Al Razzaq Program Part II

Notes by Al Nafi

Red Hat System Administration I

Author:

Osama Anwer Qazi

Get started with Red Hat Enterprise Linux

This section introduces the foundational concepts and skills required to work effectively with Red Hat Enterprise Linux (RHEL). It builds on general computing knowledge (e.g., familiarity with operating systems and networking) and sets the stage for more advanced topics—such as security hardening, containerization, and automation—that follow later in the series. Throughout these notes, real-world examples and command-line snippets are provided to clarify each topic.

Describe and define open source, Linux, Linux distributions, and Red Hat Enterprise Linux

- **Open Source**
    - Software whose source code is freely available for inspection, modification, and redistribution.
    - Governed by licenses (e.g., GPL, MIT, Apache) that ensure end users' rights to study, change, and share the software.
    - Community-driven development model: anyone can contribute fixes, enhancements, or new features.
    - Offers transparency (security patches visible), collaboration (communities of contributors), and cost savings (no licensing fees).
- **Linux**
    - A UNIX-like, multitasking operating system kernel originally created by Linus Torvalds in 1991.
    - Provides core functionality—process scheduling, memory management, device drivers, filesystem support, networking stack—upon which full operating systems (distributions) are built.
    - Modular architecture: the Linux kernel interacts with userland utilities (GNU tools, system libraries).
    - Widely used in servers, desktops, embedded devices, and cloud environments because of stability, flexibility, and strong community support.
- **Linux Distributions**
    - A "distribution" (distro) packages the Linux kernel with userland tools, system utilities, package managers, and default configurations to serve particular use cases (e.g., servers, desktops, embedded systems).
    - Examples:

        - **Debian/Ubuntu** family (APT package management) – popular for desktops and servers.
        - **Fedora** (DNF package management) – community-driven, upstream for Red Hat technologies.
        - **Red Hat Enterprise Linux (RHEL)** (YUM/DNF) – commercially supported, focusing on enterprise stability.
        - **CentOS Stream** – rolling-release model upstream of RHEL.
        - **SUSE Linux Enterprise Server (SLES)** (ZYPP package management) – enterprise distribution with its own system management tools.
    - Core differences across distros: package format (RPM vs. DEB), default init system (systemd vs. alternatives), release cadence (rolling vs. fixed), and support lifecycle (community vs. commercial).

- **Red Hat Enterprise Linux (RHEL)**
    - A commercially supported Linux distribution developed and maintained by Red Hat, Inc.
    - Focused on enterprise environments: long-term support (LTSS), certified hardware/software stacks, security updates, and compliance (e.g., FIPS, Common Criteria).
    - Uses the RPM package format and DNF/YUM as its package manager.
    - Release model: Major versions (e.g., RHEL 8, RHEL 9) followed by minor point releases (e.g., 8.1, 8.2), each supported for up to 10 years.
    - Subscription model: organizations pay for support, access to vetted repositories, and Red Hat's knowledge base.
    - Key differentiators:
        - Enterprise certification (validated on a wide range of enterprise hardware).
        - Strict Backport policy (features may not change dramatically within a major release—security fixes backported to maintain API/ABI stability).
        - Integration with Red Hat's ecosystem (OpenShift, Ansible, Satellite, Insight).

- **Continuity and Prerequisites**
    - Before diving into RHEL, familiarity with general operating system concepts (processes, memory, filesystems) and basic networking is beneficial.
    - This foundational understanding aligns with preceding topics (e.g., general Linux theory) and prepares for upcoming modules on user management, security, and

services.

Access the command line

Log into a Linux system and run simple commands using the shell.

- **What Is the Command Line?**
  - A text-based interface (also called a shell) for interacting with the operating system.
  - Default shell in RHEL: `bash` (Bourne Again SHell), though alternatives (e.g., `zsh`, `fish`) can be installed.
  - Provides direct access to system utilities, scripting, and automation.
- **Logging In to RHEL**
  - **Local Console Login**
    - Power on or reboot the RHEL system.
    - At the graphical login manager (if installed), select a user, enter a password, and click “Sign In.”
    - To drop to a text console instead of the graphical environment, press `Ctrl+Alt+F3` (through `F6` for virtual terminals).
    - At the text prompt, enter your username, press `Enter`, then enter your password to log in.
  - **Remote SSH Login**

Ensure the RHEL system’s SSH daemon (`sshd`) is running and reachable (default TCP port 22).

On the RHEL host:

sudo systemctl status sshd.service

-
- From another Linux/macOS/Windows system, open a terminal or SSH client (e.g., PuTTY).

Run:

ssh username@rhel-server.example.com

- Accept the host key fingerprint when prompted (first-time only), then enter your password.
- Successful login displays a shell prompt (often ending with `$` for regular users, `#` for root).

- **Basic Shell Commands**

`pwd` (print working directory): Shows the full path of the current directory.

```
$ pwd
/home/engineer
```

- **`ls` (list directory contents)**: Displays files and subdirectories in the current directory.
  - Common options:
    - `ls -l` (long listing, with permissions, owner, size, timestamp).
    - `ls -a` (include hidden files, i.e., those beginning with a dot).

```
$ ls -lah /var/log
drwxr-xr-x. 10 root root 4.0K May 15 10:12 .
-rw-r--r--.  1 root root  618 May 15 23:00 anaconda.log
```

`man` (manual pages): Display documentation for commands and system calls.

```
$ man ls
```

`--help` flag: Many commands provide a brief usage summary.

```
$ cp --help
```

- **`exit` or `Ctrl+D`**: Leaves the current shell session.

- **Prompt Customization (Brief Overview)**
  - Environment variable `PS1` defines the interactive prompt format.

Example:

Show default PS1

$ echo "$PS1"

\u@\h \W\$

\u = username, \h = hostname, \W = current directory basename, \$ = # if root or $ otherwise

-
- Can be customized in `~/.bashrc` (more advanced).

- **Continuity and Dependencies**
  - Mastery of basic login and shell navigation is essential before performing file management, user administration, and system configuration tasks covered in later sections.
  - Subsequent topics will assume comfort with connecting remotely (SSH) and operating within a bash shell.

Manage files from the command line

Copy, move, create, delete, and organize files while working from the bash shell.

- **Directory and File Basics**
  - In Linux, everything is a file (e.g., regular files, directories, devices, sockets).
  - Path types:
    - **Absolute path**: starts at root (`/`), e.g., `/etc/passwd`.
    - **Relative path**: relative to the current directory, e.g., `./reports/log.txt` or `../backup/`.
- **Creating Directories and Files**

`mkdir` (make directory)

Create a single directory

$ mkdir project

Create nested directories (parent and child)

$ mkdir -p project/src/bin

-

`touch` (create empty file or update timestamp)

$ touch project/README.md

-

`echo` and redirection (`>` or `>>`)

Create a file containing “Hello, RHEL!”

$ echo "Hello, RHEL!" > project/welcome.txt

Append another line

$ echo "Second line" >> project/welcome.txt

-

- **Listing and Examining Files**
  - **`ls` variants**
    - `ls -l`: long format, shows file permissions, owner, group, size, modification date.
    - `ls -lh`: human-readable sizes (e.g., `4.0K`, `1.2M`).
    - `ls -R`: recursively lists subdirectories.

`stat`: detailed metadata for a file.

$ stat project/welcome.txt

-

- **Copy, Move, and Rename**
  - **`cp` (copy files or directories)**

Simple file copy:

$ cp project/welcome.txt project/welcome_backup.txt

-

Preserve attributes (timestamps, permissions):

$ cp -p project/welcome.txt /backup/

-

Copy directories recursively:

$ cp -r project/ project_backup/

- **`mv` (move files, rename if same directory)**

Rename a file

$ mv project/welcome.txt project/intro.txt

Move file to a different directory

$ mv project/intro.txt /home/engineer/old_imports/

- **Deleting Files and Directories**
  - **`rm` (remove)**

Remove a single file:

$ rm project/old_file.txt

Remove directory and contents recursively (use with caution):

$ rm -r project_backup/

Force deletion (ignore nonexistent files, no prompts):

$ rm -rf /tmp/temp_dir/

`rmdir`: remove empty directories only.

$ rmdir project/src/temp

- **Viewing File Contents**

`cat` (concatenate and display)

$ cat project/intro.txt

`less` / `more` (paginated viewing)

$ less /var/log/messages

-
  - Navigate via arrows, `q` to quit.
- **`head` / `tail`**
  - Show first N lines: `head -n 10 file.txt`
  - Show last N lines: `tail -n 20 file.txt`

Follow live output (e.g., log tailing):

$ tail -f /var/log/secure

-

- **File Organization Best Practices**
  - Use meaningful directory names (e.g., `~/projects/2025/clientA/`).
  - Separate user data (`/home/username/`) from system configuration (`/etc/`).
  - Avoid creating files in root directory (`/`).
  - Leverage standard Linux hierarchy (Filesystem Hierarchy Standard, FHS) to know where to place custom scripts (`/usr/local/bin/`), logs (`/var/log/`), and temporary files (`/tmp/`).
- **Continuity and Dependencies**
  - These file-management commands are fundamental for almost every administrative task: editing configuration files, backing up logs, or moving scripts into executable paths.
  - Later topics—such as configuring services, managing users, and securing files—assume familiarity with these commands.

Get help in Red Hat Enterprise Linux

Resolve problems by using local help systems.

- **Why Getting Help Locally Matters**

- RHEL systems may not always have internet access (e.g., air-gapped environments, restricted corporate networks).
- Becoming familiar with built-in documentation and tools (man pages, info pages, local knowledge base) enables troubleshooting without external resources.

- **Manual Pages (`man`)**

Each command or system call typically has a manual page:

$ man ls

-
- Sections of `man` pages:
  - **Name** (command name and short description),
  - **Synopsis** (usage syntax),
  - **Description** (detailed explanation),
  - **Options** (flags and parameters),
  - **Examples** (if provided),
  - **See Also** (related commands).
- Navigate using arrow keys, `/search_term` to search within the page, `q` to quit.

- **Info Pages (`info`)**

Some packages provide more extensive documentation via GNU Info.

$ info coreutils

-
- Offers hierarchical navigation:
  - Use arrow keys (or `n`/`p`) to move between nodes, `u` to go up a level, and `q` to exit.

- **`--help` and `-h` Flags**

Many utilities provide a quick usage summary:

$ grep --help

Usage: grep [OPTION]… PATTERN [FILE]…

-

- Option lists typically appear right in the terminal—ideal for a quick syntax refresher.

- **Local Knowledge Base and Documentation Files**

`/usr/share/doc/`: contains package-specific documentation (README, examples, change logs).

$ ls /usr/share/doc/httpd-2.4.37/

AUTHORS ChangeLog COPYING INSTALL LICENSE README

RHEL-specific release notes:

$ less /usr/share/doc/redhat-release

- `/etc/` **directory** often contains sample configuration files with comments explaining each setting (e.g., `/etc/sshd_config`).

- `help` **Built-in for Shell Built-ins**

The shell itself (bash) has built-in commands that may not have separate man pages.

$ help cd

- Lists syntax, options, and a short description.

- `apropos` **and** `whatis`

`whatis <keyword>`: brief description of commands matching `<keyword>`.

$ whatis useradd

useradd (8) - create a new user or update default new user information

`apropos <keyword>`: search man page descriptions for `<keyword>`.

$ apropos network

network (7) - overview of network protocols

networkctl (1) - query/control network stati

- **`yum`/`dnf` Info Commands**

For package-related help:

$ yum info httpd

Name : httpd

Version : 2.4.37

Release : 21.el8

Summary : Apache HTTP Server

…

$ dnf search postgresql

  -

- **Log Files for Troubleshooting**
  - Many problems manifest in log entries. Reviewing logs (see “Analyze and store logs” section) is a core part of self-help.

Commands:

$ less /var/log/messages

$ journalctl -xe

  -

- **Continuity and Dependencies**
  - Efficient use of `man`, `info`, and log inspection accelerates problem resolution in subsequent topics (e.g., service troubleshooting, SSH issues, network configuration).
  - Familiarity here reduces reliance on external references, which is important when implementing production-grade environments.

Create, view, and edit text files

Manage text files from command output or in a text editor.

- **Redirecting Command Output to Files**

Overwrite with `>`:

$ ps aux > processes.txt

-

Append with `>>`:

$ echo "New entry" >> processes.txt

-

Pipe to Filters and Save:

$ dmesg | grep usb > usb_events.log

-

Here Documents (`<<`) for multi-line input:

$ cat <<EOF > script.sh

#!/bin/bash

echo "Startup script"

EOF

-

- **Viewing Files (Review Section in “Manage files from the command line”)**
  - `cat`, `less`, `head`, `tail`.
  - When editing large configuration files (e.g., `/etc/ssh/sshd_config`), use `less` or `vim` (described next) rather than `cat`.
- **Editing Text Files**
  - **`vi` / `vim` (Vi Improved)**
    - Modes:
      - **Normal mode** (navigate, delete, yank, paste).
      - **Insert mode** (insert or modify text).
      - **Command mode** (execute ex commands, e.g., save, quit).
    - **Common Workflow**:
      - Open file: `vim /etc/hosts`

- Navigate to location (arrow keys or `h`/`j`/`k`/`l`).
- Enter Insert mode: press `i` (insert before cursor) or `a` (append after cursor).
- Edit content.
- Return to Normal mode: press `Esc`.
- Save and quit: type `:wq` then `Enter`.
- Quit without saving: `:q!` then `Enter`.

- **Useful Commands in Normal Mode**:
  - `dd` deletes the current line.
  - `yy` yanks (copies) the current line.
  - `p` pastes below the current line.
  - `/pattern` searches forward for "pattern."
  - `n` goes to the next match.

- **`nano` (simple terminal editor)**
  - Easier for beginners; some RHEL installations include it.
  - Open file: `nano /etc/resolv.conf`.
  - Shortcut hints at the bottom (e.g., `^O` = Save, `^X` = Exit).
- **`sed` and `awk` for Automated Editing**

`sed` (stream editor) to perform substitutions on files without opening an interactive editor:

```
# Replace "localhost" with "127.0.0.1" in resolv.conf
$ sed -i 's/localhost/127.0.0.1/' /etc/hosts
```

-

`awk` for pattern scanning and processing, often used for generating reports or extracting fields:

```
# List usernames from /etc/passwd
$ awk -F: '{print $1}' /etc/passwd
```

-

- **File Permissions and Ownership while Editing**

Some files require elevated privileges to edit (e.g., system configs).

$ sudo vim /etc/ssh/sshd_config

  -

After editing, verify ownership and permissions:

$ ls -l /etc/ssh/sshd_config

-rw-r--r--. 1 root root 3234 May 5 12:00 /etc/ssh/sshd_config

  -

- **Continuity and Dependencies**
  - Mastery of editing tools is necessary before customizing service configuration files (e.g., SSH, syslog), user properties (`/etc/passwd`, `/etc/group`), and network definitions (`/etc/sysconfig/network-scripts/`).
  - Later, automated tasks (with `sed`/`awk`) tie directly into automation modules where scripts programmatically modify configuration files.

Manage local users and groups

Create, manage, and delete local users and groups, as well as administer local password policies.

- **Overview of User and Group Concepts**
  - **Users**: Represent individual accounts—each has a username, user ID (UID), primary group, home directory, login shell, and password.
  - **Groups**: Collections of users. Each user belongs to a primary group and can be assigned to additional (secondary) groups. Permissions on files/directories can be granted to groups for easier access control.
- **Key Configuration Files**
  - `/etc/passwd`: Stores basic user account information (username, UID, GID, home directory, shell). Does not store passwords.
  - `/etc/shadow`: Stores encrypted password hashes, password aging information, failed login counts, and expiration data. Only readable by root.
  - `/etc/group`: Defines group names, GIDs, and members of each group.

- `/etc/gshadow`: Stores group passwords and group administrators.

- **Creating and Managing Users**
  - **`useradd`** (or `adduser`)

Create a new user with default settings:

Create user “alice”

$ sudo useradd alice

By default: home directory /home/alice created if /etc/default/useradd instructs; shell /bin/bash

-
- Common options:
  - `-m` or `--create-home`: explicitly create home directory.
  - `-d /path/to/home`: specify alternate home directory.
  - `-s /bin/bash`: specify login shell.
  - `-g <group>`: set primary group.
  - `-G <group1,group2>`: set supplementary groups.

Create user “bob” in primary group “developers”, supplementary groups “wheel” and “docker”, with custom shell

$ sudo useradd -m -s /bin/bash -g developers -G wheel,docker bob

-
- After creating the account, set or enforce a password.

  - **`passwd`** (change user password)

As root or with `sudo`:

$ sudo passwd alice

Changing password for user alice.

New password:

Retype new password:

passwd: all authentication tokens updated successfully.

- As a normal user: enforces password change for one's own account.

- **Account Expiration and Password Aging**

chage: modify password aging policies for a user:

```
$ sudo chage -l alice        # List password aging info
$ sudo chage -M 90 alice     # Max days before password change: 90
$ sudo chage -m 7 alice      # Min days between password changes: 7
$ sudo chage -W 7 alice      # Warn user 7 days before expiration
```

- **userdel**: remove a user account.

Remove only account:

```
$ sudo userdel alice
```

Remove account with home directory and mail spool:

```
$ sudo userdel -r alice
```

- **Managing Groups**

groupadd: create a new group.

```
$ sudo groupadd developers
```

groupmod: modify existing group (e.g., change GID or group name).

```
$ sudo groupmod -n devops developers    # Rename "developers" to "devops"
```

groupdel: delete a group (only if no users are members).

```
$ sudo groupdel devops
```

- **Adding/Removing Users to/from Groups**

`usermod`: modify a user's group membership.

Add "charlie" to supplementary group "docker"

$ sudo usermod -aG docker charlie

Remove "bob" from group "developers" (requires editing /etc/group or using gpasswd)

-

`gpasswd`: administer /etc/group entries directly.

$ sudo gpasswd -a bob docker # Add bob to docker group

$ sudo gpasswd -d bob docker # Remove bob from docker group

-

- **Local Password Policies**
  - **PAM (Pluggable Authentication Modules)**: framework controlling authentication, account, password, and session management.
    - Configuration files: `/etc/pam.d/system-auth` and `/etc/pam.d/password-auth`.

Policies often reference `/etc/security/pwquality.conf` for password strength (e.g., minimum length, required character classes, dictionary checks).

Example pwquality.conf settings:

minlen = 12

dcredit = -1 # Require at least one digit

ucredit = -1 # Require at least one uppercase letter

lcredit = -1 # Require at least one lowercase letter

ocredit = -1 # Require at least one special character

-

  - **Enforcing Password History**

Prevent reuse of recent passwords:

In /etc/security/pwquality.conf or /etc/pam.d/password-auth

remember = 5 # Disallow last 5 passwords

-

- **Account Lockout**

Prevent brute-force attacks by locking accounts after repeated failures:

Example in /etc/pam.d/password-auth:

auth required pam_faillock.so preauth silent deny=5 unlock_time=900

auth [default=die] pam_faillock.so authfail deny=5 unlock_time=900

- **Verifying Users and Groups**

`id <username>`: Display UID, primary GID, and supplementary groups.

$ id alice

uid=1001(alice) gid=1001(alice) groups=1001(alice),10(wheel),999(docker)

`getent passwd <username>`: Query user database (works with local, LDAP, or other NSS backends).

$ getent passwd bob

bob:x:1002:1002::/home/bob:/bin/bash

- `getent group <groupname>`: Query group database.

- **Continuity and Dependencies**
  - Effective user/group management underpins all system security practices: file permissions, sudo access, service ownership.
  - Later chapters (e.g., “Control access to files,” “Configure and secure SSH”) rely on well-configured user accounts and group memberships.

Control access to files

Set Linux file system permissions on files and interpret the security effects of different permission settings.

- **Understanding Linux File Permissions**

- Each file and directory has an owner (user), a group, and permission bits for three classes:
  - **User (u)**: the file owner
  - **Group (g)**: the group assigned to the file
  - **Others (o)**: all other users
- Permission types (three bits per class):
  - **Read (r)**: allowed to view file contents or list directory contents.
  - **Write (w)**: allowed to modify file contents or create/delete entries inside a directory.
  - **Execute (x)**: allowed to run a file as a program/script or traverse into a directory.

- **Viewing Permissions**

Use `ls -l`:

```
$ ls -l /home/alice/script.sh
-rwxr--r--. 1 alice developers 1.2K Jun  1 09:10 /home/alice/script.sh
```

- Breakdown:
  - `-rwxr--r--.`
    - Leading `-` indicates a regular file (`d` for directory, `l` for symlink).
    - `rwx`: user (owner) has read, write, execute.
    - `r--`: group has read only.
    - `r--`: others have read only.
    - Trailing `.` indicates SELinux context is present.
  - Owner: `alice` (user)
  - Group: `developers`

- **Modifying Permissions with `chmod`**

Symbolic notation:

Grant execute permission to owner, group, and others

$ chmod a+x /home/alice/script.sh

Remove write permission from group

$ chmod g-w /home/alice/script.sh

Set file to be readable and executable by all, writable only by owner

$ chmod u=rwx,g=rx,o=rx /home/alice/script.sh

- **Octal notation** (each digit groups u, g, o permissions):
  - `7` = `rwx` (4+2+1)
  - `6` = `rw-` (4+2+0)
  - `5` = `r-x` (4+0+1)
  - `4` = `r--` (4+0+0)
  - `0` = `---` (0+0+0)

Make script runnable by owner and group only:

$ chmod 750 /home/alice/script.sh

Owner: rwx (7), Group: r-x (5), Others: none (0)

- **Changing Ownership with `chown` and `chgrp`**

`chown`: change file owner and optionally group.

Make “bob” the owner and “admins” the group for a file

$ sudo chown bob:admins /var/log/custom.log

- Can use just user: `$ sudo chown bob /var/log/custom.log` (keeps existing group).

- **`chgrp`**: change only the group.

```bash

$ sudo chgrp developers /home/alice/project/
```

-
- **Special Permission Bits**
    - **SetUID (s on user execute bit)**: when set on an executable, allows users to run the file with the file owner's permissions (often root). Example: `chmod u+s /usr/bin/passwd`.

SetGID (s on group execute bit): when set on a directory, new files created within inherit the directory's group. When set on an executable, runs with group privileges.

SetGID on directory so new files inherit group "developers"

$ sudo chmod g+s /home/developers-shared/

    -

Sticky Bit (t on others execute bit): used on directories (e.g., `/tmp`). Only the file's owner (or root) can delete or rename files inside that directory, even if others have write permissions.

$ sudo chmod +t /home/public_drop/

    -
- **SELinux Contexts (Brief Overview)**
    - RHEL uses SELinux (Security-Enhanced Linux) by default in enforcing mode.
    - Every file/directory has an SELinux context (`user:role:type:level`). Permissions are checked at both POSIX (u/g/o bits) and SELinux policy levels.

View context:

$ ls -Z /etc/ssh/sshd_config

-rw-r--r--. root root system_u:object_r:sshd_config_t:s0 /etc/ssh/sshd_config

    -

Change context (for advanced scenarios):

$ sudo chcon -t httpd_sys_content_t /var/www/html/index.html

    -
    - Proper contexts are crucial for service functionality (e.g., web server cannot serve files labeled incorrectly).
- **Interpreting Security Effects**
    - **Loose Permissions**:

        - Files world-writable (`chmod 777`) can be modified by any local user—security risk.
        - Directories without the sticky bit in a public folder allow users to delete each other's files.
    - **Overly Restrictive Permissions**:
        - If configuration files are not readable by a service's account (e.g., `nginx` user can't read `/etc/nginx/nginx.conf`), the service fails to start.
        - SSH keys in `~/.ssh/authorized_keys` must be `600` or stricter; otherwise, SSH daemon refuses to use them.
- **Continuity and Dependencies**
    - Understanding file permissions is essential before configuring secure services (SSH, web server), setting up user directories, and preparing secure storage for logs or backups.
    - Subsequent modules—such as "Configure and secure SSH" and "Install and update software packages"—rely on correct file permission practices.

Monitor and manage Linux processes

Evaluate and control processes running on a Red Hat Enterprise Linux system.

- **What Is a Process?**
    - A process is an instance of a running program, identified by a PID (Process ID).
    - Processes run in the foreground (interacting with a terminal) or background (daemon services, batch jobs).
- **Viewing Running Processes**
    - **`ps` (process status)**

`ps aux` shows all processes for all users in a BSD-style listing.

```
$ ps aux | head -n 5
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 169024  6912 ?        Ss   May31   0:04 /usr/lib/systemd/systemd
root         2  0.0  0.0      0     0 ?        S    May31   0:00 [kthreadd]
```


-

`ps -ef` (UNIX-style full listing):

```
$ ps -ef | grep sshd
root      1120     1  0 Apr30   0:00 /usr/sbin/sshd -D
root     13521  1120  0 Jun01   0:00 sshd: alice [priv]
alice    13523 13521  0 Jun01   0:00 sshd: alice@pts/0
```

-

- **Interactive Process Monitoring**
  - **`top`**: real-time overview of processes, sorted by CPU usage by default.
    - Key functions:
      - `P`: sort by CPU usage.
      - `M`: sort by memory usage.
      - `k`: kill a process by PID.
      - `q`: quit.

`htop` (if installed): enhanced `top` with color, tree view, easier to navigate.

```
$ sudo dnf install -y htop
$ htop
```

-

- **Detailed Process Inspection**

`pstree`: shows processes in a tree hierarchy, making parent-child relationships clear.

```
$ pstree -p | less
init(1)─┬─NetworkManager(1003)─┬─dhclient(1021)
        │                      └─{NetworkManager}(1004)
        ├─sshd(1120)─┬─sshd(13521)─┬─bash(13523)─┬─top(13650)
        │            │             │             └─vim(13652)
```

- **Controlling Processes**

`kill <PID>` (default signal `TERM`) politely asks a process to terminate.

```
$ kill 13650
```

`kill -9 <PID>` (signal `KILL`) forces immediate termination (no cleanup). Use only when necessary.

```
$ kill -9 13650
```

`pkill <pattern>`: kill processes by name or other attributes.

```
$ sudo pkill httpd   # stops all Apache processes
```

`killall <process_name>`: kills all processes matching name.

```
$ sudo killall nginx
```

- **Foreground and Background Jobs**

Starting a Command in the Background (`&`)

```
$ long_running_task.sh &
```

  - The shell displays a job number (e.g., `[1] 14252`), where 14252 is the PID.

`jobs`: list background jobs started from the current shell.

```
$ jobs
[1]+  Running                 long_running_task.sh &
```

Bringing a Job to the Foreground (`fg`)

```
$ fg %1
```


Suspending a Job (`Ctrl+Z`): suspends, then can background with `bg`.

<Ctrl+Z>

[1]+ Stopped long_running_task.sh

$ bg %1

- **Adjusting Process Priority**
  - `nice`: start a process with a modified nice value (priority).
    - Default nice is 0; values range from -20 (highest priority) to +19 (lowest).

Start backup script with low priority

$ nice -n 10 ./backup.sh

`renice`: change the nice value of a running process.

Reduce priority (increase nice value)

$ sudo renice +5 -p 14252

- **System Resource Usage**

`free -h`: display total, used, and free memory in human-readable form.

$ free -h

	total	used	free	shared	buff/cache	available
Mem:	7.7G	1.2G	3.9G	45M	2.6G	6.1G
Swap:	2.0G	0B	2.0G			

`df -h`: disk filesystem usage (size, used, avail, mount point).

$ df -h

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/sda2	50G	15G	33G	32%	/
/dev/sda1	200M	45M	155M	23%	/boot

tmpfs 3.8G 0 3.8G 0% /dev/shm

- `du -sh <path>`: summarize disk usage of a directory or file.

$ du -sh /var/log

1.2G /var/log

- **Continuity and Dependencies**
  - Effective process management is critical for troubleshooting performance issues, ensuring essential services remain available, and performing graceful shutdowns—skills that feed directly into later sections on service control and systemd.

Control services and daemons

Control and monitor network services and system daemons with the systemd service.

- **Understanding systemd**
  - systemd is the init system and service manager on RHEL. It replaces traditional SysV init scripts with declarative unit files, parallel service startup, and advanced features (cgroups, dependency management).
  - A “daemon” is a background process that provides services (e.g., HTTP, database, logging). systemd refers to these collectively as services.
- **Key systemd Commands**
  - `systemctl`: primary tool to manage services and examine system state.

`systemctl status <service>`: show current status (active, inactive, failed) and recent logs.

$ sudo systemctl status sshd.service

● sshd.service - OpenSSH server daemon

Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled)

Active: active (running) since Mon 2025-06-01 13:02:15 UTC; 1 day ago

Process: 1120 ExecStartPre=/usr/sbin/sshd-keygen (code=exited, status=0/SUCCESS)

Main PID: 13521 (sshd)

- **systemctl start <service>**: start service immediately (does not enable persistence across reboots).

```
$ sudo systemctl start httpd.service
```

- **systemctl stop <service>**: stop a running service gracefully.

```
$ sudo systemctl stop httpd.service
```

- **systemctl restart <service>**: stop and then start again (useful after configuration changes).

```
$ sudo systemctl restart ssld.service
```

- **systemctl reload <service>**: instructs a daemon to reload configuration without full restart (if supported).

```
$ sudo systemctl reload nginx.service
```

- **systemctl enable <service>**: enable a service to start automatically at boot.

```
$ sudo systemctl enable firewalld.service
```

- **systemctl disable <service>**: prevent a service from starting at boot.

```
$ sudo systemctl disable postfix.service
```

- **systemctl is-enabled <service>**: check if a service is enabled.

```
$ systemctl is-enabled sshd.service
enabled
```

- **systemctl list-units --type=service --all**: list all services (active, inactive, failed).

```
$ systemctl list-units --type=service --all | grep failed
```


atd.service loaded failed failed Deferred execution scheduler

postfix.service loaded failed failed Postfix Mail Transport Agent

- **Unit Files and Their Structure**
  - Located under `/usr/lib/systemd/system/` (packaged units) and `/etc/systemd/system/` (administrator overrides or custom units).
  - Key sections in a unit file (e.g., `httpd.service`):
    - **[Unit]**: Description, dependencies (`After=`, `Requires=`).
    - **[Service]**: How to start the service (`ExecStart=`), user to run as (`User=`), environment variables.
    - **[Install]**: Installation behavior (e.g., `WantedBy=multi-user.target` makes the service start in normal multi-user mode).

Example snippet (`/etc/systemd/system/customapp.service`):

```
[Unit]
Description=Custom Application Service
After=network.target

[Service]
Type=simple
User=appuser
ExecStart=/usr/local/bin/customapp --config /etc/customapp/config.yml
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

After modifying or creating unit files, reload systemd to apply changes:

```
$ sudo systemctl daemon-reload
```


- **Service Troubleshooting**
  - **Checking Logs with `journalctl`**

View the entire journal (systemd's central log):

```
$ sudo journalctl
```

Follow a service's recent logs (like `tail -f`):

```
$ sudo journalctl -u httpd.service -f
```

Limit by time or boot ID:

```
# Logs since today at 00:00
$ sudo journalctl --since "2025-06-02 00:00:00"
```

Identifying Failed Units

```
$ systemctl --failed
```

- Lists units in a failed state (helpful after reboot or manual changes).

- **Managing Daemons for Network Services**
  - Common RHEL network-related services:
    - **`sshd.service`**: OpenSSH server (enables remote shell access).
    - **`httpd.service`**: Apache HTTP Server.
    - **`nginx.service`**: NGINX web server.
    - **`firewalld.service`**: Dynamic firewall manager.
    - **`NetworkManager.service`**: High-level network configuration and management.

Example: Starting and enabling SSH:

```
$ sudo systemctl start sshd.service
```


$ sudo systemctl enable sshd.service

- **Service Targets and Runlevels**
  - systemd replaces traditional runlevels with “targets.” Common targets:
    - `graphical.target`: multi-user with graphical interface.
    - `multi-user.target`: multi-user, non-graphical (similar to runlevel 3).
    - `rescue.target`: single-user rescue shell.
    - `emergency.target`: emergency maintenance mode (no networking).

Switch targets:

$ sudo systemctl isolate multi-user.target

Check default target (what system boots into):

$ systemctl get-default

- **Continuity and Dependencies**
  - Proper management of services and daemons lays the groundwork for securing network daemons (SSH), installing application stacks (web, database), and ensuring critical logging services remain active (rsyslog).
  - Subsequent modules—such as “Configure and secure SSH” and “Analyze and store logs”—build upon these systemd skills.

Configure and secure SSH

Configure secure command line service on remote systems, using OpenSSH.

- **Overview of OpenSSH**
  - Provides encrypted communication channels over a network using the SSH protocol.
  - Replaces legacy Telnet, Rlogin, and FTP by offering confidentiality, integrity, and strong authentication.

- **Installing and Verifying OpenSSH**

Ensure `openssh-server` is installed:

```
$ sudo dnf install -y openssh-server openssh-clients
```

Verify `sshd` status:

```
$ sudo systemctl status sshd.service
```

If not running, start and enable:

```
$ sudo systemctl start sshd.service
$ sudo systemctl enable sshd.service
```

- **Default Configuration File (`/etc/ssh/sshd_config`)**
  - **Commonly Modified Settings**:

Port: Change default port 22 to a non-standard port (e.g., 2222) to reduce automated scanning.

```
Port 2222
```

PermitRootLogin: Disable direct root login (recommended).

```
PermitRootLogin no
```

PasswordAuthentication: Consider disabling password authentication once key-based authentication is in place.

```
PasswordAuthentication no
```

PubkeyAuthentication: Ensure public-key authentication is enabled.

```
PubkeyAuthentication yes
```


AllowUsers / AllowGroups: Restrict which users or groups can log in via SSH.

AllowUsers alice bob

AllowGroups sshusers

-

MaxAuthTries: Limit authentication attempts to mitigate brute-force attacks.

MaxAuthTries 3

-

LoginGraceTime: Time allowed for user to authenticate.

LoginGraceTime 30s

-

X11Forwarding: Disable if not needed.

X11Forwarding no

-

- **Key-Based Authentication**

Generating an SSH Key Pair (on client):

$ ssh-keygen -t rsa -b 4096 -C "alice@example.com"

-
  - Saves private key at `~/.ssh/id_rsa` and public key at `~/.ssh/id_rsa.pub` by default.
  - Protect private key with a passphrase (optional but recommended).

Copy Public Key to Server:

$ ssh-copy-id -i ~/.ssh/id_rsa.pub alice@rhel-server.example.com

-
  - Alternatively, manually append contents of `id_rsa.pub` to `~/.ssh/authorized_keys` on the server (ensure proper file permissions—`700` for `~/.ssh`, `600` for `authorized_keys`).

Verify Key-Based Login:

$ ssh alice@rhel-server.example.com -p 2222

- Should not prompt for a password (only passphrase if private key is encrypted).

- **Securing SSH Beyond the Basics**
  - **Disable Unused Authentication Methods**

In `sshd_config`:

```
ChallengeResponseAuthentication no
KerberosAuthentication no
UsePAM yes          # Or no if not using PAM
```

Enforce Protocol Version 2 Only (Protocol 1 is obsolete/insecure):

```
Protocol 2
```

  - **Limit User Access**
    - Use `AllowUsers`, `AllowGroups`, `DenyUsers`, `DenyGroups`.

Example: Only members of group `sshusers` can SSH:

```
AllowGroups sshusers
```

  - **Implement Two-Factor Authentication (2FA)**
    - RHEL supports Google Authenticator or YubiKey.

Install and configure `pam_google_authenticator`:

```
$ sudo dnf install -y google-authenticator
$ google-authenticator        # Run as user to generate QR code and secret key
```

Modify `/etc/pam.d/sshd` to include:

```
auth required pam_google_authenticator.so
```


Update `sshd_config` to require 2FA:

AuthenticationMethods publickey,keyboard-interactive

- **Firewall Configuration for SSH**

RHEL uses `firewalld` by default; ensure SSH port is allowed.

Open default port 22 (or custom port 2222)

$ sudo firewall-cmd --permanent --add-service=ssh

If using custom port:

$ sudo firewall-cmd --permanent --remove-service=ssh

$ sudo firewall-cmd --permanent --add-port=2222/tcp

$ sudo firewall-cmd --reload

- **Auditing SSH Logins**
  - **`journalctl -u sshd`**: view SSHD logs.

`/var/log/secure`: RHEL logs authentication attempts here (may require `sudo`).

$ sudo grep "Failed password" /var/log/secure

  - **Fail2Ban** (optional, open-source intrusion prevention): can monitor logs for repeated failed logins and block offending IPs.

- **Continuity and Dependencies**
  - After SSH is secured, administrators can safely perform remote management tasks—installing packages, configuring networking, or analyzing logs—without worrying about brute-force attacks or unauthorized access.
  - Later modules that rely on remote access (e.g., “Manage networking,” “Install and update software packages”) assume SSH is configured properly.

Analyze and store logs

Locate and accurately interpret logs of system events for troubleshooting purposes.

- **Where Logs Reside**
    - Traditional syslog files (via `rsyslog`) are stored under `/var/log/`. Common examples:
        - `/var/log/messages` or `/var/log/syslog`: general system messages (kernel, services).
        - `/var/log/secure`: authentication and authorization logs (SSH login attempts, sudo usage).
        - `/var/log/cron`: cron job execution and errors.
        - `/var/log/maillog`: mail server logs.
        - `/var/log/dmesg`: kernel ring buffer (boot messages).
        - `/var/log/audit/audit.log`: SELinux and audit subsystem logs (if `auditd` is running).
- **Journalctl: systemd's Centralized Logging**
    - **Basic Usage**

View entire journal (paged):

```
$ sudo journalctl
```

Follow live log entries (similar to `tail -f`):

```
$ sudo journalctl -f
```

Show logs since boot:

```
$ sudo journalctl -b
```

Show logs for a specific service:

```
$ sudo journalctl -u firewalld.service
```


Show logs within a time range:

```
$ sudo journalctl --since "2025-06-01 08:00:00" --until "2025-06-01 12:00:00"
```

- **Filtering and Searching Logs**
  - **By Priority (severity)**

Priorities range from `emerg` (0) to `debug` (7). For example, to display only errors and above:

```
$ sudo journalctl -p err
```

By Priority Keyword

```
$ sudo journalctl -p warning
```

By Source or Identifier (SYSLOG_IDENTIFIER)

```
$ sudo journalctl SYSLOG_IDENTIFIER=sshd
```

By PID

```
$ sudo journalctl _PID=13521
```

By Unit

```
$ sudo journalctl -u NetworkManager
```

- **Analyzing Traditional Log Files**

Use `grep`, `awk`, or `sed` for targeted searches:

```
# Find all “error” entries in /var/log/messages
$ sudo grep -i error /var/log/messages
# Count number of SSH failures
$ sudo grep -c "Failed password" /var/log/secure
```

- Combine commands with timestamps to isolate events around system failures.
- Employ `tail -n 100 /var/log/crond` to see the last 100 lines of cron logs.

- **Log Rotation and Archiving**
  - RHEL uses `logrotate` to rotate, compress, and remove old logs per defined policies.
  - Configuration files reside in `/etc/logrotate.conf` and `/etc/logrotate.d/`.

Example (`/etc/logrotate.d/httpd`):

```
/var/log/httpd/*log {
    missingok
    notifempty
    sharedscripts
    postrotate
        /usr/bin/systemctl reload httpd.service > /dev/null 2>/dev/null || true
    endscript
}
```

  - By default, rotated logs are compressed (gzip) and kept for a set number of cycles.

- **Centralized Logging (Brief Overview)**
  - RHEL can forward logs to a central syslog server or ELK/EFK stack (Elasticsearch, Fluentd/Logstash, Kibana) for aggregated analysis.

Basic forwarding configuration (in `/etc/rsyslog.conf` or `/etc/rsyslog.d/`):

```
*.* @@central-syslog.example.com:514
```

- **Maintaining and Storing Logs**
  - Ensure `/var/log/` partition has sufficient space—monitor with `df -h`.

- Implement retention policies: rotate logs frequently (daily or weekly) if verbose applications generate large volumes.
- Archive older logs off server for compliance (e.g., monthly tarballs moved to backup server).

- **Interpreting Common Log Entries**
  - **Kernel Messages** (`/var/log/dmesg` or `journalctl -k`): hardware initialization, driver errors.
  - **Authentication** (`/var/log/secure` or `journalctl _COMM=sshd`): successful and failed login attempts, sudo usage.
  - **Service Failures** (`journalctl -u <service>`): configuration errors, missing dependencies.
  - **SELinux Denials** (`/var/log/audit/audit.log` or `ausearch -m avc`): files or processes blocked by SELinux policy.
- **Continuity and Dependencies**
  - Proper log analysis aids in troubleshooting configuration errors made in earlier modules (e.g., incorrect SSH settings, file permission issues).
  - Subsequent sections—such as “Manage networking” and “Analyze servers and get support”—depend on interpreting network and system-level logs.

Manage networking

Configure network interfaces and settings on Red Hat Enterprise Linux servers.

- **Understanding RHEL’s Network Configuration Tools**
  - **NetworkManager**: default high-level tool for managing network interfaces via CLI (`nmcli`), text-based UI (`nmtui`), or GUI (if installed).
  - **Legacy ifcfg Scripts** (Network scripts): located under `/etc/sysconfig/network-scripts/ifcfg-*`, still supported but gradually being deprecated in favor of NetworkManager.
  - **`ip` command**: part of `iproute2`, replacing legacy `ifconfig`.
- **Viewing Current Network Configuration**

`ip addr show`: list all interfaces and their IP addresses.

```
$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:0c:29:68:22:af brd ff:ff:ff:ff:ff:ff
    inet 192.168.10.50/24 brd 192.168.10.255 scope global dynamic noprefixroute ens33
```

-

`nmcli device status`: show device names, types, and connection states.

```
$ nmcli device status
DEVICE   TYPE      STATE      CONNECTION
ens33    ethernet  connected  Wired connection 1
lo       loopback  unmanaged  --
```

-

`nmcli connection show`: list network profiles (connections).

```
$ nmcli connection show
NAME                UUID                                  TYPE      DEVICE
Wired connection 1  a1b2c3d4-5678-9abc-def0-1234567890ab  ethernet  ens33
```

-

- **Configuring a Static IP Address (via NMCLI)**

Example: Set a static IPv4 address, gateway, and DNS server on interface `ens33`.

```
$ sudo nmcli con add type ethernet con-name static-ens33 ifname ens33 \
  ipv4.method manual ipv4.addresses 10.10.10.50/24 ipv4.gateway 10.10.10.1 \
  ipv4.dns 8.8.8.8 autoconnect yes
```

-

Bring up the new connection (or restart NetworkManager to apply changes):

$ sudo nmcli con up static-ens33

-

Verify with:

$ ip addr show ens33

$ ping -c 3 10.10.10.1

-

- **Configuring DHCP**

Enable DHCP on an interface (replace any static configuration):

$ sudo nmcli con modify ens33 ipv4.method auto

$ sudo nmcli con up ens33

- Check assigned IP address:

  $ ip addr show ens33 | grep "inet "

- **Editing Legacy ifcfg Files (Optional / For Advanced Compatibility)**

Files located at `/etc/sysconfig/network-scripts/ifcfg-ens33`. Example content for a static IP:

TYPE=Ethernet

BOOTPROTO=none

NAME=ens33

DEVICE=ens33

ONBOOT=yes

IPADDR=10.10.10.50

PREFIX=24

GATEWAY=10.10.10.1

DNS1=8.8.8.8

-

After editing, restart network scripts or NetworkManager:

```
$ sudo systemctl restart NetworkManager
```

- **Managing DNS**
  - RHEL uses `systemd-resolved` (or direct `/etc/resolv.conf` if `resolvconf` is not installed).

Check `/etc/resolv.conf`:

```
$ cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
```

To add an additional DNS via NMCLI:

```
$ sudo nmcli con modify ens33 +ipv4.dns 1.1.1.1
$ sudo nmcli con up ens33
```

- **Configuring Hostname**

Temporary change (until next reboot):

```
$ sudo hostnamectl set-hostname appserver01
```

Verify:

```
$ hostnamectl
Static hostname: appserver01
```

To map hostname to IP in absence of DNS, edit `/etc/hosts`:

```
10.10.10.50   appserver01.example.com   appserver01
```

- **Firewall Configuration**

As in the SSH section, use `firewalld` to allow or block traffic. Common zones: `public`, `internal`, `trusted`.

```
# Check active zone for ens33
$ sudo firewall-cmd --get-active-zones
public
  interfaces: ens33
# Allow HTTP in public zone
$ sudo firewall-cmd --permanent --add-service=http
$ sudo firewall-cmd --reload
# Verify allowed services
$ sudo firewall-cmd --list-all
```

- **Checking Network Connectivity**

`ping`: test ICMP reachability.

```
$ ping -c 4 google.com
```

`traceroute` (install via `sudo dnf install -y traceroute`): diagnose routing issues.

```
$ traceroute dashboard.redhat.com
```

`curl` **or** **`wget`**: test HTTP connections.

```
$ curl -I https://downloads.redhat.com
```

- **Checking Routing Table**

`ip route show`: display current route table.

```
$ ip route show
default via 10.10.10.1 dev ens33 proto dhcp metric 100
10.10.10.0/24 dev ens33 proto kernel scope link src 10.10.10.50 metric 100
```


- **Bridging, Bonding, and VLANs (Brief Introduction)**

Bridge: virtual switch to connect multiple interfaces (useful for VMs or containers).

$ sudo nmcli con add type bridge ifname br0 con-name br0

$ sudo nmcli con add type bridge-slave ifname ens34 master br0

Bonding: combine multiple physical NICs into a single logical interface for redundancy or increased bandwidth.

$ sudo nmcli con add type bond ifname bond0 mode active-backup

$ sudo nmcli con add type bond-slave ifname ens35 master bond0

$ sudo nmcli con add type bond-slave ifname ens36 master bond0

VLANs: create subinterfaces tagged with VLAN IDs.

$ sudo nmcli con add type vlan ifname ens33.100 dev ens33 id 100 ip4 192.168.100.10/24

- **Troubleshooting Networking**
  - `nmcli general status`: check overall NetworkManager status.
  - `nmcli connection show --active`: which connections are up.
  - `ip link show`: check link state (UP/DOWN).

`ethtool <interface>` (install `ethtool` if needed): check NIC settings (speed, duplex).

$ sudo ethtool ens33

- **Continuity and Dependencies**
  - Solid networking skills are prerequisites for installing packages (which often require internet access), configuring remote storage (NFS, iSCSI), and participating in clustered environments.
  - Later modules—like “Install and update software packages” and “Access Linux file systems”—assume networks are correctly configured for repository access and remote mounts.

Install and update software packages

Download, install, update, and manage software packages from Red Hat and DNF package repositories.

- **Understanding RPM and DNF/YUM**
    - RHEL packages use the **RPM** format (`.rpm`). RPM is a low-level tool to install, query, and remove packages.
    - **DNF** (Dandified YUM) is the high-level package manager that handles dependency resolution, repositories, and metadata. In RHEL 8 and newer, DNF replaces YUM (though `/usr/bin/yum` is a symlink to `dnf` for backward compatibility).
- **Exploring Repositories**
    - RHEL subscriptions grant access to official Red Hat repositories.

Use **Red Hat Subscription Manager (RHSM)** to attach subscriptions and enable repos:

```
$ sudo subscription-manager register --username user@example.com --password 'P@ssw0rd'
$ sudo subscription-manager attach --auto
$ sudo subscription-manager repos --list-enabled
```

Enable or disable specific repos:

```
# Enable CodeReady Linux Builder (for developer tools)
$ sudo subscription-manager repos --enable codeready-builder-for-rhel-8-x86_64-rpms
# Disable debug repo
$ sudo subscription-manager repos --disable rhel-8-for-x86_64-debug-rpms
```

- **Basic DNF Commands**

Install Packages

```
$ sudo dnf install httpd
```

Remove Packages

$ sudo dnf remove httpd

-

Update Packages

$ sudo dnf update # Update all installed packages

$ sudo dnf update httpd # Update only httpd

-

Search for Packages

$ sudo dnf search nginx

-

Show Package Info

$ sudo dnf info nginx

Name : nginx

Version : 1.18.0

Release : 1.el8

Summary : High performance web server

-

List Installed Packages

$ sudo dnf list installed | grep kernel

kernel.x86_64 4.18.0-240.el8 @anaconda

kernel-tools.x86_64 4.18.0-240.el8 @commandline

-

- **Managing Package Groups and Modules**
  - RHEL 8+ introduces **Application Streams (AppStreams)** and **module streams** for multiple versions of software to coexist.

List modules:

$ sudo dnf module list nginx

Name Stream Profiles Summary

nginx 1.14 [d] common [d], minimal nginx web server

1.16 common, minimal

1.18 common, minimal

-

Enable a module stream and install a profile:

$ sudo dnf module enable nginx:1.18

$ sudo dnf module install nginx:1.18/common

-

Groups: predefined sets of packages (e.g., “Development Tools”).

$ sudo dnf group list

Development Tools

System Management

…

$ sudo dnf group install "Development Tools"

-

- **Working with Local RPMs**

Installing an RPM file (without resolving dependencies):

$ sudo rpm -ivh /tmp/custom-app-1.0-1.el8.x86_64.rpm

-

Upgrading or Replacing a Package:

$ sudo rpm -Uvh /tmp/custom-app-1.1-1.el8.x86_64.rpm

-

Querying an RPM:

$ rpm -qpl /tmp/custom-app-1.0-1.el8.x86_64.rpm # list files in the RPM

$ rpm -qi custom-app # info about installed package

-

Removing an RPM:

$ sudo rpm -e custom-app

- **Checking for Vulnerabilities and Updates**

`dnf updateinfo`: view security advisories and bug fixes.

$ sudo dnf updateinfo list security

Applying only security updates:

$ sudo dnf update --security

- **Enabling Third-Party or EPEL Repositories**

EPEL (Extra Packages for Enterprise Linux): community-maintained repository of additional packages.

$ sudo dnf install -y epel-release

  - Caution: validate compatibility and official support policies before enabling third-party repos in production.

- **Verifying Package Integrity**

RPM packages are GPG-signed. Import Red Hat's GPG key before installation if working offline:

$ sudo rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release

To verify an installed package's files:

$ rpm -V httpd

    - If any files have been modified or corrupted, the output shows a code (e.g., `S` for size mismatch, `D` for MD5 sum changed).

- **Continuity and Dependencies**

  - Proper package management is crucial before installing and configuring network services (e.g., web servers, databases) or security tools (e.g., firewall, intrusion

detection).

- Later chapters—such as “Access Linux file systems” (e.g., installing `nfs-utils`) and “Analyze servers and get support” (e.g., updating the kernel to fix a bug)—depend on these skills.

Access Linux files systems

Access, inspect, and use existing file systems on storage attached to a Linux server.

- **Understanding Linux Filesystem Layout**
  - RHEL follows the Filesystem Hierarchy Standard (FHS):
    - `/` – Root of the filesystem tree.
    - `/boot` – Static files of the bootloader and kernel.
    - `/etc` – Host-specific system configuration.
    - `/var` – Variable data (logs, spool files, caches).
    - `/home` – Personal directories for users.
    - `/usr` – Secondary hierarchy for user utilities and applications.
    - `/dev` – Device files (block and character special files).
    - `/mnt` or `/media` – Mount points for temporary media (USB, ISO).
- **Inspecting Attached Block Devices**

`lsblk`: list block devices (disks, partitions) in a tree.

```
$ lsblk
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0   100G  0 disk
├─sda1   8:1    0   500M  0 part /boot
└─sda2   8:2    0  99.5G  0 part /
sdb      8:16   0   200G  0 disk
└─sdb1   8:17   0   200G  0 part
```


- `fdisk -l` (requires `sudo`): list partition tables for all disks.

```
$ sudo fdisk -l /dev/sdb
Disk /dev/sdb: 200 GiB, 214748364800 bytes, 419430400 sectors
Disklabel type: gpt
Device     Start       End   Sectors Size Type
/dev/sdb1       2048 419430366 419428319 200G Linux filesystem
```

- `blkid`: show UUIDs, labels, and filesystem types.

```
$ sudo blkid /dev/sdb1
/dev/sdb1: UUID="e3f4a5b6-c7d8-4e9f-0a1b-2c3d4e5f6a7b" TYPE="ext4" PARTLABEL="primary" PARTUUID="1a2b3c4d-5e6f-7a8b-9c0d-1e2f3a4b5c6d"
```

- **Mounting Filesystems**

Temporarily Mount a Partition

```
# Create mount point
$ sudo mkdir /mnt/data
# Mount partition (e.g., ext4 in /dev/sdb1)
$ sudo mount /dev/sdb1 /mnt/data
# Verify
$ df -h | grep /mnt/data
```

Unmount a Filesystem

```
$ sudo umount /mnt/data
```

Mount by Label or UUID

```
# By label
$ sudo mount -L datapart /mnt/data
```


By UUID

$ sudo mount UUID=e3f4a5b6-c7d8-4e9f-0a1b-2c3d4e5f6a7b /mnt/data

- **Persistent Mounts (FSTAB Entry)**

Edit `/etc/fstab` to automatically mount at boot. Format:

<device> <mount_point> <type> <options> <dump> <pass>

Example (mount by UUID, ext4, defaults, no dump, fsck order 0):

UUID=e3f4a5b6-c7d8-4e9f-0a1b-2c3d4e5f6a7b /mnt/data ext4 defaults 0 0

After editing, test with:

$ sudo mount -a

  - No errors indicate the `fstab` entry is valid.

- **Inspecting Filesystem Health**

`df -h`: show mounted filesystems, usage, and available space.

$ df -h

Filesystem Size Used Avail Use% Mounted on

/dev/sda2 50G 20G 28G 42% /

/dev/sdb1 200G 50G 140G 27% /mnt/data

`du -sh <directory>`: show total disk usage of a directory.

$ du -sh /mnt/data

50G /mnt/data

  - **`fsck` (Filesystem Check)**

        - Check and repair a filesystem (must be unmounted or mounted read-only).

$ sudo umount /dev/sdb1

$ sudo fsck -f /dev/sdb1

    -
        - Add `-y` to automatically answer “yes” to repair prompts.

- **LVM (Logical Volume Manager) Basics**
    - Provides flexible volume management: create volume groups (VG), logical volumes (LV), and physical volumes (PV).
    - **`pvcreate /dev/sdc1`**: mark a partition as a physical volume.
    - **`vgcreate data_vg /dev/sdc1`**: create a volume group named `data_vg`.
    - **`lvcreate -n data_lv -L 100G data_vg`**: create a logical volume named `data_lv` with size 100GiB.

Create Filesystem and Mount:

$ sudo mkfs.xfs /dev/data_vg/data_lv

$ sudo mkdir /mnt/lvdata

$ sudo mount /dev/data_vg/data_lv /mnt/lvdata

    -

Extend LVM Volumes:

Grow logical volume by 50G

$ sudo lvextend -L +50G /dev/data_vg/data_lv

Resize filesystem (for XFS, cannot shrink; can grow online)

$ sudo xfs_growfs /mnt/lvdata

    -

Snapshots (useful for backups or testing):

$ sudo lvcreate --size 10G --snapshot --name data_lv_snap /dev/data_vg/data_lv

    -

- **Network File Systems (NFS, CIFS)**

Mount NFS Shares

Install NFS utilities

$ sudo dnf install -y nfs-utils

Mount NFS export

$ sudo mount -t nfs server.example.com:/export/data /mnt/nfsdata

-

Add to `/etc/fstab` for persistence:

server.example.com:/export/data /mnt/nfsdata nfs defaults 0 0

-

Mount CIFS/SMB Shares (e.g., from a Windows file server)

Install CIFS utils

$ sudo dnf install -y cifs-utils

Mount share with credentials

$ sudo mount -t cifs //windows-server/shared /mnt/cifsdata -o username=winuser,password='P@ssw0rd',uid=1000,gid=1000

-

- **Disk Quotas (Brief Introduction)**
  - Limit user or group disk usage on filesystems.

Enable quotas in `/etc/fstab`: add `usrquota` or `grpquota` to mount options.

/dev/sda2 /home xfs defaults,usrquota,grpquota 0 0

-

Remount and initialize quota database:

$ sudo mount -o remount /home

$ sudo quotacheck -cum /home

$ sudo quotaon /home

-

Set quotas:

$ sudo edquota -u alice

-
    - Opens an editor to specify soft/hard limits.

- **Continuity and Dependencies**
    - Controlling filesystems is a key step before creating user home directories, mounting application data disks, or provisioning storage for databases or logs.
    - Later topics—such as “Analyze servers and get support” and “Install and update software packages”—may involve troubleshooting mount issues or expanding storage.

Analyze servers and get support

Investigate and resolve issues in the web-based management interface, getting support from Red Hat to help solve problems.

- **Red Hat Customer Portal and Support Channels**
    - **https://access.redhat.com**: central site for subscribing organizations to access knowledge base articles, knowledgebase (KB), product documentation, and support cases.
    - **Opening a Support Case**:
        - Log into Red Hat Customer Portal with subscribed credentials.
        - Click on “Get Support” → “Open a Support Case.”
        - Provide detailed information: product, version, severity, system architecture, and specific error messages or logs.
        - Attach relevant logs, configuration files, and other diagnostic data.
        - Engage with Red Hat’s support engineers via the portal or phone.
- **Web-Based Management Interface (RHEL Web Console or Cockpit)**
    - **Cockpit**: RHEL’s default browser-based interface for monitoring and managing systems. Typically listens on port 9090.

Installing and Enabling Cockpit:

$ sudo dnf install -y cockpit

$ sudo systemctl enable --now cockpit.socket

-

- **Accessing Cockpit**:

Open a web browser and navigate to:

https://rhel-server.example.com:9090/

  -
  - Log in using a privileged account (e.g., a user in the `wheel` group).
- **Cockpit Features**:
  - **Dashboard**: summary of CPU, memory, storage, and network usage.
  - **Logs**: view system and service logs, filter by severity or keyword.
  - **Networking**: manage interfaces, firewall settings, and networking metrics.
  - **Storage**: view and configure disks, partitions, logical volumes, and filesystems.
  - **Services**: start, stop, enable, or disable systemd services.
  - **Software Updates**: review available updates (security, bugfix, enhancement) and apply them.
  - **Terminal**: launch a web-based terminal in the browser for command-line access.

- **Investigating Common Server Issues**
  - **High CPU or Memory Usage**
    - Use Cockpit's performance graphs or CLI tools (`top`, `htop`).
    - Identify runaway processes, consider `nice`/`renice` or resource limits (cgroups).
  - **Disk Space Exhaustion**
    - Monitor with `df -h` or Cockpit's Storage view.
    - Identify large directories with `du -sh /*` or `du -sh /var/log/*`.
    - Clean archived logs or expand partitions (LVM).
  - **Service Failures**
    - Check `systemctl status <service>` for startup errors.
    - Inspect logs via `journalctl -u <service>` or Cockpit's Logs.

        - Verify configuration file syntax (e.g., `apachectl configtest`).
    - **Networking Problems**
        - Confirm interface status with `nmcli device status` or `ip addr`.
        - Test connectivity using `ping`, `traceroute`, `curl`.
        - Check firewall rules (`firewall-cmd --list-all`) or SELinux denials (`audit2why`).
- **Collecting Diagnostic Data for Support**
    - **`Red Hat System Role: sosreport`**

Collects details about the system's hardware, configuration, logs, and other diagnostics into a compressed archive.

$ sudo dnf install -y sos

$ sudo sosreport

        -
        - The resulting archive (`sosreport-<hostname>-<date>.tar.xz`) can be securely uploaded to Red Hat Support.
    - **Gathering Logs and Configuration Files Manually**
        - Identify relevant logs (e.g., `/var/log/messages`, `/var/log/secure`, `/var/log/httpd/error_log`).

Archive config directories:

$ sudo tar czf /tmp/httpd_conf_backup.tar.gz /etc/httpd/

        -
    - **Hardware Diagnostics**

`lshw`: show hardware configuration (may require installation).

$ sudo dnf install -y lshw

$ sudo lshw -short

        -
        - **`dmidecode`**: report BIOS, motherboard, and hardware details.

    - **RAID Controller Logs** (if applicable): consult storage vendor tools (e.g., `megacli` for LSI controllers).

- **Leveraging Online Knowledge Base**
    - Red Hat Knowledgebase: search by error message, product version, or keyword.
    - Filter by severity, product life cycle (EUS, standard support).
    - Bookmark frequently used KB articles for quick reference (e.g., “How to recover from network misconfiguration,” “Troubleshooting SELinux denials”).
- **Interacting with Red Hat Insights (Proactive Analytics)**
    - **Red Hat Insights**: cloud-based service that analyzes RHEL systems for known issues and recommends remediation.

Install client:

$ sudo subscription-manager repos --enable=rhel-8-for-x86_64-insights-client-rpms

$ sudo dnf install -y insights-client

$ sudo insights-client --register

    -
    - After registration, the system sends telemetry data to Red Hat Insights.
    - View recommendations via the Customer Portal (lists vulnerabilities, configuration issues, performance problems).

- **Maintenance Windows and Impact Minimization**
    - Schedule updates or configuration changes during off-peak hours to reduce user impact.
    - Use Cockpit’s “Software Updates” panel to preview updates before applying.
    - For critical services, consider redundant nodes (load balancing) or rolling updates to maintain availability.
- **Continuity and Dependencies**
    - The ability to self-diagnose and gather the right diagnostic data accelerates support resolution and reduces downtime.
    - Upcoming modules—such as “Configure and secure SSH,” “Manage networking,” and “Install and update software packages”—expect administrators to proactively monitor system health and logs.

Overall Continuity and Next Steps

By mastering these foundational topics—understanding open source principles, logging in, managing files, using local help, editing text files, administering users and groups, controlling file access, monitoring processes, managing services, securing SSH, analyzing logs, configuring networking, installing packages, and working with filesystems—learners are prepared to tackle more advanced areas:

1. **Security Hardening**: building on file permissions, SSH hardening, and SELinux to implement intrusion detection, firewall rules, and vulnerability remediation.

2. **Automation with Ansible**: using the command-line skills to write playbooks that automate package installations, user creations, service configurations, and security policies.

3. **Containerization and Kubernetes**: leveraging knowledge of processes, services, and networking to deploy Docker/Podman containers and orchestrate them in OpenShift/EKS environments.

4. **Performance Tuning**: applying process monitoring, log analysis, and resource management to tune system performance under heavy workloads.

When moving into those topics, refer back to these notes as prerequisites—especially for remote access (SSH), package installation (DNF), and system configuration (editing files, restarting services).

Finally, once these core RHEL skills are firmly in place, the next phase will cover how to **infuse AI** into routine tasks: automating log analysis (e.g., using machine learning to detect anomalies), intelligent resource autoscaling (predictive CPU/memory provisioning), and AI-driven security monitoring (e.g., integrating Red Hat Insights with custom AI pipelines). Those advanced AI integration strategies will be introduced in subsequent modules of the series.

