Conceptual Question (1)

1. Explain how **Computer Networking** (specifically the client-server model), the **Linux operating system**, and **Git version control** act as the three foundational pillars for a modern DevOps environment. Describe a simple workflow where a developer's code on their local machine ends up on a remote Linux server, highlighting the role of each component.

Practical, Scenario-Based Questions (10)

1. **Linux File Permissions:** You've deployed a new web application on a Linux server. The application needs to write log files to the /var/log/webapp/ directory, but it's failing with "Permission Denied" errors. The application runs as the webapp user, which is part of the webapp_group. What sequence of **Linux commands** would you use to ensure the webapp user can create and write files in that directory, without giving unnecessary permissions to other users on the system?
2. **Git Branching Strategy:** You're tasked with adding a new login feature to a project. The main branch is called main. What **Git commands** would you use to create a new branch named feature/user-login, switch to it, create a new file called login.py, add some placeholder text to it, and commit the change with the message "feat: Add initial login module"?
3. **Network Troubleshooting:** A developer tells you they cannot connect to the company's private Git server at git.internal.corp. From their Linux-based machine, what are the first three commands you would use to diagnose whether the issue is with DNS resolution, network connectivity, or a firewall block? Explain what each command tells you.
4. **Linux Process Management:** Your team's application server is running slowly. You ssh into the machine to investigate. How would you **list all running processes** and identify the top 5 processes consuming the most CPU? Once you've identified a suspicious process with Process ID (PID) 9876, how would you terminate it gracefully?
5. **Git - Undoing Changes:** You just made a commit, but you realize you forgot to add a crucial file to it. You don't want to create a new, separate commit just for this one file. What **Git commands** would you use to add the forgotten file to the *most recent commit*?
6. **Simple Bash Scripting:** You need to automate the process of backing up a project directory. Write a simple **Bash script** that copies all contents from /home/dev/project-alpha to /mnt/backups/project-alpha-backup-[TIMESTAMP]. The [TIMESTAMP] should be the current date in YYYY-MM-DD format (e.g., 2025-09-24).
7. **Git - Resolving Merge Conflicts:** Your feature/user-login branch is ready. Meanwhile, another developer merged a change into the main branch that modified the same

README.md file you also edited. When you try to merge main into your feature branch to stay updated, you get a merge conflict. Describe the **steps and commands** you would take to view the conflict, fix it by choosing your changes, and complete the merge.

8. **Linux User Management:** A new developer, Alice, has joined the team. Create a new user account for her with the username alice. Ensure she has a home directory created for her and that her default shell is bash. Finally, add her to the existing developers group. What commands are needed to accomplish this?
9. **Checking Network Services:** You need to verify that your web server is running and listening for connections on port 443 (HTTPS). Which **Linux command** would you use to see a list of all active listening network sockets and filter it to check specifically for port 443?
10. **Git History Inspection:** Your manager wants to know what changes were made to the main branch in the last week and who made them. How would you use **git log** to display a concise, one-line summary for each commit from the last 7 days, showing the commit hash, author, and commit message?

Code Review / Debugging Question (1)

1. A junior engineer wrote the following Bash script to clean up old log files (.log files older than 14 days) from a specific directory. However, the script is failing with an error, and even if it didn't fail, it has a logical bug that could cause it to delete files from the wrong place.

   The Script (cleanup.sh):

   Bash

   ```
   #!/bin/bash

   LOG_DIR="/var/log/old application logs/"
   DAYS_TO_KEEP=14

   echo "Changing to log directory..."
   cd $LOG_DIR

   echo "Deleting logs older than $DAYS_TO_KEEP days."
   find . -name "*.log" -mtime +$DAYS_TO_KEEP -exec rm {} \;

   echo "Cleanup complete."
   ```

   Identify at least two distinct bugs in this script. For each bug, explain *why* it's a problem and provide the corrected code.

Take your time to work through these. I'm looking forward to reviewing your solutions! Good luck! 🚀