RHCSA Rapid Track | Quick Guide

Version 24.11

Ahmed Abdelwahed
ahmed@abdelwahed.me
LinkedIn

System Utility Commands

- `date` Shows the current date and time.
- `uptime` Shows the system running time and active users.
- `w` Shows the uptime output and working users.
- `watch -n 3 uptime` refreshes the uptime every 3 seconds.
- `tty` Displays the current logged-in shell.
- `bc` Provides a scientific calculator (binary calculator).
- `whoami` Prints the username associated with the current effective user ID.
- `lscpu; cat /proc/cpuinfo` Displays CPU information.
- `lsmem; free -h; cat /proc/meminfo` Shows memory status information.
- `lsblk -f` Displays a tree view of all block devices with additional information, including the filesystem type and UUID.
- `hostname; hostname -f` Displays the full hostname.
- `hostname -I` Shows the IPv4 address of the current machine. While `hostname -i` shows both IPv4 and v6.
- `uname -a` Prints details about the machine and operating system.
- `last` display a list of the most recent logins by users.
- `lastb` displays a list of failed login attempts by users.

Manage Files from the Command Line

Filesystem Structure (Linux File System Layout)

- `tree -L 2 / > system_structure.txt` Saves a two-level Linux directory hierarchy to system_structure.txt.
- `man hier` Displays the manual explaining the Linux filesystem hierarchy.
- `man file-hierarchy` Shows details about the filesystem hierarchy using an alternative manual.

Listing Files (Directories)

- `pwd` Prints the working directory.
- `ls` Lists files in the current directory.
- `ls -a` Lists all files, including hidden ones.
- `ls -lt /etc` Shows a long list, sorted by time.
- `ls -ld /etc Lists` information about the specified directory.
- `ls -aF /etc` Shows files and their types, with directories ending with a /

Creating Files and Directories

- `mkdir -p par1/par2/dir` Creates a directory with parent directories as needed.
- `touch f1 f2 f3` Creates multiple files.
- `touch "my file.txt"` Creates a file with spaces in its name.

File Display Commands

- `cat -n /etc/passwd` Displays file contents with line numbers.
- `less /etc/passwd` Used to read long files; press 'q' to quit.
- `less /etc/passwd`

RHCSA Rapid Track | Quick Guide

Monitoring Files

- `tail -f /var/log/messages` Monitors a system log file in real-time.

File Maintenance Commands

Copying Files

- `cp file1 /media/file4` Copies and renames a file with interactive prompts.
- `cp -f /etc/*.conf /home/data` Copies all .conf files, overwriting existing ones without prompting.

Moving and Renaming Files

- `mv file1 file2` Renames a file.
- `mv file1 file2 file3 dir Moves` multiple files to a specific directory.

Removing Files and Directories

- `rmdir` or `rm -r` Removes a directory.
- `shred passwd` Destroys file content.
- `shred -u passwd` Destroys and removes a file.

GREP and EGREP (Global Regular Expression Print)

- `grep -e 'root' -e 'aabdelwahed' /etc/passwd` Searches for lines containing either "root" or "aabdelwahed" in the `/etc/passwd` file using multiple patterns with `-e`.
- `egrep -i 'root|aabdelwahed' /etc/passwd` Searches for "root" or "aabdelwahed" in `/etc/passwd`, ignoring case sensitivity with `-i` and using extended regular expressions with `egrep`.

Wildcards

Wildcards are special characters used to represent one or more characters in a file or directory name.

- `*` Matches any number of characters, including none.
- `?` Matches exactly one character.
- `[]` Matches any one of the characters inside the brackets.
- `{}` Allows you to specify a list of alternative patterns.
- `!` Negates the pattern, matching any file that does not match the pattern.
- `ls *.txt` Lists all files in the current directory with a .txt extension.
- `rm file?.txt` Deletes matching files.
- `cp [ab]*.txt destination` Copies matching files.
- `touch file0{1..9}` Creates 9 files.
- `rm file0*` Removes all files matching the pattern.
- `mkdir dir{1..10}, touch abc0{1..9}-xyz` Creates multiple files and directories.
- `rm *-xyz` Removes everything ending with "xyz."
- `touch dir{1..10}/file{1..100}` Creates 100 files in each directory.

Redirection

- `>` Redirects stdout to a file.
- `>>` Appends stdout to a file.
- `cat passwd > passwd_orig` Redirects input to output.
- `df -h > newfile` Overwrites file content.
- `cat /dev/null > ahmedfile` Deletes file content.
- `echo “wooooow” > passwd` Overwrites content.
- `echo “wooow” >> passwd` Appends content.

Making Links Between Files (Soft and Hard Links)

Hard Links

- `ln source_file hard_link_name` Creates a duplicate directory entry pointing to the same inode as the source file. Both files share the same data, and changes to one reflect in the other. Deleting one does not affect the other.

Symbolic (Soft) Links

- `ln -s source_file_or_directory symlink_name` Creates a pointer (shortcut) to the source file or directory. The symlink becomes invalid if the source is deleted or moved.

Create, view, and edit text files

Linux File Editor (Vim)

- `Vimtutor` interactive tutorial that introduces users to the basics of Vim

Command Mode

- **Go to First Line** `1G` or `gg`
- **Go to Last Line** `G`
- **Go to Specific Line** `2G` or `20gg` for second or 20th line
- **End and Start of Line** `^` and `$`
- **Undo** `u, shift u`
- **Save and Exit** `shift zz`
- **Exit Without Save** `shift zq`

Inserting and Appending

- **Before Cursor** `i`
- **New Line Below** `o`
- **New Line Above** `O`
- **Start of Line** `I`
- **After Cursor** `a`
- **End of Line** `A`

Copying and Pasting

- **Copy Letter/Word/Line** `yl, yw, yy`
- **Copy Multiple Lines** `20yy`
- **Paste Below/Above** `p, P`

Deleting

- **Delete Letter/Word/Line** `dl, dw, dd`
- **Delete Multiple Lines** `20dd`
- **Delete to End of Line/File** `d$, dG`

Changing Text

- **Delete and Change** `cl, cw, cc`
- **Change Case** `Shift+~, g~~, gUU`
- **Merge Lines** `shift J`

Execute Mode

- **Save and Exit** `x, wq`
- **Force Exit** `q!`
- **Highlight Search** `se hlsearch, se nohlsearch`
- **Substitute Words** `%s/install/config/g`
- **Delete Lines** `$d, 1,9d, %d`
- **Add Empty Lines** `%s/$/\r/g`

Visual Mode

- Use `Ctrl+V > Shift+I > # > Esc` for commenting multiple lines.
- Use `Ctrl+V > y` copy multiple lines.
- Use `Ctrl+V > d` delete multiple lines.

Set Vim Defaults with .vimrc (Custom Vim)

- `set number` Displays line numbers in the editor.
- `set ignorecase` Makes search operations case-insensitive.
- `set hlsearch` Highlights all matches of the last search pattern.
- `set smartcase` Overrides `ignorecase` and makes searches case-sensitive if the search pattern contains uppercase letters.
- `set arabic` Enables Arabic language support, including right-to-left text direction and appropriate text shaping.
- `set noarabic` Disables Arabic language support.

Vim Tips

- **Lock and Unlock** `ctrl+s, ctrl+q`
- **Add New Screens** `ctrl+w+n, ctrl+w+v`
- **Move Between Screens** `ctrl+w.`
- **Global Settings** Edit `/etc/vimrc`
- **View the contents without editing** `vim -R /etc/passwd`

Other Editors

- **Nano** `nano nf2`
- **Gedit** `gedit file`

Manage local users and groups

ID Command for Managing Users and Groups in Linux

- **Showing User Identification** Utilizing the `id` command displays key information about Linux users. For instance, `id test1` would yield the user ID, group ID, and any associated groups.
  - **UID** The User ID, with the root ID being 0 and normal user IDs starting from 1000. Those under 1000 are special and system-related.
  - **GID** Primary (private) Group ID, which usually mirrors the UID. Private groups help maintain data privacy.
  - **Groups** These are secondary groups associated with file access permissions.
  - **Identifying a Specific User** `id test1 uid=1003(test1) gid=1003(test1) groups=1003(test1)` shows user identification for 'test1'.

Creating User Accounts in Linux

- **With Default Settings** `useradd user1` will create 'user1' with all default settings.
- **With Custom Options** `useradd -c "test account" -u 5002 -M -N -g user1 -G sales,hr -s /bin/sh ahmed`
  - `-c "test account"` This sets the comment (or full name) for the user to "test account".
  - `-u 5002` This sets the user ID (UID) for the new account to 5002.
  - `-M` This instructs the command not to create a home directory for the user.
  - `-N` This means do not create a group with the same name as the username. Without this, useradd would create a group named "ahmed" by default when you add the "ahmed" user.
  - `-g user1` This sets the primary group of the new user to "user1".
  - `-G sales,hr` This adds "ahmed" to the supplementary (or additional) groups "sales" and "hr".
  - `-s /bin/sh` This sets the default shell for the user to /bin/sh.
  - `ahmed` This is the name of the user being added.

Managing User and Password Age

- `pwscore` evaluates the quality of a given password by assigning it a score between 0 and 100.
- `chage -d 0 -M 42 -m 2 -W 4 -E 2024-12-31 user1`

  `-d 0` Sets the last password change date to the epoch (January 1, 1970). Setting it to 0 will typically force the user to change their password the next time they log in.

  `-M 42` Sets the maximum number of days the password is valid (before it expires) to 42 days.

  `-m 2` Sets the minimum number of days before which the password cannot be changed (once changed) to 2 days.

  `-W 4` Sets the number of days to warn the user before the password expires. In this case, the user will receive a warning 4 days before their password is set to expire.

  `-E 2024-12-31` Sets the account expiration date to December 31, 2024. After this date, the user will not be able to log in.

  `user1` the target user account for which these changes are being made.
- **Setting Last Password Change** `chage -d 2024-6-13 ahmed`
- `Passwd -e user1` expire a user's password immediately

- **Blocking Shell (non-interactive shell)** `chsh -s /usr/sbin/nologin ahmed`
- **Deleting User** `userdel -r ahmed` to remove user and their home directory.

Managing Local Groups

- **Linux Identity System** Linux divides identity into owner, individual user, groups of users, and the world.
- **Viewing Group Membership** Commands like `groups, groups username`, and `getent group` show group membership.

Creating Local Groups

- **Creating and Viewing Groups** Utilize `groupadd` to create groups and `cat /etc/group`, `getent group`, or `grep` to view groups.
- **Adding New Group** `groupadd sales` to create a 'sales' group.
- **Creating Group with Specific ID** `groupadd -g 555 admins`.

Group Membership Management (Secondary Group Modifications)

- `usermod -G wheel aabdelwahed` Set `wheel` as the sole secondary group for `aabdelwahed`, removing all others.
- `usermod -aG wheel aabdelwahed` Add `wheel` as a secondary group for `aabdelwahed` without removing current secondary groups.
- `usermod -aG wheel,admins ahmed` Add `wheel` and `admins` as secondary groups for `ahmed`.

Group Membership Administration with gpasswd

- `gpasswd -a user01 wheel Add user01` to the `wheel` group.
- `gpasswd -M user01,user02,user03 sales` Explicitly set members of `sales` group, overwriting existing ones.
- `gpasswd -d user01 sales` Remove `user01` from `sales` group.
- `sudo usermod -G group2 user1`

Querying Group Memberships

- `groupmems -lg wheel` List members of the `wheel` group.
- `lid ahmed` List groups associated with `ahmed`.
- `lid -g wheel` Display members of `wheel` group.

Group Management

Modifications

- `groupmod -n finance sales` Rename `sales` group to `finance`.
- `groupmod -g 1100 hr` Set GID of `hr` group to 1100.
- `groupdel finance` Remove the `finance` group.

Visudo (Enable Sudo)

- **Enabling Sudo** Utilize `visudo` to specify who can run which command without needing the root password.
- **Enable Specific Access** Enable `%wheel ALL=(root) ALL`, and add specific users.
- **Enabling sudo Access** Edit specific lines or add users to the 'wheel' group. `usermod -G wheel aabdelwahed`
- **Disabling 5-Minute Timeout** Set `timestamp_timeout=0`
- **Restart SSH Service** Use `systemctl restart sshd` to apply changes.

Control access to files with Linux file system permissions

Right Access and Commands

- **Read (R)**
  - **Files** View contents with `cat, less, more, tac`
  - **Directories** List contents with `ls`
- **Write (W)**
  - **Files** Modify contents with `echo, cat, vim`
  - **Directories** Create or remove with `mkdir, rm`
- **Execute (X)**
  - **Files** Allows execution if it's a script or program.
  - **Directories** Change to the directory with `cd`

Alphabet vs Numerical Syntax for Permissions

Numerical Representation

- `0` for No permissions; `1` for Execute only; `2` for Write only (used with command redirection), `3` for Write and Execute; `4` for Read only; `5` for Read and Execute, `6` for Read and Write, defining various access levels for users on.

Listing Permissions

- **Viewing Specific File Permissions** `ls -l file01`
- **Viewing Full Metadata** `stat file01`

Modifying Permissions with chmod

- **Changing Permissions** The `chmod` command is used to change file and directory permissions.
- **Symbolic Mode** Use characters like `u` for owner, `g` for group, `o` for others, and `a` for all, combined with `+`, `-`, or `=` to add, remove, or set specific permissions.
  - `chmod uo+x, g-w file01` to add execute permission to the owner and others and write only for group.
  - `chmod u=r,g=rw,o=rwx file01` sets specific permissions for user, group, and others.
- **Numeric Mode** Use numerical values to define permissions.
  - `chmod 755 file01` to set read, write, and execute for owner, and read and execute for group and others.

Managing File Ownership

- **Use `chown` Command** Change user and group ownership.
  - **Change User Owner** `chown user1 file01` (only root can do this).
  - **Change Group Owner** `chgrp sales file01` or `chown :Sales file01`
  - **Change Both** `chown user1:Sales file01`
  - **Recursive Change** `chown -R user1:Sales ./` for current directory and subdirectories.
- **View Numeric IDs** `ls -ldn dir1` to view numeric user and group IDs.

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.
   - `setenforce Enforcing`
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.

Monitor and manage Linux Processes

Process Inspection Using ps Command

- **HTTP Process Inspection**
  - `ps aux | grep http` Searches for processes related to HTTP from the list of all processes.
- **General Process Inspection**
  - `ps aux` Provides detailed information about most processes currently running on the system. It displays owner, CPU usage, memory usage, and the command itself.
  - `ps -elf` Displays a long-format listing of all processes, including their parent process IDs (PPID).
  - `ps -eo pid,ppid,uid,cputime,pmem,cmd` Offers a customized output for the `ps` command, showing process ID, parent process ID, user ID, CPU time, percentage of memory used, and the command itself.
  - `ps -fU ahmed` Displays full-format listing of all processes specifically owned by the user "ahmed".
- **Tree View & Paging**
  - `ps fax | less` Displays a hierarchical view (similar to a tree structure) of processes and their child processes. The `less` command is used for paging through the list.
- **Quick Analysis**
  - `ps aux | head` Outputs the first ten lines from the `ps aux` command. This usually includes the column headers and the first nine processes.
  - `ps aux | wc` Counts the number of lines, words, and characters in the output of `ps aux`. This essentially gives an idea of the number of processes running.

Backgrounding Tasks

- `sleep 1000` Pauses the shell or script's execution for 1000 seconds.
- `sleep 1000&` Starts the `sleep 1000` command in the background.
- `jobs` Displays the status of jobs in the current session.
- `bg` Resumes the last job that was stopped and runs it in the background.
- `fg` Brings the most recent background job to the foreground.
- `fg 1` Brings the job with job number 1 to the foreground.
- `ps -p 13732` or `ps -F 13732` Displays information about the process with the process ID (PID) 13732.

Query Processes

- `pgrep sleep` Searches for processes named "sleep" and prints their process IDs.
- `ps p $(pgrep sleep)` Uses the process IDs found by `pgrep sleep` to display detailed information about each "sleep" process.
- `pgrep --count sleep` Returns the number of "sleep" processes currently running.
- **List all** internet **and network files in use by all processes:** `lsof -i`
- **List** processes **listening on specific ports:** `lsof -i :22`

Terminate Processes

- Soft Kill `kill 8537, kill -15 8537, kill -term 8537, kill -sigterm 8537.`
- Hard Kill `kill -9 8537, kill -kill 8537, kill -sigkill 8537.`
- `pkill sleep` Sends the `TERM` signal to all "sleep" processes, asking them to terminate gracefully.

- `pkill -KILL -u user` Sends the `KILL` signal to all processes owned by the user named "user", forcefully terminating them.
- `killall sshd` Sends the `TERM` signal to all processes named "sshd", asking them to terminate gracefully.

Monitoring using top and htop

- `top` A monitoring tool that monitors active processes in real time, sorting them based on processor utilization. Can be customized with various keys
- `f` Add fields like PPID; `z` Color results; `h` More help.
- `k` Kill process; `i` Show only active processes; `r` Renice; `q` Quit.
- `1` Displays detailed information for each CPU.
- `dnf install htop` Installs htop (epel must be installed first).

Setting Process Priority with nice

- **Commands**
- `nice -n 5 dd if=/dev/zero of=/dev/null &` This command starts a `dd` process with a `nice` value of 5, meaning it has a lower priority than the default processes.
- `nice -n -5 dd if=/dev/zero of=/dev/null &` This starts the same `dd` process, but this time with a `nice` value of -5, giving it a higher priority.
- `renice -n 10 -p 14721` This command is used to change the `nice` value of a running process. In this case, it changes the `nice` value of the process with the PID (Process ID) 14721 to 10.

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.

1. **Checking `tuned` Service Status** `systemctl status tuned.service`
2. **Installing `tuned`** `yum install tuned`
3. **Enabling and Starting `tuned` Service** `systemctl enable --now tuned`

Using tuned-adm for Tuning Management

1. **Viewing Active Tuning Profile** `tuned-adm active`
2. **Listing Available Tuning Profiles** `tuned-adm list`
3. **Applying a Tuning Profile** `tuned-adm profile throughput-performance`
4. **Getting Recommended Profile** `tuned-adm recommend`
5. **Turning Off Dynamic Tuning** `tuned-adm off`

Control services and daemons

View and Query Services

- `systemctl -t help` Shows the available units.
- `systemctl list-units` Shows all loaded units.
- `systemctl list-unit-files --type=service` identifying which services are set to start automatically on boot and which are not.
- `systemctl status sshd.service` Check service state.
- `systemctl --type=service` Check all services.
- `systemctl --type=service | grep active | wc -l` Show total number of active processes.
- `systemctl is-active sshd` Check if sshd is active.
- `systemctl is-enabled sshd Check` if sshd is enabled.
- `systemctl --failed --type=service` List failed services.

Start, Stop, Reload Services

- `systemctl stop sshd.service` Stop sshd service.
- `systemctl start sshd.service` Start sshd service.
- `systemctl restart sshd.service` Restart sshd service.
- `systemctl reload sshd.service` Reload sshd service.
- `systemctl reload-or-restart sshd.service` Reload if available or restart sshd service.

View Dependencies

- `systemctl list-dependencies sshd.service` Display dependencies hierarchy.

Masking Services

- `systemctl mask name.service` Mask service to prevent it from being started.
- `systemctl unmask name.service` Unmask a service.
- `systemctl list-unit-files --state=masked` Get all masked services.

Configure and secure SSH

1. Identify the Package Providing sshd

- `dnf whatprovides */sshd`

2. Install OpenSSH

- `dnf install openssh`

3. Edit SSH Configuration to use port 1414

- `vi /etc/ssh/sshd_config` Modify it to work through port 1414.

4. Update Firewall Rules

- `firewall-cmd --zone=public --add-port=1414/tcp --permanent` Allow port 1414.
- `firewall-cmd --reload` Reload firewall rules.

5. Configure SELinux for Port 1414 (if enabled)

- `semanage port -a -t ssh_port_t -p tcp 1414` Allow 1414 port from SELinux.

6. Enable and Start SSHD

- `systemctl enable sshd; systemctl start sshd`

7. Verify SELinux Configuration

- `semanage port -l | grep sshd`

Configure SSH Keys

1. Generate and secure SSH Key Pair on Remote Server

- `ssh-keygen`
- `chmod 700 ~/.ssh`
- `chmod 600 ~/.ssh/authorized_keys`

2. View Keys

- `cat /home/ahmed/.ssh/id_rsa` Contains private key.
- `cat /home/ahmed/.ssh/id_rsa.pub` Contains public key.

3. Copy Public Key to Target Host

- `ssh-copy-id -i ~/.ssh/id_rsa.pub user@host`

4. Restart SSHD

- `sudo systemctl restart sshd`
- `ssh user@host` Test the connection.

5. Reload SSHD

- `systemctl reload sshd`

Installing and Updating Software in Red Hat

Package Management Commands:

- `dnf install` **or** `dnf remove`
- `dnf update` Updates all packages to the latest version available.
- `dnf check-update` Checks for available updates for installed packages without installing them.
- `dnf download --resolve <package>` Downloads a package and its dependencies without installing them.
- `sudo dnf localinstall *.rpm` Installs packages from downloaded .rpm files and resolves dependencies.

Repository Management:

- `dnf repolist all` Lists all enabled and disabled repositories.

Listing and Searching Packages:

1. `dnf list`
   - Lists installed, available, and all packages based on different filters.
     - `dnf list installed` Lists installed packages.
     - `dnf list available` Lists available packages in the repositories.
     - `dnf list | wc -l` Counts the number of packages listed by dnf list.
     - `dnf list 'http*'`: Lists all packages starting with "http".
2. `dnf list kernel` Lists kernel-related packages.
3. `dnf search all 'web server'` Searches for packages related to web servers.
4. `dnf info <package>` Provides detailed information about the specified package. (e.g., dnf info httpd)
5. `dnf provides <file>` Shows which package provides a specified file or directory. (e.g., `dnf provides /var/www/html`)

Group Management:

1. `dnf group <command>`
   - List groups `dnf group list`
   - Get group details `dnf group info "Virtualization Host"`
   - Install group with optional packages `dnf group install --with-optional "Minimal Install"`
   - Remove group with optional packages `dnf group remove --with-optional "Server with GUI"`

Logs and History:

`tail -5 /var/log/dnf.rpm.log` Displays the last 5 lines of the DNF RPM log file.

`dnf history` Shows the package installation/removal history.

`dnf history info <id>` Shows detailed information about a specific transaction in the history.

`dnf history undo <id>` Reverts changes made in a specific transaction.

Linux System Logs Monitor Guide

Overview Log files on Linux servers provide a detailed record of system activities. These logs aid in troubleshooting, monitoring, and security evaluations. To get the most out of these logs, they need to be managed and analyzed effectively.

Common Linux Log File Locations and Descriptions

System Logs

1. **`/var/log/messages`** Contains general system messages, including startup messages. It's one of the primary logs administrators check when troubleshooting.
2. **`/var/log/boot.log`** Logs related to the system booting process.
3. **`/var/log/kern.log`** Logs from the Linux kernel. Useful for troubleshooting hardware and kernel-specific issues.
4. **`/var/log/secure`** (or **`/var/log/auth.log`**) Authentication-related logs, recording user authentications, attempted logins, and other security-related events.
5. **`/var/log/utmp`** or **`/var/log/wtmp`** These are not really log files. They store information about who is currently logged in and login history. The who, w, last and lastb commands use these files.
6. **`/var/log/cron.log`** Logs from the cron daemon, showing the execution of scheduled tasks.

Application and Service Logs

7. **`/var/log/maillog`** Logs from mail servers like Sendmail or Postfix. Useful for troubleshooting email delivery issues.
8. **`/var/log/qmail/`** Directory containing logs specifically for the qmail mail server.
9. **`/var/log/httpd/`** Contains log files for the Apache HTTP server, including access.log (recording all requests processed by the server) and error.log (recording errors).
10. **`/var/log/lighttpd/`** Directory with logs for the Lighttpd web server, structured similarly to Apache's logs.
11. **`/var/log/mysqld.log`** Log file for the MySQL database server. Contains database server-related messages, including errors, warnings, and other diagnostic info.
12. **`/var/log/yum.log`** Log for the yum package manager, recording package installations, updates, and removals.

Understanding rsyslogd Configurations

`rsyslogd` provides flexible logging configurations that can be tailored based on

- **Facilities** Categories of information, e.g., system, security, mail.
- **Priorities** Severity levels such as emergency, error, warning, etc.
- **Destinations** Where the logs will be written to.

Example Configuration Edit **`/etc/rsyslog.conf`** to

#log all warning messages to warning file

```
*.warn    /var/log/warnings
*.err     /var/log/errors
systemctl restart rsyslog
```

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`
   - `journalctl -e` **# View the end of the log**
   - **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`

Persisting journal:

Default Storage Behavior

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

  `mkdir -p /var/log/journal`

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

  `systemctl restart systemd-journald.service`

Scheduling jobs

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**
  - **System-wide cron jobs:**Edit the main cron configuration **vim /etc/crontab** (including user)
  - *** * * * * root rm -rf /tmp/testdir/***
  - **17 * * * * * ahmed rm -rf /home/ahmed/Desktop/***
  - **User-specific cron jobs:** 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.
- **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.

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`

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/`

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.

Archiving and Transferring Files Using `tar` and Compression Tools

- **Archiving**
  - To archive `/home` directory into a file named home.tar `tar cvf home.tar /home/`
  - For the `/etc` directory into `etc.tar tar cvf etc.tar /etc/`
- **Checking Archive Type** `file home.tar`
- **Listing Archive Contents**
  - For `home.tar tar tf home.tar`
  - To find specific files (e.g., rsyslog in etc.tar) `tar -tf etc.tar | grep rsyslog`
- **Extraction**
  - To extract in the current location `tar xvf home.tar`
  - To extract to a different location like `/mnt/ tar xvf home.tar -C /mnt/`
  - For extracting with original permissions `tar xpvf home.tar -C /mnt/`
- **Compression with tar**
  - Using **gzip** `tar zcvf /lab01/home.gz /home/`
    - Extraction `tar zxvf home.gz -C /save`
  - Using **bzip2** `tar jcvf /lab01/home.tar.bz2 /home/`
    - Extraction `tar jxvf home.bz2 -C /save`
  - Using **xz** `tar Jcvf /lab01/home.xz /home/`
    - Extraction `tar Jxvf home.xz -C /save`
- **Backup and Restore**
  - For backup `tar zcvf home.tar.gz /home`
  - For restoration `tar zxvf home.tar.gz -C /`

Compression Using gzip

- **Compression** `gzip arch.tar`
- **Decompression** `gunzip arch*`
- **Reading Compressed Files** Use `zcat` or `zless`

Compression Using bzip2

- First, ensure the necessary software is installed `yum install bzip*`
- **Compression** `bzip2 file`
- **Decompression** `bunzip2 file*`

Compression Using xz

- **Compression**
  - For a single file `xz passwd`
  - While retaining the original `xz -k passwd`
  - For multiple files `xz f1.txt f2.txt f3.txt`
- **Decompression**
  - Basic decompression `xz -d passwd.xz`
- While retaining the .xz original `unxz -k passwd.xz`
- **Inspection and Listing**
  - Compression information `xz -l passwd.xz`
  - View contents without decompression `xzcat passwd.xz`

Using find Command

1. **Search by Name Pattern**
   - **Example** `find / -name file*`
   - **Explanation** Searches for all files starting with "file" in the root directory.
2. **Search by Size**
   - **Example** `find / -size +1G`
   - **Explanation** Searches for all files larger than 1 GB in the root directory.
3. **Search by Permissions**
   - **Example** `find -type f -perm 644`
   - **Explanation** Searches for all files with permission 644.
4. **Search by Type (e.g., Empty Directories)**
   - **Example** `find /tmp -type d -empty`
   - **Explanation** Searches for all empty directories in the /tmp directory.
5. **Search by Owner or Group**
   - **Example** `find /tmp/ -user root`
   - **Explanation** Searches for all files owned by the "root" user in /tmp.
6. **Copy Files Based on Search**
   - **Example** `find /usr/share/doc -name *.html -exec cp {} . \;`
   - **Explanation** Finds all html files in /usr/share/doc and copies them to the current directory.
7. **Change Permissions of Specific Files**
   - **Example** `find /root -type f -perm 0777 -exec chmod 500 {} \;`
   - **Explanation** Searches for all regular files under "/root" with permission 0777 and changes their permissions to 500.
8. **Find Files Without Specific Permissions**
   - **Example** `find / -type f ! -perm 777`
   - **Explanation** Finds files without 777 permissions in the root directory.
9. **Find and Delete Specific Files**
   - **Example** `find / -type f -name *.mp3 -size +10M -exec rm {} \; or find / -type f -name *.mp3 -size +10M -delete`
   - **Explanation** Finds and deletes all MP3 files larger than 10 MB in the root directory.
10. **Identifying Recently Accessed and Modified Files**

- `find / -mtime 1` Searches the entire filesystem for files modified exactly 24 hours (1 day) ago.
- `find / -mtime -1` Finds files modified within the last 24 hours from the entire filesystem.
- `find / -atime 5` Searches for files last accessed exactly 5 days ago throughout the entire filesystem.
- `find / -atime -5 2>/dev/null` Finds files accessed within the last 5 days, suppressing error messages.
- `find / -amin 10 2>/dev/null` Searches for files last accessed exactly 10 minutes ago, hiding errors.
- `find / -mmin -10` Finds files modified within the last 10 minutes across the filesystem.
- `find / -mmin +10` Searches the entire filesystem for files that were last modified more than 10 minutes ago.

Managing Red Hat Enterprise Linux Networking

Basic IP Commands

- **Display Interface Details** `ip a show eth0` or `ip a s enos3; ip -6 a`
- **Turn Interface Off** `ip link set eth0 down` Turns the `eth0` interface off.
- **Add Temporary IP Address** `ip addr add 192.168.1.100/24 dev eth0` Assign a temporary IP address to `eth0`.
- **Show Routing Table** `ip r show`
- **Add Temporary Default Gateway** `ip route add 172.16.1.0/24 via 192.168.1.1`
- **ARP Table / Neighbors** `ip n show`
- **Display Specific Interface Information** `ip -4 a s em1` Displays information about the `em1` interface for IPv4.
- **Routing Table for IPv4/IPv6** `ip r s` and `ip -6 r`
- **Trace Network Paths** `tracepath www.google.com` and `tracepath6 www.google.com`

Network Diagnostics

- **Packet Capture and Analysis** `tcpdump` .
- **Network Statistics - Netstat** `netstat -a | more` Listens to all TCP and UDP ports.
  `netstat -tupln, ss -tunap ,netstat -s, netstat -r`
  - `dnf install bind-utils`
  - `dig yahoo.com; dig mx yahoo.com; dig ns yahoo.com; dig txt yahoo.com`
  - `host -t MX p yahoo.com`
  - `dig @1.1.1.1 yahoo.com`
  - `tcpdump -i eth0 port 22`
  - `tracepath www.google.com`
  - `ping -I eth0 www.yahoo.com`
  - `nmap -p 80 192.168.1.1 ; nmap -p 22,80,443 192.168.1.1; nmap 192.68.244.1-100`

Configuration Files

1. **Global Network Configuration** `/etc/sysconfig/network`
- **The network configuration files used to be in** `/etc/sysconfig/network-scripts/ifcfg-ethx` **but have moved to** `/etc/NetworkManager/system-connections/.`
2. **Hostname** `/etc/hostname`
3. **Name Server Configuration** `/etc/resolve.conf`
4. **Static Table Lookup for Hostnames** `/etc/hosts`

```
[ipv4]
address1=192.168.244.222/24,192.168.244.2
dns=1.1.1.1;
method=manual
```

Persistently Storing Network Configurations

Edit the interface configuration using a text editor like vim

- `vim /etc/NetworkManager/system-connections/`

After making changes, restart the Network Manager `systemctl restart NetworkManager`

curl and ping Commands

- **Download Files** `wget [fileurl]`

- **Check Website Accessibility** `curl www.google.com` If it returns the website's content, the site is accessible.
- **Download Using** `curl curl -O www.site/filename` This can replace `wget` if it's not supported in your system.
- **Ping a Website** `ping www.google.com` For a limited number of pings, use
  `ping -c 5 www.google.com; Ping ping6 www.google.com`

Network Management with nmcli and nmtui

- **Show Devices** `nmcli d s` or `ip link`
- **Show Connections** `nmcli connection show`
- **Show Active Connections** `nmcli connection show --active`
- **Device Status** `nmcli device status`
- **Add a Connection to NIC (Static)**
  ```
  nmcli connection add con-name newcon1 ifname ens224 ipv4.addresses 192.168.219.140/24 ipv4.gateway 192.168.219.2 ipv4.dns 1.1.1.1 ipv4.method manual type ethernet
  ```
- **Modify an Existing Connection**
  ```
  nmcli connection modify "ens160" ipv4.addresses "192.168.219.111/24" ipv4.gateway "192.168.219.2" ipv4.dns "1.1.1.1" ipv4.dns-search "search.abdelwahed.me" ipv4.method manual autoconnect yes
  ```
- **Add extra connection to the same NIC (DHCP)**
  ```
  nmcli con add type ethernet con-name newcon2 ifname ens224 ipv4.method auto
  ```
- **Reload Connection** `nmcli connection reload`
- **Activate a Connection** `nmcli connection up "newcon1"`
- **Deactivate a Connection** `nmcli connection down newcon1`
- **Disable NIC** `nmcli device disconnect eth0`
- **Enable NIC** `nmcli device connect eth0`
- **Graphical Network Configuration Tool** `nmtui`

Managing Firewall with Firewalld on Linux

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.

Container

Basic Concept of Containers

- **Containers** are a way to package applications and their dependencies into isolated, self-contained environments. This allows applications to run consistently across different systems without conflicts.
- **Lightweight**: Unlike virtual machines, which contain a full operating system, containers only package the application and its dependencies. They share the host system's OS kernel, making them much smaller and faster to start.
- **Portable**: Containers can run the same way across various environments (developer laptops, data centers, cloud) because they package everything the application needs. This makes it easy to move applications between development, testing, and production without changes.
- **Isolated**: Each container runs in its own environment, isolated from other containers and the host system. This protects applications from interfering with each other, enhances security, and makes it easier to manage dependencies.

Podman: Daemonless, Rootless Container Engine

- **Daemonless**:
    - Traditional container engines like Docker use a background service (daemon) to manage containers. The Docker daemon runs with elevated privileges and handles tasks like starting, stopping, and monitoring containers.
    - **Podman**, on the other hand, is **daemonless**. It doesn't rely on a background service; instead, each command directly manages containers. This design makes Podman simpler and avoids the single point of failure associated with a daemon.
- **Rootless (No Root Privileges Required)**:
    - Most container engines require **root privileges** to create and manage containers, which can introduce security risks if a container is compromised.
    - Podman allows containers to run without root access (rootless mode), making it more secure. In rootless mode, users can create and manage containers within their own user namespace, isolating them from the host system.
    - This feature is especially valuable for multi-user environments where unprivileged users need to run containers safely without requiring administrative rights.

Step-by-Step Lab: Running a Container

1. Install Podman

- Start by updating the system and installing Podman:

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

- Verify the installation to ensure Podman was installed correctly:

```
podman --version
```

2. Pull the Alpine Image

- Download the **Alpine Linux** image to use as a lightweight base:

```
podman pull alpine
```

3. Run the Container with Custom CPU and Memory Limits

- Start the container with specific CPU and memory limitations. In this example, we limit the container to use 1.5 CPU cores and 512 MB of RAM.

```
podman run -d --name limited_container \
  --cpus=1.5 \
  --memory=512m \
  alpine sleep infinity
```

- **Explanation**:
  - `--cpus=1.5`: Restricts the container to using a maximum of 1.5 CPU cores.
  - `--memory=512m`: Limits the container's memory usage to 512 MB.
  - `alpine sleep infinity`: Runs the Alpine container with a command to keep it running indefinitely.

4. Verify the Container's Configuration

- **Check Container Status**:
  - List running containers to confirm that your container is up and running:

```
podman ps
```

- **Inspect Resource Limits**:
  - Use the inspect command to check that the CPU and memory limits are applied:

```
podman inspect limited_container | grep -E 'Cpu|Memory'
```

- **Monitor Resource Usage**:
  - View real-time resource usage with podman stats:

```
podman stats limited_container
```

5. Interact with the Running Container

- You can execute commands within the running container to verify its environment and observe resource usage:

```
podman exec -it limited_container sh
```

- Once inside the container, you can run simple commands like:

```
top       # To view CPU and memory usage within the container
exit      # To leave the container shell
uname -a          # Displays kernel and system information
cat /proc/meminfo     # Shows detailed memory info
```

```
cat /proc/cpuinfo     # Shows CPU details
yes > /dev/null &
pkill yes
```

To stop and remove an Alpine container in Podman (or Docker), follow these steps:

Step 1: List Running Containers

- First, check the list of running containers to find the container name or ID of your Alpine container:
  `podman ps`
- If the container is not currently running but was created, you can list all containers (including stopped ones) with:
  `podman ps -a`

Step 2: Stop the Container

- Use the container name or ID from the previous step to stop the Alpine container:
  `podman stop <container_name_or_id>`
- Example:
  `podman stop my_alpine_container`

Step 3: Remove the Container

- After stopping it, remove the Alpine container using its name or ID:
  `podman rm <container_name_or_id>`
- Example:
  `podman rm my_alpine_container`

Alternative: Stop and Remove in One Step

- You can also stop and remove the container in a single command:
  `podman rm -f <container_name_or_id>`
- The `-f` flag forces the container to stop if it's running and then removes it.