Red Hat System Administration II (RH134) | RHEL9

Quick Guide

Version 24.07

Ahmed Abdelwahed
ahmed@abdelwahed.me
LinkedIn

Understanding the Boot sequence

1. **Power On**:
    - When the computer is powered on, the BIOS (Basic Input/Output System) or UEFI (Unified Extensible Firmware Interface) is loaded from non-volatile memory. It performs the Power-On Self Test (POST) to check hardware components.
2. **Detect Devices:**
    - The BIOS/UEFI detects and initializes the hardware devices connected to the system.
3. **Choose a Boot Device:**
    - The BIOS/UEFI identifies the bootable devices (like hard drives, USB drives, CD/DVD drives) and allows the user to choose one.
4. **Boot Loader (GRUB):**
    - The boot loader, usually GRUB (GRand Unified Bootloader), is loaded. GRUB reads its configuration file /etc/grub2.cfg to determine which kernel to load.
    - GRUB then loads the selected kernel into memory and executes it.
5. **Execute systemd:**
    - The Linux kernel initializes and executes the first user space process, systemd. This process has the PID (Process ID) of 1 and is responsible for managing the system.
6. **Run .target Files:**
    - systemd loads target files, which are a grouping of units (services, mount points, devices, etc.). The default target is usually multi-user.target, which sets up a multi-user environment.
    - Other targets include basic.target, getty.target, and specific services like ssh.service.
7. **Run Startup Scripts:**
    - systemd runs startup scripts located in directories like `/etc/systemd/logind.conf, /etc/profile, ~/.bashrc`, and other initialization files.
    - These scripts set up the user environment and start necessary services.
8. **Users Can Log In:**
    - After all the initialization processes are complete, the system reaches a state where users can log in and start using the system.

GRUB (GRand Unified Bootloader) is a widely used boot loader for Unix-like operating systems, including Linux distributions like Red Hat. It is responsible for loading the operating system kernel into memory during the boot process. GRUB allows users to choose different kernel versions, boot into different operating systems (if available), and configure various boot options.

In Red Hat and other Linux distributions, GRUB version 2 (GRUB 2) is commonly used. It has replaced the older GRUB Legacy (GRUB 0.97). GRUB 2 offers numerous improvements, including better support for modern hardware and enhanced customization options.

Managing GRUB2 Defaults (Customizing the GRUB Configuration File)

The `/etc/default/grub` file contains various settings for configuring the behavior of GRUB 2. You can customize the default timeout for booting, kernel options, and other settings through this file. Here's how

1. Open the GRUB configuration file in a text editor (like vi)
   ```
   vi /etc/default/grub
   ```
2. Adjust the settings as needed. For example, to change the boot timeout, modify the `GRUB_TIMEOUT` value.
3. After making changes, save the file.
4. Regenerate the GRUB configuration file to apply the changes
   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```
5. Reboot your system to see the changes take effect.

It's important to avoid directly editing the `grub.cfg` file, as it can be overwritten by system updates. Always make your customizations in the `/etc/default/grub` file and then regenerate the configuration.

Systemd Target and Runlevels

Systemd is the init system used in many modern Linux distributions. It introduces the concept of "targets" to manage system states. Here's how to work with systemd targets and runlevels

- **Check Current Runlevel** `runlevel`
- **List All Available Targets** `systemctl list-units --type=target --all`
- **List Units Associated with Targets** `systemctl list-units | grep target`
- **View Default Target** `systemctl get-default`
- **Change Default Target** `systemctl set-default multi-user.target`
- **Isolate (Switch to) a Target** `systemctl isolate multi-user.target`
- **Booting with Specific Targets using GRUB2** You can change the boot target in GRUB2 during boot by adding the systemd.unit parameter followed by the desired target after other boot options.
- **Emergency Mode and Rescue Mode** To boot into emergency mode `systemd.unit=emergency.target` To boot into rescue mode `systemd.unit=rescue.target`

Login Shell and Non-login Shell

A login shell and a non-login shell are two types of shell sessions in Linux. Here's the distinction.

- **Login Shell**
  - User enters credentials before accessing the system.
  - Configuration files read and executed `/etc/profile, ~/.bash_profile, ~/.bashrc.`
- **Non-login Shell**
  - Opened after user has logged in.
  - Configuration files read and executed `~/.bashrc, /etc/bashrc.`

Resetting a Lost Root Password

If you've lost the root password

1. **Edit Boot Entry**
   - At the boot selection menu, press 'e' to edit the boot process.
   - Edit the Linux entry and add `init=/bin/bash`.
   - Press `ctrl+x` to continue booting with the modified entry.
2. **Remount Root Filesystem**
   ```
   mount -o remount,rw /
   ```
3. **Change Root Password**
   ```
   passwd
   ```
4. **SElinux Relabeling (Optional)** To perform SELinux relabeling after changing the password
   ```
   touch /.autorelabel
   exec /usr/lib/systemd/system
   ```

Set GRUB2 Password

Adding a password to GRUB2 enhances system security by preventing unauthorized access to bootloader options

1. **Set GRUB2 Password**
   ```
   grub2-setpassword
   ```
2. **View Password Configuration**
   ```
   cat /boot/grub2/user.cfg
   ```
3. **Update GRUB2 Configuration**
   ```
   grub2-mkconfig -o /boot/grub2/grub.cfg
   ```

4. **Testing the GRUB2 Password**
   - Reboot the system and try to break GRUB by pressing 'e'.
   - If you need to remove the GRUB password protection
     ```
     rm /boot/grub2/user.cfg
     ```

Troubleshooting Using Emergency Mode

Emergency mode in RHEL and other Linux distributions provides a minimal environment for troubleshooting severe system issues. This mode is useful when the system is unable to boot normally due to misconfigurations, file system errors, or other critical issues. Here are some effective examples and scenarios where you might use emergency mode to troubleshoot and resolve problems:

1. Repairing Filesystem Corruption

If filesystem corruption prevents a system from booting, emergency mode can be utilized for repair operations.

- **Step 1**: Boot into emergency mode by adding emergency to the kernel command line from the GRUB menu.
- **Step 2**: Once in emergency mode, remount the root filesystem as read-write if necessary:

  ```
  mount -o remount,rw /
  ```
- **Step 3**: Run fsck on your filesystems to check and repair errors. It's important to specify the filesystem type and device, for example:

  ```
  fsck -y /dev/sda1
  ```

  Step 4: Reboot the system after making repairs: `Reboot`

2. Removing Problematic Services or Applications

If a newly installed service or application prevents the system from booting, you can disable or uninstall it in emergency mode.

- **Step 1**: Boot into emergency mode.
- **Step 2**: Identify the service or application causing issues. Suppose it's a systemd service:

  ```
  systemctl disable problematic-service
  ```
- **Step 3**: If the issue is due to a recent package, you can uninstall it using yum or rpm:

  ```
  yum remove problematic-package
  yum history undo transaction-id
  dnf downgrade package-name-version
  ```
- **Step 4**: Reboot to see if the issue is resolved:

  ```
  reboot
  ```

3. Editing Misconfigured System Files

Incorrect settings in critical configuration files like `/etc/fstab,` network configurations, or `/etc/sysctl.conf` can render a system unbootable.

- **Step 1**: Access emergency mode.
- **Step 2**: Open the misconfigured file with a text editor such as `vi`:

  ```
  vi /etc/fstab
  ```
- **Step 3**: Make the necessary corrections.

- **Step 4**: Save the changes and reboot the system to see if the issue is resolved.

4. Reviewing Logs for Errors

Examining logs can provide clues to what is causing boot failures.

- **Step 1**: Boot into emergency mode.
- **Step 2**: Access system logs using journalctl or view specific log files in `/var/log/,` for example:

  ```
  journalctl -b -1
  ```
- **Step 3**: Look for error messages or warnings that occurred during the last boot attempt.
- **Step 4**: Resolve any identified issues based on the error messages.

Boot with Limited Resource

```
linux /vmlinuz-... ro quiet mem=4G maxcpus=1 emergency
```

Shell Scripting

Shell scripting is a method to automate repetitive tasks by creating a series of commands in a script file that can be executed by the shell (command interpreter).

Basics of Writing Shell Scripts

- **Creating a Script**: A shell script typically starts with a `"shebang" (#!)` followed by the path to the interpreter that should execute the script (e.g., `#!/bin/bash` for Bash scripts).
- **Writing Commands:** Scripts can contain any commands you might type at the command line. These are executed in sequence.
- **Making Scripts Executable:** After writing a script, you must make it executable using the `chmod` command, e.g., `chmod +x myscript.sh.`
- **Running a Script**: Execute a script by typing its path in the shell, like ./myscript.sh, assuming you're in the same directory.
- **Profiles and Configuration Files:** Understanding how to set and use environment variables through files like `.bashrc`, `.profile`, or `/etc/profile` for global variables is crucial for scripting. To print all the environment variables along with their values use: `printenv`, `env`, `set`
- **Quoting:** Use quotes to handle strings and variables that include spaces and special characters.
- **Exit Statuses:** Scripts should handle exit statuses of commands `(echo $?)` to check if a command succeeded or failed, where 0 for successful and Non-zero exit statuses indicate different types of errors or statuses

Understanding the set Command in Bash

The set command in Bash is a powerful built-in utility that controls the behavior of the shell itself and is often used to define how scripts handle certain situations like error handling and argument processing.

1. set --: Resetting Positional Parameters

One of the uses of the set command is to reset all positional parameters using set --. This is particularly useful when you need to clear existing positional parameters or set new ones explicitly.

- **Usage**: set -- "value1" "value2"
- **Example**: Resetting parameters for a script.

```
# Before
echo $1 $2  # Outputs current $1 and $2
set -- "Hello" "World"
# After
echo $1 $2  # Outputs: Hello World
```

In this example, the positional parameters $1 and $2 are reset to "Hello" and "World" respectively. It can be beneficial when parameters need to be redefined due to changing input requirements within a script.

2. set -e: Error Handling in Scripts

Another critical use of the set command is set -e, which tells the shell to exit immediately if a command within a script exits with a non-zero status (indicating failure).

- **Usage**: set -e

```
#!/bin/bash
set -e  # Exit on error
echo "Starting the script..."
# If the next command fails, the script will exit immediately.
cp important_file backup_file
echo "Script completed successfully.
```

Install and Configure Nginx Homepage

This script installs Nginx, starts and enables its service, adds firewall rules, and creates a welcome page.

```
# Install Nginx
      yum install -y nginx
# Enable and start Nginx service
      systemctl enable nginx
      systemctl start nginx
# Configure firewall rules
      firewall-cmd --add-port=80/tcp --permanent
      firewall-cmd --add-port=80/udp --permanent
      firewall-cmd --reload
# Create a welcome page
      echo "Welcome to Oman" > /usr/share/nginx/html/index.html
```

System Information Script

This script retrieves various system information and displays it in a formatted manner.

```
#!/bin/sh
clear
echo "------- Script Started ------"
echo
echo "Time now is " `date`
echo
echo "System Running time " `uptime | awk '{print $1}'`
echo
echo "Your User name " `whoami`
echo
echo "Server name is " `hostname`
echo
echo "Server kernel Version " `hostnamectl | grep -i Kernel | awk '{print $3}'`
echo
echo "Server CPU info " `cat /proc/cpuinfo | grep name`
echo
echo "Server memory " `free -h | grep -i mem | awk '{print $2}'`
echo
echo "Hard disk size"
df -h /
echo
echo "Server IP"
ifconfig | grep -w inet | awk '{print $2}'
echo
echo "-------- Script ended -------"
echo
sleep 3
```

Using Simple Shell Variables

```
#!/bin/sh
name=Ahmed
age=40
score=10
echo "Hello $name, Your Age is $age, and your Score is $score"
```

User Info Script (Using System Environment Variables - env)

```
#!/bin/sh
echo "Machine name is $HOSTNAME"
echo "You are working from $(tty)"
echo "Your user name is $LOGNAME"
echo "Your home directory is $HOME"
echo "Your default Shell is $SHELL"
echo "Your current directory is $PWD"
echo "All of this data was collected using $0 script"
```

Scheduling jobs

Using `at` to Schedule One-time Tasks

1. **Ensure at is installed:** `dnf install at`
2. **Start and enable the atd service:**
   ```
   systemctl start atd
   systemctl enable atd
   ```
3. **Schedule a one-time task:** Use the at command to schedule a task. For example, to schedule a script to run at 2:30 PM today:
   ```
   at 4:00 PM
   /path/to/script.sh
   at now + 2 hours
   echo "This is a test message" > /tmp/testfile
   at 3:00 PM tomorrow
   at 10:00 AM next Monday
   at midnight
   at 8:00 AM Jul 15
   ```

Press `Ctrl + D` to save and exit.

4. **View scheduled at tasks:** `atq`
5. **Remove a scheduled at task:** `atrm <job_number>`

Using `crontab` to Schedule Recurring Tasks

- **Understanding crond Service Behavior**
  - The `crond` daemon reads its configuration every minute and schedules jobs for the next minute. Allow at least a 3-minute margin between configuration and execution for optimal results.
- **Installing and Starting cron**
  - Install the `cronie` package `dnf install cronie`
  - Start the crond service `systemctl start crond.service`
- **Viewing Log Messages**
  - Monitor log messages `tail -f /var/log/messages`
- **Editing Cron Jobs**
  - Edit the main cron configuration `vim /etc/crontab` (including user)
    ```
    * * * * * root  rm -rf /tmp/testdir/*
    17 * * * * * ahmed rm -rf /home/ahmed/Desktop/*
    ```
  - Edit user's cron jobs `crontab -e`
- **Cron Job Format**
  - Example of job definition

    * * * * * user-name command to be executed
  - Fields represent `minute`, `hour`, `day of month`, `month`, `day of week`.
- **Common Cron Job Examples**
  - `0,10,20,30,40,50 17-20 15 Jun,Jul,Aug * root /usr/local/bin/my-script.sh` This job runs on minutes 0, 10, 20, 30, 40, and 50 **past the hour**, from 5 PM to 8 PM, on the 15th day of June, July, and August.
  - `0 5,17 * * * bash /cron/batch` This job runs at 5 AM and 5 PM daily.

    - `3 12 * * sun /bin/systemctl stop atd.service` This job runs at 12:03 PM every sunday.
    - `* * * * * bash /cron/batch` This job runs every minute.
    - `*/10 * * * * /scripts/monitor.sh` This job runs every 10 minutes.
    - `0 17 * * sun,fri /script/script.sh` This job runs at 5 PM on Sundays and Fridays.
- **Special Keywords**
    - Use keywords like `@yearly`, `@monthly`, `@weekly`, `@daily`, `@hourly`, `@reboot` for predefined schedules.
- **Managing Cron Jobs**
    - Remove cron job `crontab -r`
    - List user's cron jobs `crontab -l`
- **System-wide Cron Jobs**
    - Use `/etc/cron.d` for system-wide cron job configurations.
    - System-wide scripts run from directories like `/etc/cron.hourly, /etc/cron.daily,` etc.
- **Running Multiple Tasks in One Cron**
    - Use semicolons to run multiple tasks in a single cron entry.
- **User-specific Cron Management**
    - Manage cron jobs for specific users
        - Edit user's cron jobs `crontab -e -u username`
        - List user's cron jobs `crontab -l -u username`
        - Remove user's cron jobs `crontab -r -u username`
- **Directory for System-wide Cron Jobs**
    - `/etc/cron.d` Store system-wide cron job configurations.
    - `/etc/cron.hourly, /etc/cron.daily,` etc. Executed automatically by the cron daemon on an hourly basis.
- **Executing Scripts in a Directory**
    - Test cron job execution for a directory `run-parts --test /etc/cron.hourly`
    - Execute all scripts in a directory `nice run-parts /etc/cron.hourly`

Managing Cron Security

By default, all users can create cron jobs. However, you can enhance cron security using `/etc/cron.allow` and `/etc/cron.deny` files

- If `/etc/cron.allow` exists, only users listed in it can use cron.
- If `/etc/cron.deny` exists, users listed in it are denied access to cron.

Anacron

Anacron is designed to run scheduled jobs even if the system is powered off during the scheduled time. It executes missed jobs once the system is turned on.

- `/etc/anacrontab` **File**
    - This file contains specifications for anacron jobs.
    - To schedule an anacron job, use the following format
      `period delay job-identifier command`

Example

- To schedule an anacron job for a backup script called db-backups.sh
    - Run every 7 days.
    - Start with a delay of 10 minutes after anacron is executed.
    - Use a unique identifier my-backups.
    - Execute the script `/bin/bash /path/to/db-backups.sh.`

Entry in /etc/anacrontab

```
7 10 my-backups /bin/bash /path/to/db-backups.sh
```

Scheduling Tasks with Systemd Timers

Systemd Timers help schedule and automate tasks on Linux systems. They work with systemd services to run tasks at specific times. Here's a simple guide on how to use them:

1. **Unit File Naming Convention**
    - Timer files use the naming convention <name>.timer.
    - The corresponding service files use the same name with a .service extension.
2. **Creating a Timer Unit File**
    - Create a file like mytimer.timer in /etc/systemd/system or /usr/lib/systemd/system.
3. **Defining Timer Settings**
    - In the timer file, define settings like OnBootSec (delay after boot), OnUnitActiveSec (delay after the dependent unit activates), and OnCalendar (specific times).

      ```
      Example:
         [Unit]
         Description=My Timer

         [Timer]
         OnBootSec=5min
         OnUnitActiveSec=1h
         Unit=mytask.service

         [Install]
         WantedBy=timers.target
      ```
4. **Creating a Service Unit File**
    - Create a file like mytask.service that defines the task.
5. **Enabling and Starting Timers**

    ```
    systemctl enable mytimer.timer
    systemctl start mytimer.timer
    ```
6. **Checking Timer Status**

    ```
    systemctl status mytimer.timer
    systemctl status mytask.service
    ```
7. **Disabling and Stopping Timers**

    ```
    systemctl stop mytimer.timer
    systemctl disable mytimer.timer
    ```
8. **Logging and Output**
    - Use journalctl to view logs.

    ```
    journalctl -u mytask.service
    ```

Example: Scheduling a Daily Backup Task

1. **Create Timer Unit File (backup.timer)**

   ```
   [Unit]
   Description=Backup Timer

   [Timer]
   OnCalendar=daily
   Persistent=true

   [Install]
   WantedBy=timers.target
   ```

2. **Create Service Unit File (backup.service)**

   ```
   [Unit]
   Description=Backup Service

   [Service]
   Type=oneshot
   ExecStart=/path/to/backup_script.sh
   ```

3. **Enable and Start the Timer**

   ```
   systemctl enable backup.timer
   systemctl start backup.timer
   ```

4. **Check Status**

   ```
   systemctl status backup.timer
   systemctl status backup.service
   ```

5. **View Logs**

   ```
   journalctl -u backup.service
   ```

Using `OnCalendar` for Scheduling

- **Every Monday at 4 PM**: `OnCalendar=Mon *-*-* 16:00:00`
- **First day of every month at 4 PM**: `OnCalendar=*-*-01 16:00:00`
- **Daily at 4 PM and 8 PM**: `OnCalendar=*-*-* 16:00:00,20:00:00`

Managing Temporary Files

The /tmp directory is for temporary files. To manage these:

1. **Scheduled Deletion with Timer Units**
   ```
   find /tmp -atime +7 -delete
   ```
2. **Using systemd-tmpfiles**
   - Create /etc/tmpfiles.d/cleantmp.conf:
   ```
   D /tmp 1777 root root 10d
   ```
   - Apply the configuration:
   ```
   systemd-tmpfiles --create /etc/tmpfiles.d/cleantmp.conf
   ```
3. **Automated Cleaning with Systemd Timers**
   - Timer Unit File (cleantmp.timer):
   ```
   [Unit]
   Description=Clean /tmp directory every 5 minutes

   [Timer]
   OnCalendar=*00/5

   [Install]
   WantedBy=timers.target
   ```
   - Service Unit File (cleantmp.service):
   ```
   [Unit]
   Description=Clean /tmp directory

   [Service]
   Type=oneshot
   ExecStart=/usr/bin/systemd-tmpfiles --clean /etc/tmpfiles.d/cleantmp.conf
   ```
   - Enable and start the timer:
   ```
   systemctl daemon-reload
   systemctl enable cleantmp.timer
   systemctl start cleantmp.timer
   ```

Controlling Access to Files with ACLs

Access Control Lists (ACLs) provide a more flexible permission mechanism for file systems than the traditional Unix permissions system. ACLs allow you to define permissions for specific users and groups beyond the file's owner, group, and others.

Basic ACL Commands

1. **Set ACLs**:

Use the setfacl command to set ACLs on a file or directory. For example, to give user ahmed read and write permissions on file.txt:

```
setfacl -m u:ahmed:rw file.txt
```

- -m: Modify the ACL.
- u:ahmed:rw: Set read and write permissions for user ahmed.

2. **Get ACLs:**

Use the getfacl command to view the ACLs of a file or directory:

```
getfacl file.txt
```

3. **Remove ACLs:**

To remove a specific ACL entry, use the -x option. For example, to remove the ACL for user ahmed:

```
setfacl -x u:ahmed file.txt
```

To remove all ACLs from a file or directory, use the -b option:

```
setfacl -b file.txt
```

4. **Default ACLs:**

Default ACLs apply to directories and are inherited by all new files and subdirectories created within. To set a default ACL on a directory:

```
setfacl -m d:u:ahmed:rwx /mydir
```

- d:u:ahmed:rwx: Set default read, write, and execute permissions for user ahmed on the directory /mydir.

Example Scenarios

1. **Granting User-Specific Permissions:**

Suppose you have a file report.txt and you want to give user alice read and write access:

```
setfacl -m u:alice:rw report.txt
```

2. **Granting Group-Specific Permissions:**

If you want to give group sales read access to a directory data and all its files:

```
setfacl -R -m g:sales:r data
```

- -R: Apply the ACL recursively to all files and subdirectories.

3. **Setting Default Permissions on a Directory:**

To ensure all new files in the directory projects are writable by the group developers:

```
setfacl -d -m g:developers:rw projects
```

Viewing and Understanding ACL Entries

When you run getfacl on a file or directory, the output might look like this:

```
# file: file.txt
# owner: root
# group: root
user::rw-  # The owner's permissions.
user:ahmed:rw-  # The permissions for user ahmed.
group::r--  # The group's permissions.
mask::rw-  # The effective rights mask, which limits the permissions that can be granted.
other::r--  #The permissions for others.
```

Special Permission Bits in Linux

Special Permissions in Linux

1. **Set User ID (SUID)**:
   - When applied to a file, this permission allows the file to be executed with the permissions of the file owner.
   - Numeric: **4**
   - Symbolic: **s** (on the owner’s execute position)
   - **Numeric**: chmod **4755** /path/to/file
   - **Symbolic**: chmod **u+s** /path/to/file
   - **Finding SUID files**: find / -type f -perm /4000 or find / -type f -perm /u=s
2. **Set Group ID (SGID):**
   - When applied to a file, this permission allows the file to be executed with the permissions of the group owner.
   - When applied to a directory, new files created within the directory inherit the group ID of the directory.
   - Numeric: **2**
   - Symbolic: **s** (on the group's execute position)
   - **Numeric**: chmod **2755** /path/to/file
   - **Symbolic**: chmod **g+s** /path/to/file
   - chmod **2755** /path/to/directory
   - **Finding SGID files**: find / -type f -perm /2000 or find / -type f -perm /g=s
3. **Sticky Bit:**
   - When applied to a directory, it restricts file deletion. Only the file owner, directory owner, or root can delete files within the directory.
   - Numeric: **1**
   - Symbolic: **t** (on the others’ execute position)
   - **Numeric**: chmod **1755** /path/to/directory
   - **Symbolic**: chmod **+t** /path/to/directory
   - **Finding Sticky Bit directories**: find / -type d -perm /1000 or find / -type d -perm /o+t

To view the octal values of permissions, you can use the stat command or the ls command with the -l option. For example

```
stat -c "%a %n" myfile.txt
```

Or using ls

```
ls -l myfile.txt
```

Managing Basic Storage and Partitions

A. Basic Storage Management Commands

1. **Installing Hardware Information Tool**
   - `dnf install hwinfo`
2. **Gathering Disk Information**
   - `hwinfo --disk –short`
   - `hwinfo --memory`
   - `hwinfo --cpu`
   - `hwinfo --short`
3. **Disk Space Analysis**
   - `lsblk -la`
   - `lsblk /dev/sdb1`
   - `df -hT`
   - `du -ah`
   - `du -sh`
   - `mount | column -t` #Show current mounted filesystems
   - `lsof +D /path/to/directory` # List open files and their associated processes.

B. Using Fdisk for Partition Management

fdisk is a traditional partitioning tool, primarily used with MBR (Master Boot Record) supporting up to 2 TB partitions and a maximum of four primary partitions, with one being an extended partition.

1. Creating Partitions

- `fdisk -l /dev/sdb` (List all partitions for a block)
- `fdisk /dev/sdb` (Create partition table with interactive options)
- `partprobe /dev/sdb` (Inform kernel of partition table changes)
- `dd if=/dev/zero of=/dev/sdb count=1 bs=512` (Delete partitioning)

2. Adding File System and Labeling

File System Considerations

- *ext4* Max file size of 2 TB.
- *XFS* Handles larger files, up to 8 EB.
  - `mkfs.xfs -L Data /dev/sdb2` (Create XFS filesystem with label)
  - `xfs_admin -L "mydata" /dev/sdc1` (Label unmounted XFS filesystem)
  - `xfs_admin -lu /dev/sdb1` (Show filesystem label and uuid)
  - `dumpe2fs /dev/sdb1 | less` (Display filesystem metadata)
  - `Xfs_info /dev/sdb2`

3. Mounting File System to Directory

- `mount` or `findmnt` (Show all mounted filesystems)

Configuring Temporary Mounting Points

- **a.** `mkdir -p /data/{sdbdata1,sdbdata2}` (Create directories)
- **b.** `mount /dev/sdb1 /data/sdbdata1`
- **c.** `mount /dev/sdb2 /data/sdbdata2`
- **d.** `mount -a` (Activate fstab mounting points)
- **e.** `findmnt -x` (Check for mount errors)
- **f.** `umount /data/sdbdata1`

Mounting File Systems Using fstab and Systemd in RHEL 9

A. Preparing to Use fstab

1. **Editing fstab File**
   - Open the fstab file `vi /etc/fstab`
   - Use `blkid` to retrieve the UUID or file system label.
2. **Generating and Retrieving UUIDs and Labels**
   - Generate UUID for a specific device `xfs_admin -U generate /dev/sdc1`
   - Retrieve the UUID of the filesystem `ls -l /dev/disk/by-uuid/`
   - Retrieve the Label of the filesystem `ls -l /dev/disk/by-label/`
   - Retrieve the Path of the filesystem `ls -l /dev/disk/by-path/`

B. Mounting File System Using Systemd (RHEL 9)

1. **Creating Mount Unit File**
   - Use `vim` to create/edit a mount unit file `vim /etc/systemd/system/mnt-data1.mount`
   - **Naming Convention** The filename should match the mount point, replacing '/' with '-'. For example
     - Mount point `/data1` ➔ Filename `data1.mount`
     - Mount point `/tmp/data1` ➔ Filename `tmp-data1.mount`
2. **Sample Unit File Content**

```
[Unit]
Description=Data1 mount
[Mount]
What=/dev/disk/by-label/data1
Where=/mnt/data1
Type=xfs
Options=defaults
[Install]
WantedBy=multi-user.target
```

3. **Activating and Managing the Mount Unit**
   - Reload systemd daemon `systemctl daemon-reload`
   - Enable and start the mount unit `systemctl enable mnt-data1.mount --now`
   - Start the mount unit immediately `systemctl start mnt-data1.mount --now`
   - Check the status of the mount unit `systemctl status mnt-data1.mount`
   - List all mounted filesystems `findmnt`

Redirect Logs to different volume

Temporary Mount the new partition to /x1

```
Mount -t xfs /dev/sda1 /x1
tail -1 /etc/mtab
```

Backup existing logs:

```
cp -r /var/log/* /x1
```

Delete content of /var/log without deleting the directory itself:

```
rm -rf /var/log/*
```

Mount the new volume to /var/log:

```
mount -t xfs /dev/sda1 /var/log
```

Update /etc/fstab to make the mount persistent:

```
echo '/dev/sda1 /var/log ext4 defaults 0 2' >> /etc/fstab
```

Restore the backed-up logs to the new volume:

```
cp -r /root/log_backup/* /var/log/
```

Verify the log files in the new volume:

```
ls -l /var/log/
```

Redirect Logs to remote server (from node1 to node2)

On node1 (source server):

1. **Ensure rsyslog is installed:** dnf install rsyslog
2. **Edit the rsyslog configuration to forward logs and stop local logging:**

Open or create /etc/rsyslog.d/remote.conf and add the following lines:

```
*.* @@node2:514
*.* @node2:514
*.* ~
```

The *.* @@node2:514 line forwards all logs to node2 using TCP (use @ for UDP), and the *.* ~ line discards all local logs.

3. **Restart rsyslog to apply the changes:** systemctl restart rsyslog

On node2 (destination server):

1. **Ensure rsyslog is installed:** `dnf install rsyslog`
2. **Edit rsyslog configuration to receive and store logs in separate directories:**

Open `/etc/rsyslog.conf` or create a new configuration file under `/etc/rsyslog.d/` and ensure the following lines are present:

```
module(load="imtcp")  # Load the TCP module
input(type="imtcp" port="514")  # Enable the input listener on port 514
# Rule for logging remote messages
template(name="RemoteLogs" type="string"
string="/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log")
if ($fromhost-ip != '127.0.0.1') then {
action(type="omfile" dynaFile="RemoteLogs")
stop}
```

This configuration sets up rsyslog to save incoming logs from remote hosts in /var/log/remote/<hostname>/<programname>.log.

3. **Create the directory to store remote logs:** `mkdir -p /var/log/remote`
4. **Set appropriate permissions for the directory:** `chmod -R 755 /var/log/remote`
5. **Restart rsyslog to apply the changes:** `systemctl restart rsyslog`
6. **Open port 514 in the firewall:**

```
firewall-cmd --permanent --add-port=514/tcp
firewall-cmd --reload
```

Testing the Setup:

1. **On node1**, send a test message: `logger "This is a test message"`
2. **On node2**, check if the message appears in the appropriate log file: `tail -f /var/log/remote/node1/`

Important Note:

Files in **/etc/rsyslog.d/** are processed after **/etc/rsyslog.conf**, typically in lexicographical order. This means that configuration files in **/etc/rsyslog.d/** are processed alphabetically, with the last processed file having the highest priority. If there are conflicting directives, the settings in the file processed last will take precedence. To ensure desired behavior, carefully manage the naming and ordering of files in **/etc/rsyslog.d/**.

Creating Partitions Using gdisk and GNU Parted

A. Using gdisk for Partitioning

`gdisk` is used for creating GUID Partition Tables (GPT), supporting partitions larger than the 2 TB limit of MBR (Master Boot Record) partition tables.

1. **Viewing Partition Table Information**
   - `gdisk /dev/sdc` Show partition table info for `/dev/sdc`.
   - `gdisk -l /dev/sdb` Shows if GPT is present.
   - `blkid /dev/sdb` Retrieve the block device's UUID.
2. **Creating a GPT Partition Table**
   - Follow similar steps as `fdisk` for creating partitions.

B. Using GNU Parted for Partitioning

GNU Parted can work with both MBR and GPT partition tables.

1. **Viewing Partition Table Info**
   - `parted /dev/sdb print` Show partition table info for `/dev/sdb`.
2. **Creating Partitions with Parted**
   - Start `parted` tool `parted`
   - Select device `select /dev/sdb`
   - Print partition table `p`
   - Create MBR or GPT label `mklabel msdos` or `mklabel gpt`
   - Create a primary partition `mkpart primary 1 1024` (500 MB primary partition)
   - Create an extended partition `mkpart extended 1025 -1` (Rest of the hard drive as extended partition)
   - Specify start `start 1`
   - Specify end `end 1024`
   - Print partition table `p`
   - Create a logical partition `mkpart logical 1025 4096` (Logical partition numbering starts from 5)
   - Print partition table `p`
   - Exit Parted `quit`
   - Verify partitions `lsblk`
3. **Resizing a Partition**
   - Enter resize mode `resizepart`
   - Parted `parted resizepart`
   - Choose partition number (e.g., 1) `Partition number? 1`
   - Specify new end size (e.g., 15 GB) `End? [10.0GB]? 15000`
4. **Deleting a Partition**
   - Remove a partition (e.g., partition 1) `rm 1`

Managing Logical Volumes, RAID, and Disk Encryption

Managing Software RAID

1. **View RAID Devices**
    - `cat /proc/mdstat` Displays RAID devices status.
2. **Create Software RAID**
    - `mdadm -C /dev/md0 -l 5 -n 3 /dev/sd{b,c,d}` Create a RAID 5 device.
    - `lsblk /dev/md0` Shows the newly created device.
    - `cat /proc/mdstat`
3. **Formatting and Mounting RAID**
    - `mkfs.xfs -L Rdata /dev/md0` Format RAID device with XFS.
    - `blkid /dev/md0` Get RAID device UUID.
    - `mount -t xfs /dev/md0 /media/raid5/` Mount the filesystem temporarily.
    - `df -h` Check disk space.
4. **Persisting RAID Configuration**
    - `mdadm --detail --scan >> /etc/mdadm.conf` Make RAID configuration persistent.
5. **Stopping RAID Device**
    - `mdadm --stop /dev/md0` Stops RAID device (use with caution).
6. **Adding Disk to mdadm Array**
    - `mdadm --add /dev/md0 /dev/sde` Add spare disk.
    - `mdadm --grow /dev/md0 --raid-devices=4` Increase number of active devices.
    - `umount /dev/md0` Unmount RAID device.
    - `xfs_growfs /mount/point` Grow XFS filesystem.
7. **RAID Hot Spare Configuration**
    - `mdadm --create /dev/md0 -l 5 -n 3 /dev/sd[c-e]1 -x 1 /dev/sdf1` Create RAID with spare.
    - `watch cat /proc/mdstat` Monitor RAID status.
    - Simulate disk failure `mdadm /dev/md0 --fail /dev/sdb`.

Logical Volume Management (LVM)

1. **Installing LVM**
   - `dnf install lvm2` Install LVM package.
   - `pvcreate /dev/sd{a,b,c}` Create physical volumes.
2. **Creating Volume Groups and LV using GB**
   - `vgcreate vg1 /dev/sd{a,b,c}` Create a volume group.
   - `lvcreate -n lv1 -L 2G vg1` Create a 2GB logical volume.
   - `lvcreate -n lv2 -l 100%FREE vg1` Use remaining space for another logical volume.
   - `mkfs.xfs -L lvdata /dev/vg1/lv1` Format logical volume.
3. **Creating an LV using PE**
   - `vgdisplay vg0 | grep "PE Size"` Determine the PE Size
   - `40GB = 40 * 1024 MB = 40960 MB` Convert the desired LV size to MB
   - `lvcreate -l 5000 -n lvdata vg0` Create the Logical Volume using PEs
4. **Mounting and Using LVM**
   - `mkdir /lvm1` Create a mount point.
   - Add entry in `/etc/fstab lvdata /mnt/lvm xfs defaults 0 0.`
   - `mount -a` Mount all filesystems in fstab.
   - Test LVM Copy data to `/mnt/lvm.`
5. **Resizing Logical Volumes**
   - `vgextend vg1 /dev/sdd` Extend volume group.
   - `lvextend -L +50G -r /dev/vg1/lv1` Extend logical volume (add 50G to the current size)
   - `lvextend -L 50G -r /dev/vg1/lv1` Extend logical volume (change the current size to 50G)
   - `xfs_growfs /mnt/lvm/ Grow` filesystem in case you not use -r option.
6. **Remove LVM**
   - `lvremove /dev/lvmraid/lvm1` Remove logical volume.
   - `vgremove lvmraid` Remove volume group.

Managing Storage with Stratis on Linux

1. Installing Stratis

Stratis simplifies storage management on Linux, providing advanced features for data protection and performance.

- Install Stratis

```
dnf install stratisd stratis-cli
```

- Enable and start the Stratis service

```
systemctl enable --now stratisd
systemctl status stratisd
```

2. Preparing Hard Disks for Stratis

- Check existing partitions

```
lsblk
blkid -p /dev/sd{h,i}
```

- Remove existing partitions if necessary

```
wipefs -a /dev/sd{h,i}
```

3. Creating a Stratis Pool

- Create a pool from multiple block devices

```
stratis pool create pool_1 /dev/sd{h,i}
```

- List Stratis pools

```
stratis pool list
```

4. Creating Filesystems from a Pool

- Create filesystems in the pool (e.g., for DB Data and Backup)

```
stratis fs create pool_1 fs_1
stratis fs create pool_1 fs_2
```

- List filesystems and check with lsblk

```
stratis fs list
lsblk
```

5. Mounting Filesystems

- Create mount directories and get filesystem UUID

```
blkid -p /dev/stratis/pool_1/fs_1
```

- Add entries to /etc/fstab and reload system daemons

```
systemctl daemon-reload
mount -a
df -h /disk/st1
```

6. Adding a Block Device to a Stratis Pool

- Add a new block device to an existing pool

```
stratis pool add-data pool_1 /dev/sdf
```

- List updated pools and block devices

```
stratis pool list
stratis blockdev list
```

7. Creating a Stratis Snapshot

- Take a snapshot of a filesystem

```
stratis fs snapshot pool_1 fs_1 snapshot_1
mount /dev/stratis/pool_1/snapshot_1 /mnt/snap1
```

- List filesystems to verify

```
stratis fs list
```

8. Deleting Stratis Configurations

- Dismount filesystems and remove entries from /etc/fstab.
- Destroy filesystems in the pool

```
stratis filesystem destroy pool_1 fs_1
stratis filesystem destroy pool_1 fs_2
```

- Destroy the pool

```
stratis pool destroy pool_1
```

Disk Encryption (LUKS)

1. **Install Required Packages** Install Cryptsetup `yum install -y cryptsetup.`
2. **Encrypt the Disk:** `cryptsetup luksFormat /dev/sdb`
3. **Open the Encrypted Disk:** `cryptsetup open /dev/sdb encrypted_disk1`
4. **Create a Filesystem:** `mkfs.ext4 /dev/mapper/encrypted_disk1`
5. **Mount the Encrypted Disk:**
   - `mkdir /mnt/encrypted`
   - Edit /etc/fstab/`dev/mapper/encrypted_disk1 /mnt/encrypted ext4 defaults 0 2`
   - Edit /etc/crypttab `encrypt1        /dev/sda        none`
6. **Close the LUKS Device**
   - **Unmount the Filesystem:** `umount /dev/mapper/encrypted_disk1`
   - **Close the LUKS Device:** `cryptsetup close encrypted_disk1`
   - **Remove Entries from crypttab and fstab.**

Setting Up NFS (Network File System) on Linux

Server-Side Configuration

1. **Configuring Firewall for NFS**
   - Allow NFS service `firewall-cmd --add-service=nfs --permanent`
   - Allow necessary services for mounting and RPC `firewall-cmd --add-service={nfs3,mountd,rpc-bind} --permanent`
   - Reload the firewall `firewall-cmd --reload`
   - Verify added services `firewall-cmd --list-services`
2. **Checking NFS Server Service Status**
   - `systemctl status nfs-server.service`
3. **SELinux Configuration for NFS**
   - Allow NFS exports read/write `setsebool -P nfs_export_all_rw 1`
4. **Configuring NFS Shares**

```
vim /etc/exports
/var/nfsshare  192.168.244.161(rw,sync,no_root_squash,no_subtree_check)
/mnt/share *(rw,sync,no_root_squash)
```

5. **Add more than one IP**

```
/var/nfsshare 192.168.244.161(rw,sync,no_root_squash,no_subtree_check)
/var/nfsshare 192.168.244.100(ro,sync,no_root_squash,no_subtree_check)
```

Then run `exportfs -rav`

6. give full network access

```
/var/nfsshare 192.168.244.0/24(rw,sync,no_root_squash,no_subtree_check)
```

7. **Enabling and Starting NFS Server**
   - Enable NFS server at boot `systemctl enable nfs-server.service`
   - Start NFS server `systemctl start nfs-server.service`
   - Check shared directories `showmount -e localhost`

Client-Side Configuration

1. **Mounting NFS Share Manually**
   - Create a mount point `mkdir /mnt/share`
   - Mount the NFS share `mount -t nfs 10.0.0.4:/mnt/share /mnt/share`

Mount using Systemd

vim /etc/systemd/system/mnt-share.mount

```
[Unit]
Description=Mount NFS Share

[Mount]
What=192.168.244.123:/mnt/share
Where=/mnt/share
```

```
Type=nfs
Options=defaults

[Install]
WantedBy=multi-user.target

[Service]
ExecStartPre=/bin/sleep 50
#Restart=on-failure
Restart=always
RestartSec=30s
```

vim /etc/systemd/system/mnt-nfsshare.mount

```
[Unit]
Description=Mount NFS Share
[Mount]
What=192.168.244.123:/var/nfsshare
Where=/mnt/nfsshare
Type=nfs
Options=defaults
[Install]
WantedBy=multi-user.target
[Service]
ExecStartPre=/bin/sleep 50
#Restart=on-failure
Restart=always
RestartSec=30s
```

NFS Automount Configuration (Client-Side)

AutoFS automatically mounts filesystems when they are accessed and unmounts them after a period of inactivity.

1. **Installing Autofs**
   - `dnf install -y autofs`
2. **Configuring Automount**
   - Edit auto master file `vim /etc/auto.master`
     - Add line `/- /etc/auto.share`
   - Edit auto share file `vim /etc/auto.share`
     - Add mount details `/mnt/share   -fstype=nfs,timeout=60   10.0.0.4:/mnt/share`
3. **Enabling and Starting Autofs**
   - Enable and start Autofs `systemctl enable autofs --now`
   - Access the share to trigger automount `cd /mnt/share`
   - Check mounted filesystems `mount | tail -3`
   - Start Autofs if not already running `systemctl start autofs –now`

Redirect all user home directories to /rhome

- **Configure AutoFS Master File**: Edit the AutoFS master configuration file, usually located at /etc/auto.master to include a line for the home directories.

  ```
  /rhome    /etc/auto.home
  ```

  This line means that access to paths under /home will be managed by AutoFS according to the configurations in /etc/auto.home.
- **Create and Configure AutoFS Map File**: Create and edit the /etc/auto.home file (or whatever you named it in the master file) to specify the redirection to /rhome. The content of /etc/auto.home might look something like this:

  ```
  *   -fstype=nfs,rw   profiles.abdelwahed.me:/rhome/&
  ```
- **Restart AutoFS**

  ```
  systemctl restart autofs
  ```

Managing Swap Space in Linux

A. Monitoring Swap Usage

1. **Viewing Current Swap Usage**
   - `cat /proc/swaps` Display swap usage and configuration.
   - `free -h` Show memory and swap usage with human-readable output.

B. Creating a Swap Partition

1. **View Current Swap Information**
   - `swapon -s` List all swap spaces in use.
2. **Setting Up a Swap Partition**
   - `mkswap /dev/sdb5` Prepare the partition `/dev/sdb5` as swap space.
3. **Enabling and Disabling Swap on a Partition**
   - `swapon /dev/sdb5` Enable swap space on `/dev/sdb5`.
   - `swapoff /dev/sdb5` Disable swap space on `/dev/sdb5`.
   - Add to `/etc/fstab` for persistent configuration
     - `UUID="1568a6e5-0768-42b4-a707-bffa16a83ab5" swap swap defaults 0 0`

Note For systems with multiple swap partitions, each one can be assigned a priority number. A higher number indicates a higher priority.

C. Creating a Swap File

1. **Creating a Swap File**
   - `dd if=/dev/zero of=swapfile bs=1G count=1` Create a 1GB swap file.
   - `mkswap swapfile` Set up the swap file.
   - `chmod 0600 swapfile` Secure the swap file.
2. **Enabling Swap on a File**
   - Add to `/etc/fstab` for persistent configuration
     - `UUID="b28200f4-441a-41d0-bd4d-b8af36a13c48" swap swap defaults 0 0`
   - `swapon swapfile` Activate the swap file.

D. Setting Swap Priority

1. **Adjusting Swap Priority**
   - `swapoff -a` Disable all swap spaces.
   - `swapon -p 10 /dev/sdb1` Set priority for swap space `/dev/sdb1` to 10.
   - Adjust swap priority in `/etc/fstab`
     - `UUID=4434443-5465656-7-655665-8867 swap sw,pri=5 0 0`

Copy files from one system to another.

using rsync

rsync is a powerful tool utilized across Linux and Unix systems for tasks such as backup, synchronization, and file transfers.

1. **Basics of rsync**
   - Fundamental syntax `rsync [options] source destination`
   - Installing `rsync yum install rsync -y`
2. **Local File and Directory Copy/Sync**
   - Copying a single file with compression in a human-readable format `rsync -zvh backup.tar /tmp/backups/`
   - Copying or synchronizing a directory locally `rsync -avzh /root/rpmpkgs /tmp/backups/`
3. **Remote File and Directory Copy/Sync**
   - Copying a directory from the local server to a remote server `rsync -avz rpmpkgs/ root@192.168.0.101/home/`
   - Copying or synchronizing a directory from a remote server to the local machine `rsync -avzh root@192.168.0.100/home/source/rpmpkgs /tmp/myrpms`
4. **Advanced rsync Options**
   - Displaying progress during transfer `rsync -avzhe ssh --progress /home/rpmpkgs root@192.168.0.100/root/rpmpkgs`
   - Transferring files with size restrictions `rsync -avzhe ssh --max-size='200k' /var/lib/rpm/ root@192.168.0.100/root/tmprpm`
   - Deleting source files after a successful transfer `rsync --remove-source-files -zvh backup.tar /tmp/backups/`
   - Limiting I/O bandwidth during transfers `rsync --bwlimit=100 -avzhe ssh /var/lib/rpm/ root@192.168.0.100/root/tmprpm/`

using scp (Secure Copy Protocol)

- **Copying from a remote to a local machine** `scp -r aabdelwahed@10.0.0.4:/etc/passwd .`
- `scp -P 1515 root@10.0.0.4:/etc/passwd .` incase ssh port is 1515
- **Copying from one remote to another remote machine** `scp username@host1:/remote/directory/file.txt username@host2:/remote/directory/`
- **Copying multiple files locally** `scp file01 file02 file03 /repo`
- **Recursively Copy Directories** (-r) `scp -r /path/to/local_directory user@remote_host:/path/to/destination_directory`
- **Enable Compression (-C)** `scp -C /path/to/local_file user@remote_host:/path/to/remote_destination`

Managing System Logging with systemd-journald

Basic Commands for systemd-journald

1. **Checking Service Status**
   - `systemctl status systemd-journald.service` Check the status of the journald service.
2. **Viewing Logs**
   - Live system logs `journalctl -f`
   - Live logs for a specific service (e.g., SSH) `journalctl -u sshd -f`
3. **Filtering Logs by Time**
   - Logs in a specific time frame `journalctl --since "2022-8-21 800" --until "2022-8-22"`
   - Logs since today `journalctl --since today`
   - Logs since yesterday `journalctl --since yesterday`
4. **Listing and Viewing Boot Logs**
   - List recorded boot sessions `journalctl --list-boots`
   - Logs from the current boot `journalctl -b`
   - **Displaying Logs in Reverse Order** `journalctl -r`
   - **Limiting the Number of Log Entries** `journalctl -n 100`
5. **Filtering Logs by Priority**
   - Show only error logs and higher `journalctl -p err, journalctl -p warn`
   - `journalctl -u sshd -p err --since "yesterday"`
6. **Viewing Logs of Specific Units**
   - Logs for a specific unit (e.g., cron) `journalctl -u crond, journalctl -u sshd -u httpd`
7. **Kernel Logs**
   - Display only kernel logs `journalctl -k`
8. **Filtering Logs by Recent Times**
   - Logs from the last hour `journalctl --since "1 hour ago"`
   - Logs from the last minute `journalctl --since "1 minute ago"`
9. **Checking Disk Usage of Logs**
   - Display journal disk usage `journalctl --disk-usage`
10. **Filtering Logs by Process, User, or Group ID**
    - **`journalctl _PID=1234,`** `journalctl _UID=1000, journalctl _GID=1000`

Preserving the Systemd Journal

1. **Default Storage Behavior**
   - By default, `systemd-journald` uses volatile storage (logs stored in memory and lost on reboot) unless `/var/log/journal` exists.
2. **Enabling Persistent Storage**

   `mkdir -p /var/log/journal`

   Edit `/etc/systemd/journald.conf and` set `Storage=persistent.`

   `systemctl restart systemd-journald.service`

Managing SELinux (Security-Enhanced Linux)

Introduction to SELinux

SELinux is an advanced security architecture integrated into the Linux kernel, developed by the National Security Agency (NSA) and the Linux community. It enhances Linux's security by enforcing mandatory access controls over processes, applications, and users, and controls file access.

Basic SELinux Commands and Configuration

1. **Checking SELinux Status**
   - `getenforce` Display the current SELinux mode.
2. **Setting SELinux Mode**
   - `setenforce 0` or `setenforce 1` Temporarily set SELinux to Permissive (0) or Enforcing (1) mode.
3. **Configuring SELinux in System Config**
   - Edit SELinux configuration file `vim /etc/sysconfig/selinux`
   - To disable SELinux on boot, change to `SELINUX=disabled.`
4. **Adjusting SELinux for Boot**
   - Set `selinux=0` in the GRUB2 configuration for boot-time disablement.
5. **Managing SELinux Ports**
   - `semanage port -l` List SELinux port contexts.
   - `semanage port -l | grep http` Check if a specific port (e.g., 80 for HTTP) is allowed by SELinux.
6. **Adding/Removing Custom Ports in SELinux**
   - `semanage port -a -t <SELinux_type> -p <protocol> <port_number>`
   - `semanage port -a -t http_port_t -p tcp 5555` allow 5555 for http
   - `semanage port -a -t unreserved_port_t -p tcp 6789` allowing any service to use 6789
   - `semanage port -d -t http_port_t -p tcp 5555` Remove the custom port definition
7. **Configuring HTTP Service for New Port**
   - Edit Apache HTTP configuration `vim /etc/httpd/conf/httpd.conf` to allow the new port.
8. **Adjusting Firewall for the New Port**
   - Add the custom port to the firewall
   `firewall-cmd --add-port=5555/tcp --permanent`
   - Reload the firewall configuration `firewall-cmd --reload.`
9. **Restarting HTTP Service**
   - `systemctl restart httpd` to apply changes.
   - **Installing SELinux Management Tools**
   - If `semanage` is not found, install necessary tools
   - `yum install policycoreutils-python-utils`

Understanding SELinux Modes

- **Enforcing** SELinux enforces its policies and denies access based on these policies.
- **Permissive** SELinux allows actions that would be denied in enforcing mode but logs them.
- **Disabled** SELinux is completely turned off.

Note

- SELinux provides an additional layer of security by managing access permissions for processes, applications, and files, complementing traditional Linux permissions.
- Changes to SELinux policies or modes should be performed carefully, as incorrect settings can affect system functionality and security.
- Always ensure that your firewall settings are consistent with SELinux policies for smooth operation of services.

Managing Firewall with Firewalld on Linux

Introduction

Firewalld is a dynamic firewall management tool available on many Linux distributions. It provides a way to configure and manage network firewalls, including creating, modifying, and deleting firewall rules.

Basic Firewalld Configuration and Management

1. **Firewall Configuration File**
   - Location `/etc/firewalld/firewalld.conf`
2. **Checking Firewalld Service Status**
   - `systemctl status firewalld.service` View the status of the Firewalld service.
3. **Enabling Firewalld Service**
   - `systemctl enable firewalld.service` Enable Firewalld to start at boot.
4. **Starting/Stopping Firewalld Service**
   - `systemctl start firewalld.service` Start Firewalld service.
   - `systemctl stop firewalld.service` Stop Firewalld service.
5. **Managing Firewall with firewall-cmd**
   - `firewall-cmd` Primary command to manage the firewall.
   - `firewall-cmd --list-all` List all current settings.
6. **Modifying Firewall Services**
   - Remove a service `firewall-cmd --remove-service=ssh`
   - Add a service temporarily firewall`-cmd --add-service=ssh`
   - Add a service permanently `firewall-cmd --add-service=ssh --permanent`
   - Reload firewall to apply permanent changes `firewall-cmd --reload`
7. **Listing Allowed Services and Ports**
   - `firewall-cmd --list-services` List all allowed services.
   - `firewall-cmd --list-ports` List all open ports.
8. **Adding Ports**
   - Add a port permanently `firewall-cmd --add-port=5050/tcp --permanent`
   - Reload firewall to apply changes `firewall-cmd --reload`
9. **Working with Zones**
   - List available zones `firewall-cmd --get-zones`
   - List all settings for all zones `firewall-cmd --list-all-zones`
   - Get default zone `firewall-cmd --get-default-zone`
   - Set default zone `firewall-cmd --set-default-zone=internal`
   - Add service to a zone `firewall-cmd --zone=public --add-service=http`
   - List services in a zone `firewall-cmd --zone=public --list-services`
10. **Understanding Firewalld Service Definitions**
    - Service definitions are stored in /usr/lib/firewalld/services.
    - These XML files define ports and protocols for services.
    - They are crucial for configuring firewall rules.

Tuning System Performance with tuned on Linux

Overview of Dynamic Tuning with tuned

The tuned daemon dynamically tunes system settings based on current workload and activity. It adjusts parameters for storage, CPU, and network devices to optimize performance. This is done through predefined tuning profiles that cater to various use cases.

Managing tuned Service

1. **Checking tuned Service Status**
   - To check if tuned is active
     ```
     systemctl status tuned.service
     ```
2. **Installing tuned**
   - If tuned is not installed
     ```
     yum install tuned
     ```
3. **Enabling and Starting tuned Service**
   - Enable and start tuned
     ```
     systemctl enable --now tuned
     ```

Using tuned-adm for Tuning Management

1. **Viewing Active Tuning Profile**
   - Check which tuning profile is currently active
     ```
     tuned-adm active
     ```
2. **Listing Available Tuning Profiles**
   - List all available profiles
     ```
     tuned-adm list
     ```
3. **Applying a Tuning Profile**
   - Apply a specific profile (e.g., throughput-performance)
     ```
     tuned-adm profile throughput-performance
     ```
4. **Getting Recommended Profile**
   - Display the recommended tuning profile for your system
     ```
     tuned-adm recommend
     ```
5. **Turning Off Dynamic Tuning**
   - To disable all dynamic tuning
     ```
     tuned-adm off
     ```

Notes on Using tuned

- **Profile Selection** Choose a profile that best matches your system's role. For example, use throughput-performance for servers, balanced for regular desktops, or powersave for laptops.
- **Custom Profiles** Advanced users can create custom profiles tailored to specific needs.
- **Monitoring Effects** Observe system behavior after applying a profile, as the impact can vary based on hardware and workload.
- **Disabling tuned** In some specific scenarios, it may be beneficial to disable tuned, especially on systems where manual tuning is preferred.
- **Compatibility with Other Tuning Tools** Be cautious when using tuned in conjunction with other performance tuning tools to avoid conflicting settings.

Linux Virtualization Management Using KVM

1. **KVM Installation**

```
dnf install -y qemu-kvm libvirt libvirt-daemon libvirt-daemon-driver-qemu virt-install virt-manager
```

```
egrep -c '(vmx|svm)' /proc/cpuinfo
```

2. **Enabling and Starting libvirtd**

   - Enable the libvirtd service for managing virtual machines

     ```
     systemctl enable libvirtd
     systemctl start libvirtd
     ```

   - Check the status of `libvirtd`

     ```
     systemctl status libvirtd
     ```

1. **Creating VMs via Shell**

```
virt-install --name vm1 --memory 512 --vcpus 1 --disk size=8 --cdrom /path/to/rhel-iso
```

Managing VMs Using Virsh

1. **Using Virsh for VM Management**
   - `sudo virsh list --all`
   - `virsh net-list` List network interfaces for VMs.
   - `virsh` Enter the Virsh command-line interface.
     - `list` List all VMs.
     - `start vm1` Start a VM named vm1.
     - `shutdown vm2` Shutdown a VM named vm2.
     - `reboot vm1` Reboot vm1.
     - `autostart vm2` Set vm2 to start automatically on boot.
     - `autostart vm1 --disable` Disable autostart for vm1.

Container

Containers are a powerful tool for developing, deploying, and managing applications more efficiently. In the context of RHEL (Red Hat Enterprise Linux), understanding container concepts is essential, especially with tools like Podman and Buildah that are part of the Red Hat ecosystem. Here's a simple summary of the key container concepts along with examples:

1. Container Images

- **Concept:** A container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, libraries, environment variables, and configuration files.
- **Example:** The Fedora image in a container can be used as the base to install and run applications.

2. Containers

- **Concept:** A container is a runtime instance of an image—what the image becomes in memory when executed (i.e., an image with state, or a user process).
- **Example:** Running an Apache server using the Fedora image. You can start the container using Podman with:

```
podman run -d --name my-apache -p 8080:80 fedora/apache
```

3. Container Registries

- **Concept:** A container registry is a repository, or collection of repositories, used to store container images.
- **Example:** Docker Hub, from where you can pull images like MariaDB or Fedora. In RHEL, you can configure private registries or use Red Hat's registry.

4. Container Orchestration

- **Concept:** Container orchestration automates the deployment, management, scaling, networking, and availability of containerized applications. Common tools include Kubernetes.
- **Example:** Using OpenShift, a Kubernetes distribution maintained by Red Hat, to manage multiple containers across several RHEL servers.

5. Pods

- **Concept:** A pod is a group of one or more containers, with shared storage/network, and a specification for how to run the containers. Pods are the atomic unit on the Kubernetes platform.
- **Example:** A pod in OpenShift might contain a container running a web server and another container running a background application that updates the web server's content.

6. Volumes

- **Concept:** Volumes are persistent data stores for containers. They are used to keep data even when containers are restarted or removed.
- **Example:** Using a Podman command to mount a volume to a MariaDB container ensures that database data persists across container restarts:

```
podman run -d --name my-mariadb -v my-mariadb-data:/var/lib/mysql mariadb
```

7. Networking

- **Concept:** Containers can communicate with each other and the outside world through networking. This involves configuring network settings such as network bridges, IPs, and ports.
- **Example:** Mapping port 80 of a container to port 8080 on the host using Podman:

```
podman run -d --name my-webapp -p 8080:80 my-custom-webapp
```

- **Concept:** Tools like Buildah and Dockerfile are used to create container images from scratch or by modifying existing images.

- **Example:** Using Buildah to build a custom Apache image on RHEL:
- buildah from fedora

  ```
  buildah run $containerID -- dnf install httpd -y
  ```

- **Install Podman**

  ```
  sudo dnf -y install podman
  ```

- **Search for Container Images**

  To find images in public registries, use the podman search command. This allows you to search for images by name or description.

  ```
  podman search fedora
  ```

- **Run Hello World**

  ```
  podman pull hello-world
  podman images
  podman run hello-world
  ```

- **nginx**

  ```
  podman pull nginx
  podman run -d --name my-nginx -p 8080:80 nginx
  podman logs my-nginx
  podman run -d --restart=always --name my-nginx -p 8080:80 nginx
  podman inspect nginx:latest
  ```

- **Open Ubuntu in an Interactive Terminal**

  If the Ubuntu image does not exist, Podman will automatically download it before running it:

  ```
  podman run -it ubuntu bash
  ```

- **Detach from a Container Without Stopping It**

  With Podman, you use `Ctrl-p` followed by `Ctrl-q` to detach from a container without stopping it.

- **Reattach to a Container**

  First, find the container ID using `podman ps,` and then:

  ```
  podman attach <container_id>
  ```

- **List All Containers, Including Stopped Ones**

  ```
  podman ps -a
  ```

- **Run a Container in Detached Mode**

  ```
  podman run -it -d --name myfirst ubuntu
  ```

- **Running Containers Interactively**: For tasks that require user interaction or to run an application in the foreground, remove the -d flag:

  ```
  podman run -it --name my-interactive-fedora fedora /bin/bash
  ```

- **Start and Stop Containers**

  ```
  podman stop myfirst
  podman start myfirst
  ```

- **Rename a Container**

  Podman does not support renaming a container directly. You would typically stop, commit the container to an image, and then start a new container with the new name:

  ```
  podman stop myfirst
  podman commit myfirst new-image-name
  podman run --name myapp new-image-name
  ```

- **Remove Containers and Images**

  ```
  podman rm myapp  # Remove a stopped container
  podman rm -f myapp  # Force remove a running container
  podman rm -f $(podman ps -aq)  # Remove all containers, force stop if needed
  podman rmi -f ubuntu  # Force remove an image
  podman rmi -f $(podman images -aq)  # Remove all images
  ```

- **Execute Commands Inside a Running Container**

  ```
  podman exec myfirst /bin/ps -aux
  podman top myfirst
  ```

- **Monitor Containers**

  ```
  podman stats myfirst
  podman events --since '1h'
  podman logs my-fedora
  podman inspect myfirst
  ```