Red Hat System Administration II

Complete Quick Reference Guide

Ahmed Abdelwahed

Technical Trainer | Cloud & Infrastructure Specialist

Email	Blog	LinkedIn
ahmed@abdelwahed.me	www.abdelwahed.me	linkedin.com/in/ahmedmct

01. Linux Boot Sequence & GRUB2

The 8-Stage Boot Process

#	Stage	What Happens
1	**Power On / POST**	BIOS/UEFI loads from non-volatile memory. Performs POST — checks CPU, RAM, disk controllers, hardware.
2	**Hardware Detection**	BIOS/UEFI probes and initializes storage controllers, disks, USB, network adapters, and graphics.
3	**Select Boot Device**	Reads boot order, scans bootable media (HDD, USB, PXE, DVD), hands off to boot sector or EFI loader.
4	**GRUB2 Boot Loader**	Loads from MBR or EFI System Partition. Reads /boot/grub2/grub.cfg. Presents boot menu.
5	**Kernel Initialization**	GRUB loads vmlinuz + initramfs. Kernel unpacks, loads drivers, detects hardware, mounts real root filesystem.
6	**systemd (PID 1)**	systemd starts as the first user-space process. Activates units in dependency order, switches to default target.
7	**Startup Services**	Networking, sshd, firewalld, cron/timers start. Init files loaded: /etc/profile, ~/.bash_profile, PAM modules.
8	**Login Available**	getty or GDM/SDDM login managers start. Users can authenticate via console or SSH.

Systemd Targets (Runlevels)

Command / Option	Description
`systemctl get-default`	Show current default boot target
`systemctl --type=target`	List all loaded targets
`systemctl --type=target --all`	List all targets including inactive
`runlevel`	Show current runlevel (legacy compatibility)
`systemctl set-default multi-user.target`	Boot to CLI mode persistently
`systemctl set-default graphical.target`	Boot to GUI mode persistently
`systemctl isolate rescue.target`	Switch to rescue mode now (non-persistent)
`systemctl isolate multi-user.target`	Switch to multi-user CLI mode now
`systemctl poweroff`	Shut down the system
`systemctl reboot`	Reboot the system

Target	SysV Equiv	Purpose
`emergency.target`	—	Minimal shell; root filesystem read-only. Use when filesystems are corrupt.
`rescue.target`	Runlevel 1	Single-user mode; root shell + basic services. Use for config errors or password resets.
`multi-user.target`	Runlevel 3	Full multi-user CLI (no GUI). Standard for servers.

Target	SysV Equiv	Purpose
`graphical.target`	Runlevel 5	Full multi-user with GUI (GDM). Standard for desktops.
`reboot.target`	Runlevel 6	System reboot.

GRUB2 Configuration

Never edit grub.cfg directly — it gets overwritten on updates. Use the method below instead.

Method A: Permanent Changes

```
1  cp /etc/default/grub /etc/default/grub.bak  — backup first
2  nano /etc/default/grub  — edit GRUB_CMDLINE_LINUX (remove 'rhgb quiet' for
   verbose boot)
3  grub2-mkconfig -o /boot/grub2/grub.cfg  — regenerate config
```

Method B: Temporary (One-time) Boot Parameters

```
1  Reboot → highlight kernel in GRUB menu
2  Press 'e' to edit the boot entry
3  Find line starting with 'linux' and append your target at the end
4  Press Ctrl+X to boot with modified parameters
```

Command / Option	Description
`systemd.unit=rescue.target`	Boot to rescue mode (append to linux line)
`systemd.unit=emergency.target`	Boot to emergency mode (append to linux line)
`rd.break`	Stop before mounting real root — used for password resets
`init=/bin/bash`	Boot to a raw bash shell (no systemd)

GRUB2 Password Protection

```
1  grub2-setpassword  — set GRUB superuser password
2  cat /boot/grub2/user.cfg  — view hashed password
3  grub2-mkconfig -o /boot/grub2/grub.cfg  — apply changes
4  Reboot → press 'e' to test password protection
5  rm /boot/grub2/user.cfg  — remove password if needed
```

Reset Lost Root Password

```
1  At GRUB menu, press 'e' to edit boot entry
2  Append  init=/bin/bash  to the linux line, then Ctrl+X
3  mount -o remount,rw /  — remount root as read-write
4  passwd  — set new root password
```

```
touch /.autorelabel  — trigger SELinux relabeling on next boot
exec /usr/lib/systemd/systemd  — or reboot -f
```

Login Shell vs Non-Login Shell

Property	Login Shell	Non-Login Shell
When	SSH login, console login, su - user	SSH login, console login, su - user
Files Read	/etc/profile → ~/.bash_profile → ~/.bashrc	/etc/profile → ~/.bash_profile → ~/.bashrc
Purpose	Full environment initialization	Full environment initialization

02. Scheduling Jobs — at, cron, anacron, systemd Timers

at — One-Time Scheduled Tasks

```
dnf install at  — install the at package
systemctl enable --now atd  — enable and start the atd service
at 4:00 PM  — schedule a task (type commands, then Ctrl+D to save)
```

Command / Option	Description
`at 4:00 PM`	Run at 4:00 PM today
`at now + 2 hours`	Run 2 hours from now
`at 3:00 PM tomorrow`	Run at 3 PM tomorrow
`at 10:00 AM next Monday`	Run at 10 AM next Monday
`at 8:00 AM Jul 15`	Run at specific date
`atq`	View scheduled at jobs
`atrm <job_number>`	Remove a scheduled at job

cron — Recurring Scheduled Tasks

Crontab Format

```
* * * * * command_to_run
| | | | |
| | | | └── Day of week (0-7, 0/7=Sun)
| | | └──── Month (1-12)
| | └────── Day of month (1-31)
| └──────── Hour (0-23)
└────────── Minute (0-59)
```

crontab Management

Command / Option	Description
`crontab -e`	Edit current user's crontab
`crontab -l`	List current user's crontab
`crontab -r`	Remove current user's crontab
`crontab -e -u username`	Edit crontab for specific user (root)
`crontab -l -u username`	List crontab for specific user
`vim /etc/crontab`	Edit system-wide crontab (includes user field)

Cron Job Examples

Cron Expression	Meaning
`* * * * * /cmd`	Every minute

Cron Expression	Meaning
`0 5,17 * * * /cmd`	At 5 AM and 5 PM every day
`*/10 * * * * /cmd`	Every 10 minutes
`0 17 * * sun,fri /cmd`	5 PM on Sundays and Fridays
`3 12 * * sun /cmd`	12:03 PM every Sunday
`0 0 15 Jun,Jul,Aug * /cmd`	Midnight on 15th of Jun, Jul, Aug
`@reboot /cmd`	Once at system startup
`@daily /cmd`	Once per day at midnight
`@weekly /cmd`	Once per week on Sunday
`@monthly /cmd`	Once per month on the 1st

NOTE Cron security: Create /etc/cron.allow to whitelist users. Create /etc/cron.deny to blacklist. If cron.allow exists, only listed users can schedule.

anacron — Missed Job Recovery

anacron runs missed scheduled jobs after the system powers on. Ideal for non-24/7 systems.

Command / Option	Description
`/etc/anacrontab`	Anacron configuration file
`7 10 my-backup /bin/bash /scripts/backup.sh`	Format: period(days) delay(mins) ID command

NOTE Anacron format: period delay job-id command. Period=days, Delay=minutes to wait after boot before running.

systemd Timers — The Modern Cron

systemd timers replace cron with dependency-aware, logged, reliable job scheduling. Each timer requires a matching .service file.

Timer Types

Command / Option	Description
`OnBootSec=5min`	Monotonic: run 5 minutes after boot
`OnUnitActiveSec=1h`	Monotonic: run 1 hour after last activation
`OnCalendar=daily`	Real-time: run daily at midnight
`OnCalendar=Mon *-*-* 08:00:00`	Real-time: every Monday at 8 AM
`OnCalendar=*-*-01 04:00:00`	Real-time: 1st of every month at 4 AM
`OnCalendar=*-*-* *:00/15:00`	Real-time: every 15 minutes

Creating a Timer — Example (Daily Backup at 2 AM)

```
# /etc/systemd/system/backup.service
[Unit]
Description=Daily Website Backup
```

```
[Service]
Type=oneshot
ExecStart=/usr/local/bin/backup.sh
```

```
# /etc/systemd/system/backup.timer
[Unit]
Description=Run backup daily at 2AM

[Timer]
OnCalendar=*-*-* 02:00:00
Persistent=true
RandomizedDelaySec=300

[Install]
WantedBy=timers.target
```

Command / Option	Description
`systemctl daemon-reload`	Reload after creating/editing unit files
`systemctl enable --now backup.timer`	Enable and start the timer
`systemctl list-timers`	Show all active timers and next run time
`systemd-analyze calendar 'Mon *-*-* 08:00:00'`	Validate and test OnCalendar syntax
`journalctl -u backup.timer`	View timer execution logs
`journalctl -u backup.service`	View service execution logs

03. ACLs, Special Permissions & Shell Scripting

Access Control Lists (ACLs)

ACLs extend standard Linux permissions to grant access to specific users or groups beyond the file owner, group, and others.

Command / Option	Description
`setfacl -m u:ahmed:rw file.txt`	Grant user ahmed read+write on file.txt
`setfacl -m g:sales:r data/`	Grant group sales read on directory
`setfacl -R -m g:sales:r data/`	Apply ACL recursively to all files
`setfacl -m d:u:ahmed:rwx /mydir`	Set DEFAULT ACL (inherited by new files in dir)
`setfacl -d -m g:developers:rw projects/`	Default ACL for group on directory
`setfacl -x u:ahmed file.txt`	Remove ACL entry for user ahmed
`setfacl -b file.txt`	Remove ALL ACLs from file
`getfacl file.txt`	View ACLs on a file

NOTE When a file has ACLs, ls -l shows a '+' after the permissions string: -rw-rw-r--+

Special Permission Bits

Bit	Octal	Symbol	Set Command	Effect
SUID	4	s (owner)	`chmod 4755 file` or `chmod u+s file`	File runs with owner's privileges. Find: find / -perm /4000
SGID	2	s (group)	`chmod 2755 dir` or `chmod g+s dir`	Files in dir inherit group ID. File runs as group. Find: find / -perm /2000
Sticky Bit	1	t (others)	`chmod 1777 dir` or `chmod +t dir`	Only owner can delete their files in dir (like /tmp). Find: find / -perm /1000

Shell Scripting Fundamentals

Script Structure

```
#!/bin/bash              # Shebang — defines interpreter
set -e                   # Exit immediately on error
set -u                   # Error on unset variables
set -o pipefail          # Fail on pipeline errors

VARIABLE="value"         # Variable assignment (no spaces around =)
echo "Hello $VARIABLE"  # Double quotes: variable expansion
echo 'Literal $VARIABLE'  # Single quotes: no expansion
```

Command / Option	Description
`chmod +x script.sh`	Make script executable
`./script.sh`	Run script

Command / Option	Description
`bash script.sh`	Run without execute permission
`set -x`	Debug mode: print each command before running

Special Variables

Command / Option	Description
`$0`	Script name
`$1, $2, ...`	Positional arguments passed to script
`$#`	Total number of arguments
`$@`	All arguments as a list
`$?`	Exit status of last command (0=success, non-zero=fail)
`$$`	PID of the current script

Conditionals

```
if [ -f "/etc/passwd" ]; then
  echo "File exists"
elif [ -d "/tmp" ]; then
  echo "Directory exists"
else
  echo "Nothing found"
fi

# Short-circuit operators
cp data.txt /backup/ && echo "Success" || echo "Failed"
```

Command / Option	Description
`-f file`	File exists and is a regular file
`-d dir`	Directory exists
`-z str`	String is empty
`-n str`	String is not empty
`-eq / -ne`	Equal / Not equal (numeric)
`-lt / -gt`	Less than / Greater than (numeric)

Loops

```
# for loop
for server in app1 app2 db1; do
  echo "Pinging $server..."
  ping -c 1 $server
done

# while loop
count=1
while [ $count -le 5 ]; do
```

```
  echo "Count is $count"
  ((count++))
done
```

trap — Self-Cleaning Scripts

```
#!/bin/bash
temp_file=$(mktemp)
trap "rm -f $temp_file; echo 'Cleanup done.'" EXIT
echo "Working with $temp_file..."
# Even Ctrl+C triggers cleanup
```

getopts — Command-Line Flags

```
# Usage: ./adduser.sh -u john -g developers
while getopts "u:g:" opt; do
  case $opt in
    u) USERNAME="$OPTARG" ;;
    g) GROUP="$OPTARG" ;;
    *) echo "Usage: $0 -u <user> -g <group>"; exit 1 ;;
  esac
done
```

04. Storage: Partitions, LVM, RAID, Stratis & Encryption

Storage Inspection Commands

Command / Option	Description
`lsblk -la`	List all block devices in tree format
`blkid`	Show UUIDs and filesystem types for all devices
`df -hT`	Show disk usage with filesystem types
`du -sh /path`	Show total size of a directory
`mount | column -t`	Show all currently mounted filesystems
`findmnt`	Show mounts in tree format
`findmnt -x`	Check for mount errors
`lsof +D /mnt/data`	List open files on a mount point

Partitioning Tools Comparison

Tool	Table Type	Max Size	Best Use
`fdisk`	MBR	2 TB	Legacy systems, simple partitioning
`gdisk`	GPT	Exabytes	Disks > 2 TB, modern UEFI systems
`parted`	MBR/GPT	Both	Flexible, scriptable, supports both table types

fdisk — MBR Partitioning

Command / Option	Description
`fdisk -l /dev/sdb`	List partitions on sdb
`fdisk /dev/sdb`	Open interactive partitioning (n=new, d=delete, p=print, w=write)
`partprobe /dev/sdb`	Inform kernel of partition table changes

gdisk — GPT Partitioning

Command / Option	Description
`gdisk -l /dev/sdb`	Show partition table (shows if GPT is present)
`gdisk /dev/sdc`	Interactive GPT partitioning

parted — MBR & GPT

```
parted /dev/sdb
select /dev/sdb
mklabel gpt              # or msdos for MBR
mkpart primary 1 1024    # 1 MB to 1024 MB
p                        # print table
quit
```

Filesystems

Command / Option	Description
`mkfs.xfs -L Data /dev/sdb2`	Create XFS with label 'Data'
`mkfs.ext4 /dev/sdb3`	Create ext4 filesystem
`xfs_info /dev/sdb2`	Show XFS filesystem info
`xfs_admin -U generate /dev/sdb2`	Generate new UUID for XFS filesystem
`blkid /dev/sdb2`	Show filesystem UUID, label, and type
`wipefs -a /dev/sdb`	Wipe all filesystem signatures from device

NOTE XFS: max file size 8 EB, cannot shrink. ext4: max file size 2 TB, can shrink. XFS is the RHEL default.

Mounting Filesystems

Temporary Mount

Command / Option	Description
`mount /dev/sdb1 /data/sdbdata1`	Mount device to directory
`mount -t xfs /dev/sdb1 /data/`	Explicit filesystem type
`umount /data/sdbdata1`	Unmount (close all files first)

Persistent Mount (/etc/fstab)

```
# Format: device  mountpoint  fstype  options  dump  pass
UUID=abc123  /data  xfs  defaults  0  2
LABEL=Data   /data  xfs  defaults  0  2
/dev/sdb1    /data  xfs  defaults  0  2
```

Command / Option	Description
`blkid`	Get UUIDs for fstab entries
`ls -l /dev/disk/by-uuid/`	View UUIDs as symlinks
`ls -l /dev/disk/by-label/`	View labels as symlinks
`mount -a`	Activate all fstab entries
`findmnt --verify`	Verify fstab for errors

NOTE fstab pass field: 0=no fsck, 1=root filesystem, 2=other filesystems (checked after root)

LVM — Logical Volume Management

LVM abstracts physical storage into flexible logical volumes. Order: Physical Volumes → Volume Groups → Logical Volumes.

LVM Setup

```
1  dnf install lvm2  — install LVM tools
2  pvcreate /dev/sdb /dev/sdc  — create physical volumes
```

```
3  vgcreate vg1 /dev/sdb /dev/sdc  — create volume group
4  lvcreate -n lv1 -L 2G vg1  — create 2GB logical volume
5  mkfs.xfs -L lvdata /dev/vg1/lv1  — format the LV
6  mkdir /lvm1 && mount /dev/vg1/lv1 /lvm1  — mount it
7  Add to /etc/fstab for persistence
```

LVM Management Commands

Command / Option	Description
`pvdisplay / pvs`	Show physical volumes
`vgdisplay / vgs`	Show volume groups
`lvdisplay / lvs`	Show logical volumes
`vgextend vg1 /dev/sdd`	Add a new disk to volume group
`lvextend -L +50G -r /dev/vg1/lv1`	Extend LV by 50GB (auto-resizes filesystem)
`lvextend -l 100%FREE -r /dev/vg1/lv1`	Extend to use all free space
`xfs_growfs /lvm1/`	Grow XFS filesystem manually (if -r not used)
`lvremove /dev/vg1/lv1`	Remove logical volume
`vgremove vg1`	Remove volume group
`pvremove /dev/sdb`	Remove physical volume

NOTE PE-based sizing: lvcreate -l 50 -n lv1 vg1 (creates LV of 50 Physical Extents). Use vgdisplay to find PE size.

Software RAID (mdadm)

Command / Option	Description
`cat /proc/mdstat`	View RAID status
`mdadm -C /dev/md0 -l 5 -n 3 /dev/sdb /dev/sdc /dev/sdd`	Create RAID-5 with 3 disks
`mdadm --create /dev/md0 -l 5 -n 3 /dev/sd[c-e]1 -x 1 /dev/sdf1`	RAID-5 with 1 hot spare
`mkfs.xfs -L Rdata /dev/md0`	Format RAID device
`mdadm --detail --scan >> /etc/mdadm.conf`	Make RAID config persistent
`mdadm --add /dev/md0 /dev/sde`	Add spare disk to array
`mdadm --grow /dev/md0 --raid-devices=4`	Increase active disks
`mdadm /dev/md0 --fail /dev/sdb`	Simulate disk failure (testing)
`watch cat /proc/mdstat`	Monitor RAID rebuild in real-time
`mdadm --stop /dev/md0`	Stop RAID device

Stratis — Advanced Storage Management

Stratis simplifies thin-provisioning, snapshots, and monitoring. Pools contain filesystems that auto-grow.

```
dnf install stratisd stratis-cli && systemctl enable --now stratisd
wipefs -a /dev/sdh /dev/sdi  — clean disks first
stratis pool create pool_1 /dev/sdh /dev/sdi  — create pool from disks
stratis fs create pool_1 fs_1  — create filesystem in pool
blkid -p /dev/stratis/pool_1/fs_1  — get UUID for fstab
Add to /etc/fstab: UUID=... /mount/point xfs defaults,x-
systemd.requires=stratisd.service 0 0
```

Command / Option	Description
stratis pool list	List pools
stratis fs list	List filesystems
stratis blockdev list	List block devices in pools
stratis pool add-data pool_1 /dev/sdf	Add disk to existing pool
stratis fs snapshot pool_1 fs_1 snap_1	Create snapshot
stratis fs destroy pool_1 fs_1	Delete filesystem
stratis pool destroy pool_1	Delete pool

LUKS Disk Encryption

```
dnf install -y cryptsetup  — install cryptsetup
cryptsetup luksFormat /dev/sda1  — initialize LUKS (writes header)
cryptsetup luksOpen /dev/sda1 securedata  — unlock as /dev/mapper/securedata
mkfs.ext4 /dev/mapper/securedata  — create filesystem
Add to /etc/fstab: /dev/mapper/securedata /mnt/encrypted ext4 defaults 0 2
Add to /etc/crypttab: securedata /dev/sda1 none luks  — auto-unlock at boot
```

Command / Option	Description
cryptsetup luksDump /dev/sda1	Show LUKS header info and key slots
cryptsetup status securedata	Show mapped device status
cryptsetup luksAddKey /dev/sda1	Add a second passphrase/key
cryptsetup luksRemoveKey /dev/sda1	Remove a passphrase
cryptsetup luksHeaderBackup /dev/sda1 --header-backup-file /root/backup.img	Backup LUKS header
cryptsetup luksHeaderRestore /dev/sda1 --header-backup-file /root/backup.img	Restore LUKS header
umount /mnt/encrypted && cryptsetup luksClose securedata	Unmount and re-lock

Swap Space Management

Command / Option	Description
`free -h`	Show RAM and swap usage
`swapon -s`	List active swap spaces
`cat /proc/swaps`	Show swap usage details
`mkswap /dev/sdb5`	Initialize partition as swap
`swapon /dev/sdb5`	Enable swap partition
`swapoff /dev/sdb5`	Disable swap partition
`swapon -p 10 /dev/sdb1`	Enable with priority 10
`swapoff -a`	Disable all swap spaces
`dd if=/dev/zero of=swapfile bs=1G count=1`	Create a 1GB swap file
`mkswap swapfile && chmod 0600 swapfile`	Prepare swap file
`swapon swapfile`	Enable swap file

05. System Logging — journald & Remote rsyslog

systemd-journald

journald is systemd's logging subsystem. Logs are stored in binary format and queried with journalctl.

Basic journalctl Commands

Command / Option	Description
`journalctl -f`	Follow live system logs (like tail -f)
`journalctl -u sshd -f`	Follow logs for a specific service
`journalctl -b`	Logs from current boot
`journalctl --list-boots`	List all recorded boot sessions
`journalctl -b -1`	Logs from previous boot
`journalctl -r`	Show logs in reverse (newest first)
`journalctl -n 100`	Show last 100 log entries
`journalctl -k`	Show kernel messages only

Filtering by Time

Command / Option	Description
`journalctl --since today`	All logs since today midnight
`journalctl --since yesterday`	All logs since yesterday
`journalctl --since '1 hour ago'`	Logs from last hour
`journalctl --since '2025-08-21 08:00' --until '2025-08-22'`	Logs in a specific time window

Filtering by Priority

Command / Option	Description
`journalctl -p err`	Show errors and above (crit, alert, emerg)
`journalctl -p warn`	Show warnings and above
`journalctl -u sshd -p err --since yesterday`	SSH errors since yesterday

Filtering by Process / User

Command / Option	Description
`journalctl _PID=1234`	Logs from a specific PID
`journalctl _UID=1000`	Logs from a specific user
`journalctl _COMM=su --since '24 hours ago'`	Logs from 'su' command in last 24 hours
`journalctl --disk-usage`	Show disk space used by journal

Persistent Journal Storage

```
mkdir -p /var/log/journal  — create persistent log directory
Edit /etc/systemd/journald.conf → set: Storage=persistent
systemctl restart systemd-journald
```

> **NOTE** By default, journald uses volatile storage (lost on reboot). Creating /var/log/journal enables persistence.

Redirecting Logs to a Different Volume

```
mount -t xfs /dev/sda1 /x1  — mount new volume temporarily
cp -r /var/log/* /x1  — backup existing logs
rm -rf /var/log/*  — clear /var/log directory
mount -t xfs /dev/sda1 /var/log  — mount new volume to /var/log
echo '/dev/sda1 /var/log xfs defaults 0 2' >> /etc/fstab  — make persistent
cp -r /x1/* /var/log/  — restore backed-up logs
```

Remote Log Forwarding (node1 → node2)

On node1 (Source Server)

```
# /etc/rsyslog.d/remote.conf
*.* @@node2:514    # TCP forwarding
*.* @node2:514     # UDP forwarding
*.* ~              # Stop local logging (optional, saves space)
systemctl restart rsyslog
```

On node2 (Destination Server)

```
# /etc/rsyslog.d/receive.conf
module(load="imtcp")
input(type="imtcp" port="514")

template(name="RemoteLogs" type="string"
  string="/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log")

if ($fromhost-ip != '127.0.0.1') then {
  action(type="omfile" dynaFile="RemoteLogs")
  stop
}
```

```
mkdir -p /var/log/remote && chmod 755 /var/log/remote
systemctl restart rsyslog
firewall-cmd --permanent --add-port=514/tcp && firewall-cmd --reload
On node1: logger 'Test message'  — send test
```

```
5  On node2: tail -f /var/log/remote/node1/  — verify receipt
```

06. Systemd Advanced — Units, Sockets, Mounts & Timers

systemd Unit Types

Extension	Type	Purpose	Example
`.service`	Service	Traditional daemon/process	sshd.service, nginx.service
`.socket`	Socket	Network listeners — on-demand activation	sshd.socket
`.timer`	Timer	Scheduled tasks — cron replacement	logrotate.timer
`.mount`	Mount	Filesystem mounts with dependencies	var-lib-docker.mount
`.target`	Target	Group of units — like runlevels	multi-user.target
`.path`	Path	Watch filesystem path for changes	cups.path

Command / Option	Description
`systemctl list-units --type=service`	List all loaded services
`systemctl list-units --type=socket`	List all loaded sockets
`systemctl list-units --type=timer`	List all loaded timers
`systemctl list-unit-files --type=service --state=enabled`	List enabled services
`systemctl cat nginx.service`	Show full unit file
`systemctl show nginx.service`	Show all settable properties
`systemctl edit nginx.service`	Create override (drop-in) file
`systemctl daemon-reload`	Reload after editing unit files

Socket Activation

Socket activation starts services only when a connection arrives — reducing RAM usage for rarely-used services.

Command / Option	Description
`systemctl enable --now sshd.socket`	Enable SSH socket activation (starts sshd only on connection)
`systemctl disable sshd.socket`	Disable socket activation
`systemctl enable --now sshd.service`	Re-enable persistent sshd service

NOTE Socket activation benefit: sshd.service uses ~10MB RAM constantly. Socket activation = 0 MB until first connection.

Systemd Mount Units

Mount units provide dependency-aware mounting — ensuring required services start before the mount.

NOTE CRITICAL: The filename MUST match the mount point path with slashes replaced by dashes. /data → data.mount | /mnt/backup → mnt-backup.mount | /var/www/html → var-www-html.mount

```
# /etc/systemd/system/mnt-data.mount
[Unit]
Description=Mount Data Disk
After=blockdev@dev-sdb1.target

[Mount]
What=/dev/sdb1
Where=/mnt/data
Type=xfs
Options=defaults,noatime

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

Command / Option	Description
`systemctl daemon-reload`	Reload after creating unit file
`systemctl enable --now mnt-data.mount`	Enable and mount
`systemctl list-units --type=mount`	List all mount units
`journalctl -u mnt-data.mount`	View mount unit logs

Service Resilience (Failure Recovery)

Add these directives to your [Service] section for automatic recovery:

```
[Service]
Restart=on-failure        # Restart if it crashes
RestartSec=5s             # Wait 5 seconds before restart
WatchdogSec=30s           # Kill and restart if hung for 30s
StartLimitBurst=5         # Max 5 restart attempts
StartLimitInterval=300    # ...within 300 seconds
```

07. File Transfer — rsync, scp & NFS/AutoFS

rsync — Efficient Synchronization

rsync transfers only changed blocks, making it ideal for backups and large-scale synchronization.

Command / Option	Description
`rsync -zvh backup.tar /tmp/backups/`	Local file copy with compression and verbose output
`rsync -avzh /root/rpmpkgs /tmp/backups/`	Sync directory locally (archive mode)
`rsync -avz rpmpkgs/ root@192.168.0.101:/home/`	Push directory to remote server
`rsync -avzh root@192.168.0.100:/home/source/ /tmp/myrpms`	Pull from remote to local
`rsync -avzhe ssh --progress /home/ root@192.168.0.100:/root/`	With progress display
`rsync --max-size='200k' -avzhe ssh /var/lib/rpm/ root@host:/tmp/`	Limit file size
`rsync --remove-source-files -zvh backup.tar /tmp/`	Delete source after transfer
`rsync --bwlimit=100 -avzhe ssh /var/lib/rpm/ root@host:/tmp/`	Limit bandwidth to 100KB/s

scp — Secure Copy Protocol

Command / Option	Description
`scp user@10.0.0.4:/etc/passwd .`	Download file from remote to current dir
`scp -P 1515 root@10.0.0.4:/etc/passwd .`	Download using non-standard SSH port
`scp file01 file02 file03 /repo`	Copy multiple files locally
`scp -r /local/dir user@host:/remote/dir`	Recursively copy directory to remote
`scp -C file.tar.gz user@host:/path/`	Copy with compression enabled
`scp user@host1:/file.txt user@host2:/path/`	Copy between two remote hosts

NFS — Network File System

NFS Server Setup

```
1  yum install -y nfs-utils && systemctl enable --now nfs-server
2  mkdir -p /srv/nfs/share && chmod 777 /srv/nfs/share
3  nano /etc/exports  →  /srv/nfs/share 192.168.1.0/24(rw,sync,no_root_squash)
4  exportfs -r  — reload exports  |  exportfs -v  — verify
5  firewall-cmd --permanent --add-service=nfs && firewall-cmd --reload
```

NFS Client Setup

```
yum install -y nfs-utils
mkdir -p /mnt/nfs-share
mount -t nfs 192.168.1.100:/srv/nfs/share /mnt/nfs-share
df -h | grep nfs  — verify mount
nano /etc/fstab  →  192.168.1.100:/srv/nfs/share /mnt/nfs-share nfs defaults 0 0
mount -a  — activate fstab
```

AutoFS — Automatic NFS Mounting

AutoFS mounts filesystems on-demand and unmounts them after a period of inactivity.

```
yum install -y autofs
nano /etc/auto.master  →  /mnt/nfs-auto /etc/auto.nfs
nano /etc/auto.nfs  →  share -fstype=nfs,rw 192.168.1.100:/srv/nfs/share
systemctl enable --now autofs
ls /mnt/nfs-auto/share  — access triggers auto-mount
```

Command / Option	Description
/etc/auto.master	Master map: defines mount points and map files
/etc/auto.nfs	Map file: defines share names and NFS sources
/etc/autofs.conf	AutoFS config: set timeout = 60 for 60-second idle unmount

08. Managing SELinux

SELinux Overview

SELinux (Security-Enhanced Linux) provides Mandatory Access Control (MAC) on top of standard Linux permissions. Even root cannot bypass SELinux policy.

Mode	Behavior
Enforcing	SELinux enforces all policies. Violations are denied AND logged. Default in RHEL.
Permissive	Violations are LOGGED but NOT denied. Use for troubleshooting SELinux issues.
Disabled	SELinux is completely off. Requires relabeling the entire filesystem on re-enable. Avoid in production.

Core SELinux Commands

Command / Option	Description
`getenforce`	Show current SELinux mode
`setenforce 0`	Temporarily set to Permissive (until reboot)
`setenforce 1`	Temporarily set to Enforcing
`sestatus`	Show detailed SELinux status and policy info
`vim /etc/sysconfig/selinux`	Edit persistent mode (SELINUX=enforcing/permissive/disabled)

NOTE Changing SELINUX=disabled requires a full reboot AND filesystem relabeling. Set SELINUX=permissive for troubleshooting, not disabled.

Managing SELinux Port Contexts

Command / Option	Description
`dnf install -y policycoreutils-python-utils`	Install semanage tools
`semanage port -l`	List all SELinux port contexts
`semanage port -l | grep http`	Check if port is allowed for HTTP
`semanage port -a -t http_port_t -p tcp 5555`	Allow port 5555 for HTTP
`semanage port -a -t ssh_port_t -p tcp 1414`	Allow port 1414 for SSH
`semanage port -a -t unreserved_port_t -p tcp 6789`	Allow port 6789 for any service
`semanage port -d -t http_port_t -p tcp 5555`	Remove custom port definition
`semanage port -l | grep sshd`	Verify SSH port configuration

SELinux File Contexts

Command / Option	Description
`ls -Z /var/www/html/`	Show file SELinux contexts
`ps -eZ | grep httpd`	Show process contexts
`chcon -t httpd_sys_content_t file.html`	Change file context temporarily
`restorecon -v file.html`	Restore default context from policy
`restorecon -Rv /var/www/html/`	Recursively restore all contexts
`semanage fcontext -a -t httpd_sys_content_t '/mydir(/.*)?'`	Add persistent context rule

NOTE TIP: Use chcon for temporary fixes, semanage fcontext + restorecon for permanent solutions.

Troubleshooting SELinux Denials

Command / Option	Description
`ausearch -m avc -ts recent`	Search recent SELinux denial messages
`journalctl -t setroubleshoot`	View SELinux troubleshoot messages
`sealert -l <alert-id>`	Get detailed explanation and suggested fix
`audit2allow -a`	Generate allow rules from denials (review before applying)

09. Managing Firewall with Firewalld

Firewalld Architecture

Firewalld uses zones to define trust levels for network connections. Every interface/source belongs to a zone, and each zone has its own set of rules.

Command / Option	Description
`systemctl status firewalld`	Check firewall service status
`systemctl enable --now firewalld`	Enable and start firewalld
`firewall-cmd --state`	Show if firewalld is running
`firewall-cmd --list-all`	Show all rules for default zone

Managing Services

Command / Option	Description
`firewall-cmd --list-services`	List allowed services
`firewall-cmd --add-service=http`	Temporarily allow HTTP (lost on reload)
`firewall-cmd --add-service=http --permanent`	Permanently allow HTTP
`firewall-cmd --remove-service=http --permanent`	Remove HTTP service
`firewall-cmd --reload`	Apply permanent rules to runtime
`firewall-cmd --runtime-to-permanent`	Save current runtime rules as permanent

Managing Ports

Command / Option	Description
`firewall-cmd --list-ports`	List all open ports
`firewall-cmd --add-port=5050/tcp --permanent`	Permanently open TCP port 5050
`firewall-cmd --add-port=514/tcp --permanent`	Open port for remote syslog
`firewall-cmd --remove-port=5050/tcp --permanent`	Remove port rule
`firewall-cmd --reload`	Apply changes

Zones

Command / Option	Description
`firewall-cmd --get-zones`	List all available zones
`firewall-cmd --get-default-zone`	Show current default zone
`firewall-cmd --set-default-zone=internal`	Change default zone

Command / Option	Description
`firewall-cmd --list-all-zones`	Show rules for ALL zones
`firewall-cmd --zone=public --add-service=http`	Add service to public zone
`firewall-cmd --zone=public --list-services`	List services in public zone
`firewall-cmd --zone=public --add-interface=eth1`	Assign interface to zone

Zone	Trust Level
`drop`	All incoming dropped, no reply. Most restrictive.
`block`	All incoming rejected with icmp-admin-prohibited.
`public`	Default zone. Don't trust other hosts. Basic services allowed.
`work`	Mostly trusted — work machines on same network.
`home`	Mostly trusted — home machines.
`internal`	Trusted internal network. Most services allowed.
`trusted`	All connections accepted. Most permissive.

NOTE Service definition XML files are in /usr/lib/firewalld/services/. Custom services go in /etc/firewalld/services/.

10. Performance Tuning (tuned) & KVM Virtualization

tuned — Adaptive System Tuning

The tuned daemon automatically adjusts kernel and system parameters based on current workload using predefined profiles.

Command / Option	Description
`systemctl status tuned`	Check tuned service status
`dnf install tuned && systemctl enable --now tuned`	Install and start tuned
`tuned-adm active`	Show currently active profile
`tuned-adm list`	List all available profiles
`tuned-adm recommend`	Show recommended profile for this system
`tuned-adm profile throughput-performance`	Apply throughput-performance profile
`tuned-adm profile balanced`	Apply balanced profile
`tuned-adm profile powersave`	Apply power-saving profile
`tuned-adm off`	Disable all dynamic tuning

Profile	Best For
`throughput-performance`	Servers requiring maximum I/O and network throughput
`latency-performance`	Real-time applications requiring low latency
`virtual-host`	KVM/QEMU hypervisor hosts
`virtual-guest`	VMs running inside hypervisors
`balanced`	General-purpose desktops and mixed workloads
`powersave`	Laptops and low-power environments
`desktop`	Desktop systems — improved responsiveness
`network-latency`	Low-latency network applications

KVM — Kernel-based Virtual Machine

Installation

```
dnf install -y qemu-kvm libvirt libvirt-daemon libvirt-daemon-driver-qemu virt-install virt-manager
systemctl enable --now libvirtd
```

Create a VM (virt-install)

```
virt-install \
  --name test-vm \
  --memory 2048 \
  --vcpus 2 \
  --disk size=20 \
  --os-variant rhel9.0 \
```

```
  --cdrom /path/to/rhel9.iso \
  --boot uefi \
  --graphics vnc

# Without OS (import):
virt-install --name test-vm --memory 512 --vcpus 1 --disk size=8 --os-variant
generic --boot uefi --noautoconsole --import
```

virsh — VM Management CLI

Command / Option	Description
`virsh list --all`	List all VMs (running and stopped)
`virsh start vm1`	Start VM named vm1
`virsh shutdown vm1`	Graceful shutdown of vm1
`virsh destroy vm1`	Force stop vm1 (like pulling the power cord)
`virsh reboot vm1`	Reboot vm1
`virsh autostart vm1`	Start vm1 automatically at host boot
`virsh autostart vm1 --disable`	Disable autostart for vm1
`virsh net-list`	List virtual networks
`virsh dominfo vm1`	Show VM configuration details
`virt-manager`	Open graphical VM management interface

11. Containers with Podman on RHEL

Container Concepts

Concept	Description
Image	Read-only blueprint containing code, runtime, libraries, and config. The template for containers.
Container	A running instance of an image — isolated user-space process with its own filesystem, network, and PID space.
Registry	Storage for container images. Examples: registry.redhat.io, quay.io, Docker Hub.
Pod	Group of one or more containers sharing network/storage (Kubernetes concept). Podman supports pods natively.
Volume	Persistent storage that survives container removal. Mounted into the container at runtime.
Orchestration	Automates deployment, scaling, and management of containers (Kubernetes, OpenShift).

Podman Installation & Images

Command / Option	Description
`dnf -y install podman`	Install Podman
`podman --version`	Verify installation
`podman search fedora`	Search registries for fedora images
`podman pull nginx`	Download nginx image
`podman images`	List downloaded images
`podman rmi ubuntu`	Remove image
`podman rmi -f $(podman images -aq)`	Remove all images
`podman inspect nginx:latest`	Show image details and layers

Running Containers

Command / Option	Description
`podman run hello-world`	Run and exit (test)
`podman run -d --name my-nginx -p 8080:80 nginx`	Run nginx in background, map port 8080→80
`podman run -it ubuntu bash`	Interactive Ubuntu terminal
`podman run -it -d --name myfirst ubuntu`	Detached interactive container
`podman run -d --restart=always --name my-nginx -p 8080:80 nginx`	Auto-restart on failure
`podman run -d --name my-mariadb -v my-data:/var/lib/mysql mariadb`	With persistent volume

NOTE Port mapping: -p HOST_PORT:CONTAINER_PORT | -d = detached (background) | -it = interactive terminal

Managing Containers

Command / Option	Description
`podman ps`	List running containers
`podman ps -a`	List all containers (including stopped)
`podman stop myfirst`	Gracefully stop container
`podman start myfirst`	Start a stopped container
`podman restart myfirst`	Stop and start container
`podman rm myapp`	Remove stopped container
`podman rm -f myapp`	Force-remove running container
`podman rm -f $(podman ps -aq)`	Remove all containers

Interacting with Running Containers

Command / Option	Description
`podman exec myfirst /bin/ps -aux`	Run command in running container
`podman exec -it myfirst bash`	Open interactive shell in container
`podman attach myfirst`	Re-attach to running container
`Ctrl-p then Ctrl-q`	Detach from container without stopping it
`podman logs my-nginx`	View container logs
`podman logs -f my-nginx`	Follow container logs live
`podman top myfirst`	Show processes in container
`podman stats myfirst`	Live resource usage (CPU, RAM, I/O)
`podman events --since '1h'`	Show container events from last hour

Building Images with Buildah

```
# Build a custom Apache image on RHEL
containerID=$(buildah from fedora)
buildah run $containerID -- dnf install httpd -y
buildah config --entrypoint '["httpd", "-D", "FOREGROUND"]' $containerID
buildah commit $containerID my-apache:latest

# Then run it:
podman run -d -p 8080:80 my-apache:latest
```

NOTE Podman is daemonless and rootless — runs without a background daemon. Buildah creates OCI-compliant images compatible with Docker.