Assignment: Linux Mastery for Junior Developers

Objective:

The goal of this assignment is to provide junior developers with a comprehensive understanding of Linux, focusing on Ubuntu, covering essential Linux commands and concepts. By the end of this assignment, participants should be proficient in navigating the Ubuntu environment, executing common tasks, and leveraging Linux for development purposes.

- Duration: Approximately 40 hours
- Submission Platform: GitHub
- Use googles, ChatGPT online course or any outer source of information inorder to complete your tasks'.

Module 1: Ubuntu Setup and Basic Command

1. Task 1: Ubuntu Installation

- Install Ubuntu (latest LTS version) on a virtual machine or as a dual-boot system.
- Document the installation process and any challenges faced.

2. Task 2: Terminal Basics

- Familiarize yourself with the Ubuntu Terminal.
- Execute basic commands: `ls`, `cd`, `pwd`, `mkdir`, `touch` + 10 more commands of you choice
- Provide screenshots of each command's output.

Module 2: File Operations and Navigation

3. Task 3: File Manipulation

- Create, move, copy, and delete files and directories.
- Use wildcards (`*`, `?`) for efficient file operations.
- Showcase the use of `cp`, `mv`, `rm` commands.

4. Task 4: Working with Text Files

- Use `cat`, `nano`, or `vim` to view and edit text files.
- Demonstrate appending and replacing content in a file.

Module 3: User Management and Permissions

5. Task 5: User Accounts**

- Create a new user account and set a password.
- Switch between users using `su` and `sudo`.
- Provide screenshots of user creation and switching.

6. Task 6: File Permissions

- Understand and modify file permissions using `chmod`.

- Explore the concepts of user, group, and others.

Module 4: Process Management

7. Task 7: Process Monitoring

- Use commands like `ps`, `top`, and `htop` to monitor system processes.
- Identify resource-intensive processes and terminate them.

8. Task 8: Background and Foreground Processes

- Run processes in the background and bring them to the foreground.
- Utilize `bg`, `fg`, and `jobs` commands.

Module 5: Package Management

9. Task 9: Package Installation**

- Use `apt` to install a development-related package.
- Document the process and verify the installation.

10. Task 10: System Updates

- Update the package lists and upgrade installed packages.

- Showcase the use of `apt-get` commands for system updates.

Module 6: Basic Shell Scripting

11. Task 11: Write a Shell Script

- Create a shell script that automates a repetitive task (e.g., file backup).

- Include variables, loops, and conditional statements in the script.

12. Task 12: Execute and Document the Shell Script

- Run the script and document the output.

- Include comments explaining each part of the script.

Module 7 – Final exseresise

Exercise: Report System Health Check and Performance Monitoring Bash Script

Objective: Write a Bash script that performs a system health check and performance monitoring.

Task Description:

System Health Check:

- Check for essential services (SSH, HTTP, HTTPS, and MySQL database services) and report their status (Running or Down).
- Monitor disk usage, identifying partitions with usage over a set threshold (e.g., 80%).
- Check for read-only file systems.
- Monitor CPU and memory usage for each running process, flagging values exceeding normal operating parameters (Using >50% CPU and more than 2GB RAM).
- Analyze system logs (e.g., /var/log/syslog, /var/log/messages) for signs of errors or warnings.

Security Check:

- List all current user failed SSH login sessions and check for any unauthorized access attempts (Output: username used, count login attempts, first attempts date and last attempt date)
- Scan for open ports and report any unexpected open ports.

Additional Requirements:

Format the output in a clear, readable manner.

- Include timestamped logs of each health check.
- Optionally, email the report to a specified system administrator.
- Error Handling and Logging:
- Implement robust error handling within the script to manage unexpected scenarios.
- Log each action performed by the script for later analysis.
- Ensure the script runs efficiently with minimal system impact.
- Optimize the script to run periodically (e.g., as a cron job) or on-demand.

Submission

13. Task 13: Project Documentation

Create a GitHub repository for the assignment.

Submit a comprehensive README.md documenting each task.

Include a summary of challenges faced and lessons learned.