Extensive List of Linux Commands for General, DevOps, and Server Management

Basic Linux Commands

1. `ls`: Lists files and directories in the current directory. Use options like `-l` for long format or `-a` to show hidden files.
2. `cd`: Changes the current directory. Use `cd <directory>` to navigate to a specific directory or `cd ..` to go up one level.
3. `pwd`: Prints the current working directory, showing the path of the current directory.
4. `mkdir`: Creates a new directory. Use `mkdir <directory>` to create a new directory with the specified name.
5. `rm`: Removes files and directories. Use `rm <file>` to remove a file or `rm -r <directory>` to remove a directory and its contents recursively.
6. `cp`: Copies files and directories. Use `cp <source> <destination>` to copy a file or directory to a specified destination.
7. `mv`: Moves or renames files and directories. Use `mv <source> <destination>` to move a file or directory to a specified destination or rename a file/directory.
8. `cat`: Displays the contents of a file. Use `cat <file>` to print the contents of a file in the terminal.
9. `grep`: Searches for a pattern in files. Use `grep <pattern> <file>` to search for a specific pattern within a file.
10. `chmod`: Changes permissions of files and directories. Use `chmod <permissions> <file>` to modify the permissions of a file or directory.
11. `chown`: Changes ownership of files and directories. Use `chown <user>:<group> <file>` to change the owner and group of a file or directory.
12. `sudo`: Executes a command with root/administrator privileges. Use `sudo <command>` to run a command as a superuser.
13. `apt-get` (or `apt`): Package manager for Debian-based systems. Use `apt-get install <package>` to install a package or `apt-get remove <package>` to uninstall a package.
14. `yum`: Package manager for Red Hat-based systems. Use `yum install <package>` to install a package or `yum remove <package>` to uninstall a package.
15. `ssh`: Connects to a remote server using Secure Shell. Use `ssh <username>@<hostname>` to establish a secure connection to a remote server.
16. `scp`: Securely copies files between local and remote machines. Use `scp <source> <destination>` to copy files between the local machine and a remote server.

17. `tar`: Archives files into a single file. Use `tar <options> <archive_name.tar> <files>` to create a tar archive.
18. `gzip`: Compresses files. Use `gzip <file>` to compress a file. The compressed file will have a .gz extension.
19. `wget`: Downloads files from the web. Use `wget <URL>` to download a file from a specified URL.
20. `man`: Displays the manual page of a command. Use `man <command>` to view the manual for a specific command, providing detailed information and usage instructions.

Linux Commands for DevOps and Server Management

In regulated industries, maintaining server infrastructure, ensuring security, and complying with regulatory standards are paramount. Linux commands provide essential tools to achieve these goals. Here are some crucial Linux commands that play a significant role in maintaining server infrastructure and complying with regulations:

1. `iptables`: This command allows administrators to configure firewall rules and network address translation (NAT) to protect the server from unauthorized access and secure network traffic.
    - **Allow incoming SSH connections**: `sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT`
    - **Block incoming connections from specific IP address**: `sudo iptables -A INPUT -s <IP_ADDRESS> -j DROP`
    - **Practical Use Case**: Setting up a firewall to allow specific incoming and outgoing network traffic while blocking unauthorized access.
2. `fail2ban`: It helps prevent brute-force attacks by monitoring log files and dynamically blocking IP addresses that exhibit suspicious behavior.
    - **Display banned IP addresses**: `sudo fail2ban-client status sshd`
    - **Unban an IP address**: `sudo fail2ban-client set sshd unbanip <IP_ADDRESS>`
    - **Practical Use Case**: Protecting SSH servers from brute-force login attempts by dynamically blocking IP addresses after multiple failed login attempts.
3. `auditd`: This command enables administrators to configure system auditing to track and log changes to files, processes, and system configurations. It helps maintain an audit trail for compliance purposes.
    - **Enable auditing for a specific directory**: `sudo auditctl -w /path/to/directory -p rwa -k directory-access`
    - **Search audit logs for specific events**: `sudo ausearch -k directory-access`
    - **Practical Use Case**: Implementing system auditing to track and log changes made to critical files, configurations, and user activities for compliance and security purposes.
4. `chroot`: This command creates a confined environment for running processes, restricting their access to the rest of the system. It helps enhance server security by isolating potentially vulnerable

applications.

   - **Create a chroot environment**: `sudo chroot /path/to/chroot /bin/bash`
   - **Run a command within a chroot environment**: `sudo chroot /path/to/chroot <command>`
   - **Practical Use Case**: Running potentially vulnerable applications or executing untrusted code within a confined environment to isolate them from the rest of the system for enhanced security.

5. `semanage`: It is used to manage SELinux (Security-Enhanced Linux) policies, which provide fine-grained access control and mandatory access controls (MAC) to protect sensitive system resources.
   - **List SELinux policy types**: `semanage ptype -l`
   - **Allow a process to bind to a non-standard port**: `sudo semanage port -a -t <type> -p tcp <port>`
   - **Practical Use Case**: Managing SELinux policies to enforce mandatory access controls and fine-grained permissions on sensitive system resources.
6. `gpg`: The `gpg` command facilitates encryption, decryption, and digital signatures, ensuring data integrity, confidentiality, and authenticity when transmitting sensitive information.
   - **Encrypt a file**: `gpg -e -r <recipient> <filename>`
   - **Verify a signed file**: `gpg --verify <filename>.asc`
   - **Practical Use Case**: Encrypting sensitive files or emails to ensure confidentiality and secure communication between parties.
7. `tcpdump`: It allows capturing and analyzing network traffic in real-time, aiding in troubleshooting, network monitoring, and identifying potential security issues.
   - **Capture packets on a specific network interface**: `sudo tcpdump -i eth0`
   - **Filter captured packets by port**: `sudo tcpdump port 80`
   - **Practical Use Case**: Capturing and analyzing network traffic to troubleshoot network connectivity issues, monitor network activity, or investigate security incidents.
8. `aide`: The Advanced Intrusion Detection Environment (AIDE) command enables administrators to perform file integrity checks, verifying that critical system files have not been tampered with.
   - **Initialize AIDE database**: `sudo aideinit`
   - **Check file integrity using AIDE**: `sudo aide --check`
   - **Practical Use Case**: Verifying the integrity of critical system files and configurations to detect unauthorized modifications or potential security breaches.
9. `logrotate`: This command automates log file management by compressing, rotating, and archiving logs. It helps ensure log retention and compliance with regulatory requirements.
   - **Manually rotate log files**: `sudo logrotate -f /etc/logrotate.conf`
   - **Display status of logrotate**: `sudo logrotate -d /etc/logrotate.conf`

- **Practical Use Case**: Managing and rotating log files to control disk space usage, maintain log history for compliance purposes, and facilitate log analysis.

10. `openssl`: It provides a versatile set of cryptographic functions, enabling administrators to generate and manage SSL/TLS certificates, encrypt data, and perform various cryptographic operations.
    - **Generate a new RSA key pair**: `openssl genpkey -algorithm RSA -out private.key`
    - **Create a self-signed SSL certificate**: `openssl req -x509 -newkey rsa:4096 -keyout private.key -out certificate.crt -days 365`
    - **Practical Use Case**: Generating SSL/TLS certificates, encrypting sensitive data, or implementing secure communication channels (e.g., HTTPS) to protect data in transit.

By utilizing these Linux commands, server administrators in regulated industries can maintain infrastructure integrity, bolster security measures, and meet compliance requirements effectively. These commands provide essential functionality to enhance server management and security practices in regulated environments.