"Hello, my name is Abood, and I am a Linux system administrator. With 7 years of experience in managing Linux environments, I am responsible for taking care of the operation, maintenance, and security of Linux-based systems. We have couple of distribution of Linux, primarily its RedHat and CentOS running on VMware and OVM. So, on a day to day, I handle system installations and configurations, ensuring that servers are properly set up and optimized for performance. I manage user accounts, permissions, and access control to maintain a secure environment. Regularly, I perform system updates, patch management, and vulnerability assessments to protect against potential security threats. So basically we follow a 30 day patch cycle, we use RedHat satellite as a patch management solution, so every 1st of the month we pull the latest updates and send out the notifications to the different business units and get approvals, create change tickets and perform the updates.

In addition, I monitor system performance and troubleshoot any issues that arise, such as network connectivity problems, hardware failures, or software glitches. So we have monitoring system in place that sends out the alerts based on the threshold that is set if the cpu or memory is high or any disk space issues etc.

And for the automation I do utilize shell scripts and Ansible playbooks, I have created Ansible playbooks for multiple day to day tasks to reduce human errors and to save time,

Examples: for the post build tasks after the server is provisioned, to install and configure different applications like Netbackup, SCOM/Splunk forwarders, Antivirus, these are the core products that we usually install but depending upon the application needs we do install and configure apache, nginx, mysql, php etc. these products we don’t include in our golden template that we use to deploy the servers.

**Kickstart installation:**

**Purpose:** The purpose of Kickstart installations is to automate and streamline the installation process for Linux systems, especially in large-scale deployments. It allows you to quickly deploy multiple systems with consistent configurations and reduces the need for manual intervention during installation.

1. So the very first step is to create a kickstart configuration file, either you can create a new one or use a existing one from any of the server which is already deployed. It will be under root directory called **anaconda-ks.cfg**

#version=RHEL7

install

keyboard 'us'

lang en\_US.UTF-8

network --bootproto=dhcp --device=eth0

timezone America/New\_York

rootpw --plaintext your\_root\_password

firewall --disabled

authconfig --enableshadow --passalgo=sha512

selinux --disabled

# Disk partitioning

clearpart --all --initlabel

autopart

# Packages and software selection

%packages

@base

@server

@network-tools

%end

1. **Place the Kickstart file**: Place the Kickstart file on an accessible server or the installation media. You can use FTP, HTTP, or NFS to make the Kickstart file available to the installation process. For example, if you're using HTTP, copy the Kickstart file to the web server's document root directory.
2. **Start the installation**: Boot the server using the installation media (such as a DVD or USB) or network boot option. During the boot process, you'll need to provide the necessary boot options to specify the Kickstart file location.

For example, if you're using network boot, interrupt the boot process and enter a command similar to the following:

Replace **http://server/kickstart.cfg** with the actual location of your Kickstart file. Or ks=nfs:IP:/Share/ks.cfg ksdevice-eth0 if using NFS

1. **Initiate the installation**: After specifying the Kickstart file location, the installation process will begin automatically. The installer will read the Kickstart file and use its configurations to perform the installation.
2. **Post-installation configuration**: Once the installation completes, you may need to perform additional post-installation configurations, such as setting up network services, configuring security settings, or customizing the server according to your needs.

**Auditing:**

1. Collect server information: Gather relevant details about the server, such as its IP address, hostname, operating system version, installed software, and services running on the system.
2. Review system configuration: Examine the server's configuration files to ensure they are secure and properly set up. Check files like **/etc/ssh/sshd\_config**, **/etc/sudoers**, and **/etc/security/limits.conf** for any misconfigurations or unnecessary access privileges.
3. Analyze user accounts and permissions: Review the user accounts on the server, including privileged accounts and service accounts. Make sure user permissions are correctly assigned and there are no unused or suspicious accounts.
4. Check file system permissions: Inspect the permissions of critical files and directories on the server. Verify that sensitive files, such as system binaries and configuration files, are protected from unauthorized modifications.
5. Monitor log files: Examine system log files (**/var/log/messages**, **/var/log/secure**, etc.) for any unusual or suspicious activities. Look for signs of intrusion attempts, failed logins, or any other security-related events.
6. Evaluate software patches and updates: Check if the server is up to date with the latest security patches and software updates. Outdated software may contain vulnerabilities that can be exploited by attackers.
7. Scan for vulnerabilities: Perform vulnerability scanning using tools like OpenVAS, Nessus, or Nmap to identify any known security weaknesses on the server. Address the identified vulnerabilities accordingly.
8. Harden security settings: Implement security best practices by configuring tools like firewalls, intrusion detection systems (IDS), and implementing secure protocols such as SSH. Consider using tools like fail2ban to protect against brute-force attacks.

**Setup auditing with audit.d:**

To set up auditing in Red Hat Linux, you can use the Audit daemon (auditd) and the auditctl command. The auditd service collects and stores information about system events, while the auditctl command allows you to configure the auditing rules. Here's an example of how you can set up auditing for a specific directory:

1. Install the audit package (if not already installed) by running the following command as root:

# yum install audit

1. Start and enable the auditd service:

# systemctl start auditd

# systemctl enable auditd

1. Configure the auditing rules using the auditctl command. In this example, we'll audit access to a specific directory, **/var/log/audit**:

# auditctl -w /var/log/audit -p wa -k audit-access

**-w /var/log/audit**: Specifies the directory to be audited.

* + **-p wa**: Specifies the permissions to be audited. In this case, "wa" stands for write and attribute changes.
  + **-k audit-access**: Assigns a unique key to the rule, which can be used for filtering and searching in the audit logs.

1. Restart the auditd service to apply the new rules:

# systemctl restart auditd

1. Verify that the auditing is set up correctly by generating an event in the audited directory. For example, you can create a file inside **/var/log/audit**:

# touch /var/log/audit/test.txt

1. View the audit logs using the ausearch command:

# ausearch -k audit-access

This command will display the audit events matching the key "audit-access." You should see an entry related to the file creation event.

**Performance tuning:**

1. Changing the priority of the processes with nice and renice
2. Changing kernel parameters for databases.

Huge pages, also known as large pages, are a feature in the Linux operating system that allows applications to use larger memory pages compared to the default page size. The purpose of using huge pages is to improve system performance by reducing the memory management overhead

By default, the Linux kernel uses small memory pages, typically 4 KB in size. However, with huge pages, you can allocate memory in larger chunks, such as 2 MB or 1 GB. This can be beneficial for applications that require large memory regions, such as databases or high-performance computing workloads.

#nano /etc/sysctl.conf

#vm.nr\_hugepages = <num-pages>

1. Apply the Configuration: To apply the changes made to the configuration file, run the following command:

sudo sysctl -p

1. Reserve Huge Pages: Once the configuration is set, you need to reserve the allocated huge pages so that they can be used by applications. This can be done by running the following command:
2. sudo mkdir /mnt/huge
3. sudo mount -t hugetlbfs nodev /mnt/huge

This will create a mount point for huge pages and mount the hugetlbfs filesystem on it.

1. Validate Huge Pages: To confirm that huge pages are successfully allocated and mounted, you can run the following command:

grep Huge /proc/meminfo

1. Kernel Parameters: Adjusting kernel parameters can significantly impact system performance. Parameters like **vm.swappiness**, **vm.dirty\_ratio**, and **net.core.somaxconn** can be modified to optimize memory usage, disk I/O, and network performance, respectively. You can edit the **/etc/sysctl.conf** file to make permanent changes.
2. Network Tuning: Adjusting network parameters can enhance network performance. Tweaking parameters like **net.core.rmem\_max** and **net.core.wmem\_max** can optimize TCP/IP network buffers. You can modify these settings in **/etc/sysctl.conf**.
3. Service Optimization: Fine-tuning services and daemons can improve system responsiveness. Disabling unnecessary services, reducing their resource consumption, or optimizing their configurations (e.g., Apache **httpd**, MySQL, or PostgreSQL) can lead to better performance.

**Performance troubleshooting:**

1. High CPU Usage:
   * Identify the process consuming the most CPU using the **top** or **htop** command.
   * Analyze the process using tools like **strace** or **perf** to trace system calls and identify any inefficiencies.
   * Check system logs for any error messages related to the process.
   * Consider optimizing the code or configuration of the process if possible.
2. Memory Bottlenecks:
   * Monitor memory usage with tools like **top**, **free**, or **vmstat** to identify if the system is experiencing excessive memory usage or swapping.
   * Analyze memory usage by individual processes using tools like **pmap**, **smem**, or **ps**.
   * Check if any processes have memory leaks or are unnecessarily consuming large amounts of memory.
   * Adjust system parameters such as **vm.swappiness** to optimize memory management.
3. Disk I/O Issues:
   * Identify disk I/O bottlenecks using tools like **iotop** or **iostat**.
   * Check for high disk I/O wait times or excessive I/O utilization.
   * Analyze processes and services that heavily utilize disk I/O and optimize them if possible.
   * Consider using tools like **strace** or **lsof** to identify processes performing excessive disk I/O operations.
4. Network Performance:
   * Monitor network traffic using tools like **iftop** or **nload** to identify any network bottlenecks.
   * Check for network errors or dropped packets using tools like **ifconfig** or **netstat**.
   * Analyze network connections and their performance using tools like **tcpdump** or **wireshark**.
   * Ensure that network drivers and firmware are up to date.

**Swap space:**

Swap space in Red Hat Linux is a designated area on the hard disk that is used as virtual memory by the operating system. When the physical RAM (Random Access Memory) becomes full, the Linux kernel moves inactive pages of memory from RAM to the swap space to free up memory for active processes.

As a Linux system administrator, I also manage various services and daemons, including web servers (such as Apache or Nginx), database servers (like MySQL or PostgreSQL), email servers (such as Postfix or Sendmail), and file servers (such as Samba or NFS). I ensure their availability, performance, and reliability, and I'm well-versed in performance tuning, load balancing, and high availability techniques.

Furthermore, I prioritize security measures by implementing firewalls, intrusion detection systems, and access controls. I regularly conduct system audits, log analysis, and implement security best practices to safeguard the infrastructure against unauthorized access and potential vulnerabilities.

Overall, my role as a Linux system administrator encompasses a broad range of tasks, from system setup and maintenance to troubleshooting, scripting, and security. I am dedicated to ensuring a stable, secure, and efficient Linux environment for the organization I work with."