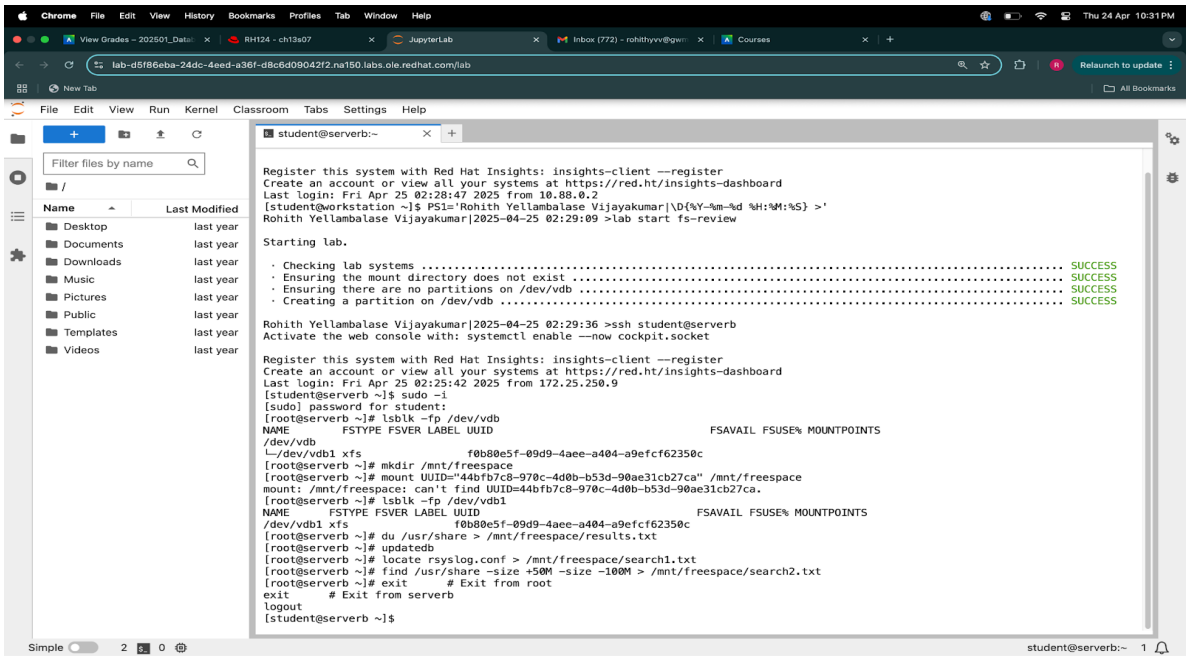
Summary:

- Managing Repositories:
  Repositories are configured using .repo files in /etc/yum.repos.d/. Each repo contains settings like baseurl, enabled, and gpgcheck, which define where and how packages are retrieved.
- Installing and Updating Packages:
  The dnf package manager is used to install, update, or remove software. You can use dnf install <package>, dnf remove, and dnf update to manage software efficiently.
- Verifying and Querying Packages:
  The rpm command lets you query installed packages (rpm -q), check .rpm file details before installation (rpm -qpi), and verify installations.
- Lab Practice:
  In the lab, I created a new repository, installed the rht-system and rhcsa-script packages, and removed the cups package to block printing. This reinforced hands-on skills in RPM and DNF-based package management.

Chapter 13: Access Linux File Systems

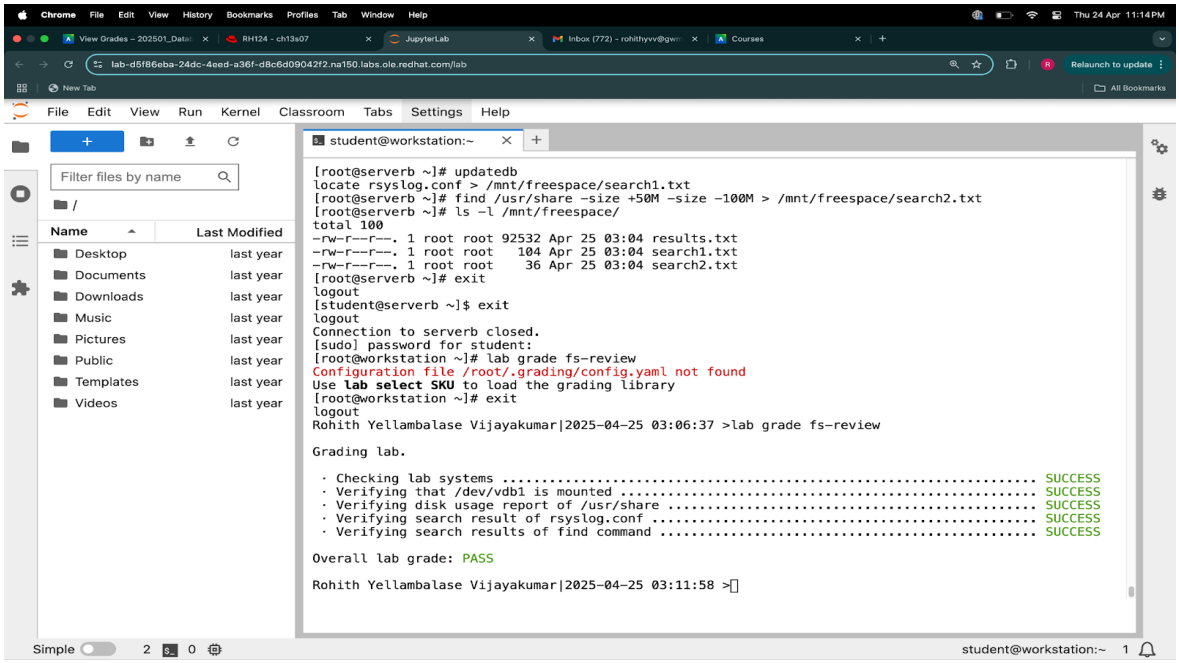
Lab: Access Linux File Systems

Fig 13.1: UUID Mount Fix and Search Operations in Mounted Filesystem



Note: Correct UUID used to mount /dev/vdb1, followed by successful storage usage checks, file copy, and search command execution.

Fig 13.2: shows the successful mounting of the /dev/vdb1 partition using its UUID, along with file search and disk usage tasks completed under /mnt/freespace, resulting in a final lab grade of "PASS."



Note: In this lab, I identified the UUID of a disk partition and mounted it to /mnt/freespace. I generated a disk usage report, used locate to find configuration files, and applied find to locate files by size. This lab reinforced managing mounts and searching content within Linux file systems.

Summary:

- Understanding File System Mounting:
  Devices like partitions or external drives must be mounted to a directory (mount point) before accessing their contents. This can be done using the mount command with device paths or UUIDs.
- Using UUIDs for Mounting:
  Mounting file systems by UUID ensures reliability across reboots and device name changes. UUIDs can be retrieved using lsblk -fp or blkid.
- Temporary vs. Persistent Mounts:
  Mounting via mount is temporary (lost on reboot). For persistent mounts, you modify /etc/fstab with the device's UUID, mount point, and file system type.
- File Searching Utilities:
  The locate command quickly finds files by name using an indexed database (updated with updatedb). The find command allows deeper, real-time searches based on size, type, or other criteria.
- Lab Practice:
  In the lab, I mounted a device using its UUID, generated disk usage reports with du, located files with locate and find, and redirected outputs to files for reporting and validation.

Chapter 14: Analyze Servers and Get Support

Summary:

- Collecting System Information:
  Commands like uname, hostnamectl, and lsb_release provide important system details such as kernel version, architecture, and OS release—useful for troubleshooting and support.
- Analyzing Logs:
  The journalctl command lets you view logs managed by systemd. You can filter logs by boot, service, priority, or time to diagnose issues.
- Viewing Hardware and Device Info:
  Tools like lscpu, lsblk, lsusb, and lspci help inspect CPU, memory, disks, USB devices, and other hardware components.
- Red Hat Support Tools:
  Utilities like sosreport and subscription-manager assist in collecting diagnostic data and managing Red Hat subscriptions for official support.
- Lab Practice:
  In the lab, I gathered system information, reviewed logs, and verified hardware components to simulate real-world troubleshooting and support readiness.