RHCSA Certification Study Guide

Comprehensive Study Materials for Red Hat Certified System Administrator Exam

RHCSA Study Repository

Study materials for RHCSA certification preparation

Table of Contents

1. RHCSA Certification Study Guide 11
    - 1.1 Study Materials 11
        - RHCSA Synthesis Modules 11
        - Quick References 11
    - 1.2 Anki Flashcards 12
        - Flashcard Categories 12
    - 1.3 Lab Environment 12
    - 1.4 Study Workflow 12
    - 1.5 Exam Preparation Tips 12
    - 1.6 Repository Structure 13
2. Study Modules 14
    - 2.1 RHCSA Synthesis - Unified Knowledge Base 14
        - Overview 14
        - Study Approach 14
        - Module Index 15
        - Quick Navigation 15
        - Study Progress Tracking 16
        - Additional Resources 16
    - 2.2 00 - RHCSA Exam Overview & Strategy 18
        - 1. Executive Summary 18
        - 2. RHCSA Exam Fundamentals 18
        - 3. Strategic Approach 19
        - 4. Exam Environment Deep Dive 20
        - 5. Task Categories and Strategies 21
        - 6. Common Exam Mistakes to Avoid 23
        - 7. Verification Techniques 24
        - 8. Lab Environment Setup for Practice 25

9. Final Preparation Checklist 25

10. Exam Day Tactics 26

Summary 27

2.3 01 - System Installation & Initial Configuration 28

1. Executive Summary 28

2. Conceptual Foundation 28

3. Command Mastery 29

4. Installation Workflows 30

5. Configuration Deep Dive 32

6. Hands-On Labs 33

7. Troubleshooting Playbook 37

8. Quick Reference Card 39

9. Knowledge Check 40

10. Exam Strategy 41

Summary 42

2.4 02 - File Management & Text Processing 43

1. Executive Summary 43

2. Conceptual Foundation 43

3. Command Mastery 44

4. Procedural Workflows 46

5. Configuration Deep Dive 47

6. Hands-On Labs 48

7. Troubleshooting Playbook 52

8. Quick Reference Card 54

9. Knowledge Check 55

10. Exam Strategy 56

Summary 57

2.5 03 - User & Group Management 58

1. Executive Summary 58

2. Conceptual Foundation 58

3. Command Mastery 60

4. Procedural Workflows 62

5. Configuration Deep Dive 64

6. Hands-On Labs 66

7. Troubleshooting Playbook 72

8. Quick Reference Card 75

9. Knowledge Check 76

10. Exam Strategy 77

Summary 78

2.6 04 - File Permissions & Access Control 79

1. Executive Summary 79

2. Conceptual Foundation 79

3. Command Mastery 80

4. Procedural Workflows 81

5. Configuration Deep Dive 83

6. Hands-On Labs 84

7. Troubleshooting Playbook 88

8. Quick Reference Card 89

9. Knowledge Check 90

10. Exam Strategy 91

Summary 91

Supplementary Reference: ACLs (Not on RHEL 10 Exam) 92

Supplementary Reference: Special Permissions (Not on RHEL 10 Exam) 92

2.7 05 - Process & Service Management 94

1. Executive Summary 94

2. Conceptual Foundation 94

3. Command Mastery 96

4. Procedural Workflows 98

5. Configuration Deep Dive 101

6. Hands-On Labs 102

7. Troubleshooting Playbook 112

8. Quick Reference Card 116

9. Knowledge Check 117

10. Exam Strategy 117

Summary 118

2.8 06 - Package Management 120

1. Executive Summary 120

2. Conceptual Foundation 120

3. Command Mastery 121

4. Procedural Workflows 123

5. Configuration Deep Dive 126

6. Hands-On Labs 127

7. Troubleshooting Playbook 137

8. Quick Reference Card 141

9. Knowledge Check 142

10. Exam Strategy 143

Summary 143

2.9 07 - Storage & LVM Management 145

1. Executive Summary 145

2. Conceptual Foundation 145

3. Command Mastery 147

4. Procedural Workflows 150

5. Configuration Deep Dive 155

6. Hands-On Labs 157

7. Troubleshooting Playbook 168

8. Quick Reference Card 171

9. Knowledge Check 172

10. Exam Strategy 173

Summary 173

2.10 08 - Network Configuration 175

1. Executive Summary 175

2. Conceptual Foundation 175

3. Command Mastery 177

4. Procedural Workflows 179

5. Configuration Deep Dive 182

6. Hands-On Labs 184

7. Troubleshooting Playbook 196

8. Quick Reference Card 199

9. Knowledge Check 200

10. Exam Strategy 201

Summary 202

2.11 09 - SELinux Management 204

1. Executive Summary 204

2. Conceptual Foundation 204

3. Command Mastery 205

4. Procedural Workflows 208

5. Configuration Deep Dive 212

6. Hands-On Labs 214

7. Troubleshooting Playbook 223

8. Quick Reference Card 227

9. Knowledge Check 228

10. Exam Strategy 229

Summary 230

2.12 10 - Firewall Configuration 231

1. Executive Summary 231

2. Conceptual Foundation 231

3. Command Mastery 232

4. Procedural Workflows 236

5. Configuration Deep Dive 239

6. Hands-On Labs 241

7. Troubleshooting Playbook 255

8. Quick Reference Card 259

9. Knowledge Check 260

10. Exam Strategy 261

Summary 262

2.13 Module 11: Boot Process & GRUB Configuration 263

1. Learning Objectives 263

2. Key Concepts 263

3. Essential Commands 264

4. Asghar Ghori's Approach 265

5. Sander van Vugt's Approach 266

6. Command Examples and Scenarios 267

7. Lab Exercises 268

8. Troubleshooting Common Issues 270

9. Best Practices 272

10. Integration with Other RHCSA Topics 273

2.14 Module 12: Logging & Monitoring 275

1. Learning Objectives 275

2. Key Concepts 275

3. Essential Commands 276

4. Asghar Ghori's Approach 278

5. Sander van Vugt's Approach 280

6. Command Examples and Scenarios 281

7. Lab Exercises 282

8. Troubleshooting Common Issues 284

9. Best Practices 286

10. Integration with Other RHCSA Topics 287

2.15 Module 13: Scheduled Tasks & Automation 289

1. Learning Objectives 289

2. Key Concepts 289
3. Essential Commands 290
4. Asghar Ghori's Approach 292
5. Sander van Vugt's Approach 294
6. Command Examples and Scenarios 296
7. Lab Exercises 297
8. Troubleshooting Common Issues 299
9. Best Practices 301
10. Integration with Other RHCSA Topics 302
2.16 14 - Flatpak Software Management 304
1. Executive Summary 304
2. Conceptual Foundation 304
3. Command Mastery 306
4. Procedural Workflows 307
5. Configuration Deep Dive 309
6. Hands-On Labs 310
7. Troubleshooting Playbook 316
8. Quick Reference Card 318
9. Knowledge Check 319
10. Exam Strategy 320
Summary 321
2.17 Module 15: System Troubleshooting & Recovery 323
1. Learning Objectives 323
2. Key Concepts 323
3. Essential Commands 324
4. Asghar Ghori's Approach 326
5. Sander van Vugt's Approach 327
6. Command Examples and Scenarios 329
7. Lab Exercises 330
8. Troubleshooting Common Issues 333

9. Best Practices 335

10. Integration with Other RHCSA Topics 336

3. Quick References 338

3.1 RHCSA Exam Quick Reference 338

Essential Acronyms & Terms 338

Pre-Exam Checklist 338

File Management & Text Processing 339

User and Group Management 343

File Permissions & Access Control 345

Package Management 349

Storage and LVM Management 353

Network Configuration 357

SELinux Management 359

Firewall Management 364

Services and Systemd Management 366

Boot Process & GRUB Configuration 368

Logging & System Monitoring 373

NFS and AutoFS 378

Flatpak Software Management 384

Scheduled Tasks & Automation 386

Emergency Recovery Procedures 393

SSH & Remote Access 394

Shell Environment & Scripting Basics 400

Quick Verification Commands 409

Last-Minute Exam Reminders 410

3.2 RHCSA Command Reference by Topic Area 411

System Information and Management 411

File System and Storage Management 412

Text Processing and File Content 413

Permissions and Security 415

User and Group Management 416

Process and Job Management 418

System Services and Systemd 419

Logging and Monitoring 420

Network Configuration and Management 420

Network File System (NFS) and AutoFS 422

Package Management 426

Storage Management and File Systems 427

Firewall Management 430

SELinux Management 430

Boot Process and GRUB 432

Scheduled Tasks 432

Flatpak Software Management 433

SSH and Remote Access 434

3.3 RHCSA Acronyms and Glossary 435

Quick Navigation 435

Alphabetical Index 435

Acronyms by Category 441

Key Non-Acronym Terms 446

3.4 RHCSA Study Guide Summary: Topics and Commands from Both EPUBs 448

Book Structure Overview 448

Topic-by-Topic Breakdown with Lab Exercises 450

Command Summary by Category 465

1. RHCSA Certification Study Guide

Welcome to the comprehensive Red Hat Certified System Administrator (RHCSA) study repository. This resource provides everything you need to prepare for the RHCSA exam, including practical lab scenarios, comprehensive study materials, and memorization aids.

PDF Download: Download the complete study guide as PDF

1.1 Study Materials

Complete knowledge base with 15 detailed modules covering all RHCSA exam objectives:

- **Module 00**: Exam Overview - Strategy and format guide
- **Module 01**: System Installation - Installation and initial configuration
- **Module 02**: File Management - File operations and navigation
- **Module 03**: User & Group Management - Account administration
- **Module 04**: File Permissions - Access controls and security
- **Module 05**: Process & Service Management - System management
- **Module 06**: Package Management - Software installation
- **Module 07**: Storage & LVM - Storage management
- **Module 08**: Networking - Network configuration
- **Module 09**: SELinux - Security Enhanced Linux
- **Module 10**: Firewall - Network security
- **Module 11**: Boot & GRUB - System boot process
- **Module 12**: Logging & Monitoring - System monitoring
- **Module 13**: Scheduled Tasks - Automation
- **Module 14**: Flatpak Management - Flatpak software management
- **Module 15**: Troubleshooting - Problem resolution

Quick References

- **Exam Quick Reference** - Essential commands for exam day
- **Command Reference by Topic** - Commands organized by functional area
- **RHCSA Acronyms & Glossary** - Comprehensive terminology guide
- **Ebook Summary** - Analysis from major RHCSA study books

1.2 Anki Flashcards

Import the comprehensive flashcard deck for spaced repetition learning:

- **Location**: anki/rhcsa_deck.csv
- **Cards**: 169 essential commands and concepts
- **Topics**: All RHCSA exam objectives with practical examples

Flashcard Categories

- User & Group Management
- File Operations & Permissions
- System Services (systemd)
- Storage & LVM
- SELinux & Security
- Firewall Configuration
- Networking
- Flatpak Management

1.3 Lab Environment

Set up hands-on practice environment using Vagrant:

- **Location**: vagrant/ directory
- **VMs**: RHEL 10 instances (rhel10a, rhel10b)
- **Features**: Automated provisioning, storage configuration, networking
- **Prerequisites**: Vagrant, VirtualBox, Red Hat Developer subscription

1.4 Study Workflow

1. **Start with Synthesis Modules** - Begin with comprehensive topic coverage
2. **Practice with Anki Deck** - Use spaced repetition for command memorization
3. **Use Quick References** - Keep handy during study sessions
4. **Hands-on Practice** - Use Vagrant VMs for real-world scenarios
5. **Focus on Verification** - Always test and verify your configurations

1.5 Exam Preparation Tips

- **Time Management**: Practice under exam time constraints

- **Command Accuracy**: Focus on exact syntax and options
- **Verification**: Always verify your configurations work
- **Documentation**: Use `man pages` and `--help` during practice
- **Persistence**: Ensure configurations survive reboots

1.6 Repository Structure

```
rhcsa/
├── docs/                    # Study materials (this site)
├── anki/                    # Flashcard deck
├── vagrant/                 # Lab environment
├── sources/                 # External reference materials
└── README.md               # Quick start guide
```

For the complete repository structure, visit: codeberg.org/kraker/rhcsa

Ready to start your RHCSA journey? Begin with Module 00: Exam Overview to understand the exam format and strategy!

2. Study Modules

2.1 RHCSA Synthesis - Unified Knowledge Base

A comprehensive synthesis combining the methodologies of Asghar Ghori and Sander van Vugt's RHCSA study guides

Overview

This knowledge base synthesizes content from two authoritative RHCSA study resources:

- **Asghar Ghori**: "RHCSA Red Hat Enterprise Linux 10" (Dec 2025 edition)
- **Sander van Vugt**: "Red Hat RHCSA 9 Cert Guide" (concepts still applicable)

Each topic module combines the best approaches from both authors, providing comprehensive coverage with practical labs, detailed explanations, and exam-focused strategies.

Study Approach

For First-Time Learners

1. Start with Exam Overview for context
2. Follow modules in numerical order (01-15)
3. Complete all hands-on labs in each module
4. Use Quick Reference Cards for review

For Exam Preparation

1. Review Exam Overview for strategy
2. Focus on modules matching your weak areas
3. Practice all troubleshooting scenarios
4. Use Knowledge Checks for self-assessment

For Reference/Review

1. Use this index to jump to specific topics
2. Leverage Quick Reference Cards for rapid lookup
3. Consult Troubleshooting Playbooks for specific issues

Module Index

Foundation Topics

Module	Topic	Focus Areas	Exam Weight
00	**Exam Overview**	Format, strategy, environment setup	Essential
01	**System Installation**	RHEL installation, initial configuration	High
02	**File Management**	Basic operations, text processing	High
03	**User & Group Management**	Account creation, policies, sudo	Critical

System Administration

Module	Topic	Focus Areas	Exam Weight
04	**File Permissions**	File permissions, access controls	Critical
05	**Process & Service Management**	Systemd, process control	Critical
06	**Package Management**	RPM, YUM, DNF, repositories	High
07	**Storage & LVM**	Partitions, filesystems, LVM	Critical
08	**Network Configuration**	IP configuration, DNS, routing	High

Security and Advanced Topics

Module	Topic	Focus Areas	Exam Weight
09	**SELinux Management**	Contexts, booleans, troubleshooting	Critical
10	**Firewall Configuration**	firewall-cmd, zones, services	High
11	**Boot Process & GRUB**	Boot sequence, GRUB configuration	Medium
12	**Logging & Monitoring**	rsyslog, journald, log analysis	Medium
13	**Scheduled Tasks**	cron, at, systemd timers	Medium

Modern RHEL Features

Module	Topic	Focus Areas	Exam Weight
14	**Flatpak Management**	Flatpak repos, application management	High
15	**Troubleshooting**	System recovery, boot issues	Medium

Quick Navigation

By Exam Objective

- **Understand and use essential tools** → Modules 01, 02
- **Create simple shell scripts** → Module 02, 13
- **Operate running systems** → Modules 05, 06, 12, 13
- **Configure local storage** → Module 07
- **Create and configure file systems** → Modules 04, 07
- **Deploy, configure, and maintain systems** → Modules 01, 08, 09, 14, 15
- **Manage basic networking** → Module 08

- **Manage users and groups** → Module 03
- **Manage security** → Modules 04, 09, 10

By Common Tasks

- **System Setup**: Modules 01, 03, 08
- **Storage Configuration**: Modules 07
- **Security Hardening**: Modules 04, 09, 10
- **Service Management**: Modules 05, 13, 14, 15
- **Troubleshooting**: All modules (dedicated sections)

Study Progress Tracking

Track your progress through the synthesis modules:

- [ ] 00 - Exam Overview
- [ ] 01 - System Installation
- [ ] 02 - File Management
- [ ] 03 - User & Group Management
- [ ] 04 - File Permissions
- [ ] 05 - Process & Service Management
- [ ] 06 - Package Management
- [ ] 07 - Storage & LVM
- [ ] 08 - Network Configuration
- [ ] 09 - SELinux Management
- [ ] 10 - Firewall Configuration
- [ ] 11 - Boot Process & GRUB
- [ ] 12 - Logging & Monitoring
- [ ] 13 - Scheduled Tasks
- [ ] 14 - Flatpak Management
- [ ] 15 - Troubleshooting

Additional Resources

Original Sources

- Current ebook analysis: ebook_summary.md
- Comprehensive flashcards: rhcsa_deck.csv
- Quick exam reference: exam_quick_reference.md

Command References

- Organized by topic: command_reference_by_topic.md
- Acronyms and terminology: rhcsa_acronyms_glossary.md

Study Tip: Each module is self-contained but cross-references related topics. Don't feel obligated to read linearly - jump to what you need to learn or review!

2.2 00 - RHCSA Exam Overview & Strategy

Navigation: Index | Next → System Installation

1. Executive Summary

Topic Scope: Understanding the RHCSA exam format, environment, and strategic approach to maximize success

RHCSA Relevance: Essential foundation for all other topics - understanding the exam is critical for effective preparation

Exam Weight: Essential - This knowledge affects performance on every exam task

Prerequisites: None - start here for complete exam preparation

Related Topics: All modules benefit from this overview

2. RHCSA Exam Fundamentals

Exam Format

- **Type**: Performance-based, hands-on exam (no multiple choice)
- **Duration**: 3 hours
- **Environment**: Virtual machines running RHEL 10
- **Tasks**: Practical tasks to complete
- **Passing Score**: Typically 210/300 points (70%)
- **Delivery**: Red Hat Training Centers or remote proctoring

Exam Environment

- **Systems**: RHEL 10 virtual machines
- **Access**: SSH and console access to systems
- **Tools**: Standard RHEL 10 command line tools and documentation
- **Network**: Limited internet access (man pages available)
- **Time Pressure**: Approximately 9-12 minutes per task average

Key Exam Objectives (Red Hat Official)

1. **Understand and use essential tools**
2. **Create simple shell scripts**
3. **Operate running systems**
4. **Configure local storage**
5. **Create and configure file systems**
6. **Deploy, configure, and maintain systems**
7. **Manage basic networking**
8. **Manage users and groups**
9. **Manage security**

---

3. Strategic Approach

Pre-Exam Preparation

Mental Preparation

- **Sleep well**: 7-8 hours before exam day
- **Eat properly**: Light meal 2 hours before exam
- **Arrive early**: 15-30 minutes before appointment
- **Review briefly**: Quick scan of command reference only

Knowledge Verification

- Complete all synthesis modules at least twice
- Practice all hands-on labs until automatic
- Memorize critical command syntax
- Understand troubleshooting workflows

During the Exam

Initial Setup (First 10 minutes)

1. **Read all tasks quickly** - get overview of what's needed
2. **Check system hostnames** - understand the environment
3. **Test connectivity** - ensure SSH works between systems
4. **Note dependencies** - identify tasks that depend on others
5. **Plan your sequence** - tackle easier tasks first for confidence

Task Execution Strategy

1. **Time boxing**: Allocate maximum time per task (don't get stuck)
2. **Verify immediately**: Test each task completion before moving on
3. **Skip and return**: If stuck, mark task and continue
4. **Document issues**: Quick notes on partial completions
5. **Save regularly**: Some tasks auto-save, others need manual saves

Time Management

- **First hour**: Complete 6-8 easier tasks
- **Second hour**: Tackle 4-5 medium difficulty tasks
- **Final hour**: Address remaining difficult tasks and review
- **Last 15 minutes**: Final verification of all completed tasks

---

4. Exam Environment Deep Dive

System Layout (Typical)

```
Exam Environment:
├── workstation.lab.example.com (your main system)
├── server1.lab.example.com    (target system 1)
└── server2.lab.example.com    (target system 2)
```

Common Hostnames and IPs

- **workstation**: 192.168.1.10 (your working system)
- **server1**: 192.168.1.11 (primary target)
- **server2**: 192.168.1.12 (secondary target)

Available Tools

```
# Text editors
vim, nano

# File managers
mc (Midnight Commander - if available)

# Network tools
ssh, scp, rsync

# Documentation
man pages, info pages
/usr/share/doc/ (system documentation)

# System tools
All standard RHEL 10 command-line utilities
```

What's NOT Available

- GUI applications (exam is command-line only)
- Internet browsers
- External documentation sites
- Personal notes or cheat sheets

---

5. Task Categories and Strategies

Category 1: System Configuration (30-35% of exam)

Typical Tasks:

- Configure network settings
- Set up user accounts and groups
- Configure SSH access
- Set hostname and timezone

Strategy:

- These are usually straightforward
- Complete early for confidence
- Double-check with verification commands
- Common commands: `nmtui`, `useradd`, `systemctl`

Category 2: Storage Management (25-30% of exam)

Typical Tasks:

- Create partitions and filesystems
- Configure LVM
- Mount filesystems persistently
- Set up swap space

Strategy:

- Be very careful with disk operations
- Always verify before writing changes
- Test mounts with `mount -a`
- Common commands: `fdisk`, `mkfs`, `pvcreate`, `mount`

Category 3: Security Configuration (20-25% of exam)

Typical Tasks:

- Configure firewall rules
- Set up SELinux contexts
- Configure file permissions
- Set up sudo access

Strategy:

- Security tasks often have dependencies
- Test access from different accounts
- Verify with appropriate tools
- Common commands: `firewall-cmd`, `restorecon`, `chmod`, `visudo`

Category 4: Service Management (15-20% of exam)

Typical Tasks:

- Configure and start services
- Set up scheduled tasks
- Configure logging
- Manage Flatpak software

Strategy:

- Services must be enabled AND started
- Test functionality after configuration
- Check logs for errors
- Common commands: `systemctl`, `crontab`, `flatpak`

---

6. Common Exam Mistakes to Avoid

Critical Mistakes (Task Failure)

1. **Wrong system**: Performing task on wrong host
2. **Permissions errors**: Forgetting to use sudo when needed
3. **Service not enabled**: Starting but not enabling services
4. **Persistent configuration**: Making temporary changes only
5. **Wrong user context**: Performing tasks as wrong user

Time-Wasting Mistakes

1. **Perfectionism**: Over-configuring beyond requirements
2. **Rabbit holes**: Spending too long troubleshooting one task
3. **Verification obsession**: Testing same thing multiple times
4. **Manual pages deep dive**: Reading entire man pages
5. **Second-guessing**: Changing correct configurations

Recovery Strategies

1. **Task failure**: Move on, return later if time permits
2. **System broken**: Use rescue mode or reinstall if necessary
3. **Partial credit**: Document what you accomplished

4. **Time pressure**: Focus on verification of completed tasks
5. **Configuration errors**: Revert to working state and retry

---

7. Verification Techniques

Standard Verification Workflow

```
# For each completed task:
1. Test the functionality as requested
2. Verify persistence (reboot if necessary)
3. Check from the perspective described in task
4. Document success in your notes
```

Service Configuration Verification

```
# Always verify services are:
systemctl status service-name      # Running
systemctl is-enabled service-name  # Enabled for boot
# Test actual functionality
```

Storage Configuration Verification

```
# Always verify storage is:
lsblk                    # Properly configured
mount | grep mountpoint  # Currently mounted
cat /etc/fstab          # Persistent configuration
```

Network Configuration Verification

```
# Always verify network is:
ip addr show            # Interface has correct IP
ping target-host        # Connectivity works
ss -tuln               # Services listening on correct ports
```

User/Security Verification

```
# Always verify access is:
su - username          # User can log in
sudo -l -U username    # User has correct sudo rights
ssh username@host      # Remote access works
```

---

8. Lab Environment Setup for Practice

Recommended Practice Environment

```
# Minimum setup for RHCSA practice:
- 2 RHEL 10 VMs (4GB RAM each, 20GB+ disk)
- Network connectivity between systems
- Additional storage devices for LVM practice
- SSH configured between systems
```

Using Vagrant for Practice (from this repository)

```
# Navigate to vagrant directory
cd /path/to/rhcsa/vagrant

# Source credentials and start VMs
source .rhel-credentials && vagrant up

# This provides:
# - rhel10a: Primary practice system
# - rhel10b: Secondary system with extra storage
# - Automatic Red Hat registration
# - Network connectivity configured
```

Practice Scenarios

1. **Daily practice**: 30-45 minutes on 2-3 tasks
2. **Weekend labs**: 2-3 hour sessions simulating exam conditions
3. **Mock exams**: Full 3-hour timed practice with all task types
4. **Weak area focus**: Dedicated sessions on challenging topics

---

9. Final Preparation Checklist

One Week Before Exam

- [ ] Complete all synthesis modules
- [ ] Practice all hands-on labs
- [ ] Take at least 2 full mock exams
- [ ] Review troubleshooting playbooks
- [ ] Verify exam logistics (location, time, requirements)

Day Before Exam

- [ ] Light review of command syntax only
- [ ] Prepare required identification
- [ ] Confirm exam location and arrival time
- [ ] Get good night's sleep (7-8 hours)
- [ ] Avoid intensive studying

Morning of Exam

- [ ] Light breakfast 2 hours before
- [ ] Review quick reference cards only (15 minutes max)
- [ ] Arrive 15-30 minutes early
- [ ] Bring required identification
- [ ] Relax and trust your preparation

---

10. Exam Day Tactics

When You Sit Down

1. **Take deep breaths** - calm your nerves
2. **Read instructions carefully** - understand exam rules
3. **Scan all tasks** - get the big picture
4. **Identify easy wins** - build confidence first
5. **Note time limits** - plan your approach

During Task Execution

1. **Read twice, execute once** - understand requirements fully
2. **Work systematically** - complete one task fully before starting next
3. **Verify immediately** - test each task before moving on
4. **Stay calm** - if something fails, try a different approach
5. **Use time wisely** - don't spend too long on any single task

Final Review Phase

1. **Test critical functionality** - make sure key services work
2. **Check persistent configuration** - verify settings survive reboot
3. **Review partial completions** - ensure maximum partial credit

4. **Don't overthink** - resist urge to change working configurations
5. **Submit confidently** - trust your preparation and execution

---

Summary

Key Takeaways

- **RHCSA is performance-based** - hands-on skills matter more than theory
- **Time management is critical** - don't get stuck on any single task
- **Verification is essential** - always test your work immediately
- **Practice under pressure** - simulate exam conditions during preparation

Success Factors

1. **Thorough preparation** through all synthesis modules
2. **Practical experience** with hands-on labs
3. **Strategic thinking** during the exam
4. **Calm execution** under time pressure
5. **Systematic verification** of all work

Next Steps

- Begin with Module 01: System Installation
- Set up your practice environment using Vagrant
- Start with easier modules and progress to advanced topics

Navigation: Index | Next → System Installation

2.3 01 - System Installation & Initial Configuration

Navigation: ← Exam Overview | Index | Next → File Management

1. Executive Summary

Topic Scope: RHEL 10 installation process, initial system configuration, and post-installation setup

RHCSA Relevance: Foundation knowledge - while not directly tested, understanding installation helps with system administration tasks

Exam Weight: Medium - Installation concepts appear in troubleshooting and system configuration scenarios

Prerequisites: Basic understanding of Linux concepts and virtualization

Related Topics: Boot Process & GRUB, Storage & LVM, Network Configuration

2. Conceptual Foundation

Core Theory

RHEL installation involves deploying the Red Hat Enterprise Linux operating system using the **Anaconda** installer. The process includes:

- **System preparation**: Hardware verification and boot media creation
- **Installation configuration**: Language, storage, network, and user setup
- **Package selection**: Choosing software packages based on intended use
- **Post-installation**: Initial login and system verification

Real-World Applications

- **Data center deployments**: Automated installation using Kickstart files
- **Development environments**: Virtual machine installations for testing
- **Production servers**: Careful configuration for specific workloads
- **Lab environments**: Practice installations for certification preparation

Common Misconceptions

- **Installation = configuration**: Installation only provides the base system
- **Default settings are optimal**: Production systems require careful customization
- **GUI required**: RHEL can be fully managed from command line
- **Single partition layout**: Multiple partitions provide better organization and security

Key Terminology

- **Anaconda**: The RHEL installer program
- **ISO image**: Bootable installation media file
- **Kickstart**: Automated installation configuration file
- **Base environment**: Predefined software package collections
- **Root filesystem**: Primary filesystem containing the operating system
- **Boot partition**: Separate partition containing boot loader and kernel files

---

3. Command Mastery

Pre-Installation Commands

```
# Verify system requirements
lscpu                    # Check CPU information
free -h                  # Check memory availability
lsblk                    # List available storage devices
ip addr show             # Check network interfaces

# ISO verification (if needed)
sha256sum rhel-9.1-x86_64-dvd.iso
```

Post-Installation Verification

```
# System information
hostnamectl              # Display system hostname and info
uname -a                 # Kernel and system information
cat /etc/os-release      # Operating system version details

# Storage verification
lsblk                    # List all block devices
df -h                    # Check filesystem usage
mount | column -t        # Display mounted filesystems

# Network verification
ip addr show             # Display IP configuration
ping -c 3 8.8.8.8       # Test network connectivity
```

Initial System Configuration

```
# Set system hostname
hostnamectl set-hostname server1.example.com

# Configure timezone
timedatectl set-timezone America/New_York
timedatectl status

# Update system (post-installation)
dnf update -y

# Check enabled services
systemctl list-unit-files --type=service --state=enabled
```

---

4. Installation Workflows

Standard Installation Procedure

1. **Boot from Installation Media**

   • Select "Install Red Hat Enterprise Linux 9.x"

   • Wait for Anaconda to load (may take several minutes)

2. **Language and Localization**

   • Select installation language

   • Configure keyboard layout

   • Set date and time/timezone

3. **Installation Source**

   - Verify installation media is detected
   - Configure additional repositories if needed

4. **Software Selection**

```
Available Base Environments:
├── Server (recommended for RHCSA)
├── Minimal Install (command line only)
├── Workstation (desktop environment)
├── Custom Operating System (advanced users)
└── Virtualization Host (for hypervisors)
```

5. **Storage Configuration**

   - **Automatic partitioning**: Simple, good for learning
   - **Custom partitioning**: More control, better for production

6. **Network Configuration**

   - Configure hostname
   - Set up network interfaces
   - Configure static IP if needed

7. **User Configuration**

   - Set root password (required)
   - Create regular user account (recommended)
   - Configure sudo access

8. **Begin Installation**

   - Review settings summary
   - Start installation process
   - Configure users while installation proceeds

Recommended Partitioning Scheme

```
# For RHCSA practice (20GB disk):
/boot     1GB   (ext4)    # Boot files and kernels
/         18GB  (xfs)     # Root filesystem
swap      1GB   (swap)    # Virtual memory

# For production environments:
/boot     1GB   (ext4)    # Boot files
/         10GB  (xfs)     # Root filesystem
/home     5GB   (xfs)     # User data
/var      3GB   (xfs)     # Variable data (logs, etc.)
swap      1GB   (swap)    # Virtual memory
```

Base Environment Comparison

Environment	Size	GUI	Services	Use Case
Server	~3GB	No	Standard server services	RHCSA practice, production servers
Minimal	~1GB	No	Essential only	Containers, embedded systems
Workstation	~5GB	Yes	Desktop services	Development, desktop use

---

5. Configuration Deep Dive

Anaconda Installation Configuration

During installation, Anaconda creates several key configuration files:

Network Configuration

```
# /etc/hostname
server1.example.com
```

User Configuration

```
# /etc/passwd (user entries created)
root:x:0:0:root:/root:/bin/bash
user1:x:1000:1000:User One:/home/user1:/bin/bash
```

Filesystem Configuration

```
# /etc/fstab (automatically generated)
/dev/mapper/rhel-root   /       xfs     defaults        0 0
UUID=abc123-def456      /boot   ext4    defaults        1 2
/dev/mapper/rhel-swap   swap    swap    defaults        0 0
```

Post-Installation Configuration Files

System Information

```
# /etc/os-release
NAME="Red Hat Enterprise Linux"
VERSION="9.1 (Plow)"
ID="rhel"
VERSION_ID="9.1"
PLATFORM_ID="platform:el9"
```

Installed Package Information

```
# View installation log
cat /var/log/anaconda/anaconda.log

# List packages installed during installation
dnf history info 1
```

---

6. Hands-On Labs

Lab 6.1: Basic RHEL Installation (Asghar Ghori Method)

Objective: Install RHEL 10 with standard configuration for RHCSA practice

Prerequisites:

- RHEL 10 ISO image
- Virtual machine with 20GB disk, 2GB RAM
- Network connectivity

Steps:

1. **Create Virtual Machine**

```
# In VirtualBox/VMware:
# - Name: rhel10-server1
# - RAM: 2048MB
# - Disk: 20GB dynamically allocated
# - Network: NAT or Bridged
```

2. **Boot Installation Media**

   • Attach RHEL 10 ISO to VM
   • Boot from ISO
   • Select "Install Red Hat Enterprise Linux 9.x"

3. **Configure Installation**

   • Language: English (US)
   • Software Selection: Server
   • Installation Destination: Use entire disk, automatic partitioning

4. **Network Configuration**

   • Set hostname: `server1.example.com`
   • Configure network interface with DHCP or static IP

5. **User Configuration**

   • Root password: Set secure password
   • Create user: Regular user with sudo privileges

6. **Complete Installation**

   • Review summary and begin installation
   • Wait for completion (20-30 minutes)
   • Reboot system

Verification:

```
# After reboot, verify installation
hostnamectl                    # Check hostname
cat /etc/os-release           # Verify RHEL version
lsblk                         # Check disk partitioning
ip addr show                  # Verify network configuration
systemctl status              # Check system status
```

Lab 6.2: Custom Partitioning Installation (Sander van Vugt Method)

Objective: Install RHEL 10 with custom partitioning scheme

Steps:

1. **Follow initial steps from Lab 6.1** through software selection
2. **Custom Storage Configuration**

   • Installation Destination → Custom → Done

   • Create new mount points:

```
/boot    1GB    ext4
/        10GB   xfs
/home    5GB    xfs
/var     3GB    xfs
swap     1GB    swap
```

3. **Configure each partition**:

```
# For each mount point:
# - Click "+" to add mount point
# - Specify mount point and size
# - Select filesystem type
# - Click "Add mount point"
```

4. **Complete installation** following remaining steps from Lab 6.1

Verification:

```
# Verify custom partitioning
lsblk                          # Check partition layout
df -h                          # Check filesystem usage
cat /etc/fstab                # Verify fstab entries
mount | grep "^/" | sort     # List mounted filesystems
```

Lab 6.3: Post-Installation Configuration

Objective: Configure newly installed system for RHCSA practice

Steps:

1. **System Updates**

```
# Register system (if using RHEL subscription)
subscription-manager register --username your_username

# Update all packages
dnf update -y
```

2. **Additional Software Installation**

```
# Install useful tools for RHCSA practice
dnf groupinstall "Development Tools" -y
dnf install vim wget curl man-pages -y
```

3. **Security Configuration**

```
# Configure firewall
firewall-cmd --state
firewall-cmd --list-all

# Enable SELinux (verify)
getenforce
```

4. **User Environment**

```
# Configure bash aliases for root
echo 'alias ll="ls -la"' >> /root/.bashrc
echo 'alias grep="grep --color=auto"' >> /root/.bashrc
```

Verification:

```
# Verify post-installation configuration
dnf list installed | wc -l   # Count installed packages
systemctl list-unit-files --state=enabled | wc -l  # Count enabled services
firewall-cmd --list-all       # Check firewall status
getenforce                    # Verify SELinux status
```

---

7. Troubleshooting Playbook

Common Installation Issues

Issue 1: Installation Media Not Detected

Symptoms:

- Boot process hangs or shows errors
- "No installation source found" message

Diagnosis:

```
# Check ISO integrity before installation
sha256sum /path/to/rhel-9.x-x86_64-dvd.iso
# Compare with official checksum from Red Hat
```

Resolution:

- Re-download ISO image if corrupted
- Verify virtual machine CD/DVD settings
- Try different boot order in BIOS/UEFI

Prevention: Always verify ISO checksums before installation

Issue 2: Insufficient Disk Space

Symptoms:

- "Not enough space" error during partitioning
- Installation fails during package installation

Diagnosis:

```
# In installer, check available disk space
# Minimum requirements:
# - Server: 3GB
# - Workstation: 5GB
# - Recommended: 20GB+ for practice
```

Resolution:

- Increase virtual machine disk size
- Choose Minimal Install if space limited
- Use custom partitioning to optimize space usage

Issue 3: Network Configuration Problems

Symptoms:

- Cannot set hostname
- Network interface not detected
- No network connectivity post-installation

Diagnosis:

```
# During installation, check network tab
# Post-installation:
ip link show                  # Check if interfaces exist
ip addr show                  # Check IP configuration
```

Resolution:

```
# Post-installation network fix:
nmcli connection show
nmcli connection up "interface-name"
systemctl restart NetworkManager
```

Boot Issues After Installation

Issue 4: System Won't Boot

Symptoms:

- GRUB rescue prompt
- Kernel panic messages
- Black screen after boot

Diagnosis:

```
# From rescue media:
mkdir /mnt/sysimage
mount /dev/mapper/rhel-root /mnt/sysimage
chroot /mnt/sysimage
```

Resolution:

```
# Reinstall GRUB bootloader
grub2-install /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg
```

---

8. Quick Reference Card

Essential Installation Commands

```
# Pre-installation verification
lscpu                    # Check CPU
free -h                  # Check memory
lsblk                    # Check storage

# Post-installation verification
hostnamectl              # System info
uname -a                 # Kernel info
df -h                    # Storage usage
ip addr show             # Network config
```

Key File Locations

- **Installation logs**: `/var/log/anaconda/`
- **System configuration**: `/etc/os-release`

- **Filesystem mounts**: `/etc/fstab`
- **Network configuration**: `/etc/NetworkManager/`

Installation Options

- **Graphical**: Default installation interface
- **Text mode**: Add `inst.text` to boot parameters
- **VNC**: Add `inst.vnc` for remote installation
- **Kickstart**: Add `inst.ks=URL` for automated installation

Verification Commands

```
# Quick system health check
systemctl status         # System status
journalctl -b           # Boot messages
dmesg | tail           # Kernel messages
```

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What is the name of the RHEL installer program? **Answer**: Anaconda - this is the graphical and text-based installer used for all RHEL installations.
2. **Question**: What are the minimum partition requirements for RHEL installation? **Answer**: Root filesystem (/) and swap partition. While /boot is recommended as separate partition, it can reside within the root filesystem in simple installations.
3. **Question**: What is the difference between Server and Minimal Install base environments? **Answer**: Server includes standard server services and networking tools (~3GB), while Minimal Install contains only essential packages for basic system operation (~1GB).

Practical Scenarios

1. **Scenario**: You need to install RHEL on a system with only 10GB available disk space. **Solution**: Use Minimal Install base environment, create 8GB root partition and 2GB swap, or use custom partitioning to optimize space allocation.
2. **Scenario**: Installation completed but system won't boot, showing GRUB rescue prompt. **Solution**: Boot from installation media in rescue mode, chroot to installed system, reinstall GRUB bootloader using grub2-install and grub2-mkconfig commands.

Command Challenges

1. **Challenge**: Write commands to verify a successful RHEL installation. **Answer**:

```
hostnamectl              # Check system info
cat /etc/os-release     # Verify RHEL version
lsblk && df -h          # Check storage
ip addr show            # Verify network
systemctl status        # Check system health
```

---

10. Exam Strategy

Topic-Specific Tips

- Installation knowledge helps with boot troubleshooting tasks
- Understand default partitioning schemes for storage questions
- Know post-installation configuration locations
- Practice both graphical and text-mode installations

Common Exam Scenarios

1. **Scenario**: Fix boot issues on system that won't start **Approach**: Use rescue mode, check /boot contents, verify GRUB configuration
2. **Scenario**: Configure hostname during system setup **Approach**: Use `hostnamectl set-hostname` command, verify with `hostnamectl status`

Time Management

- **Installation tasks**: Usually 5-10 minutes for configuration
- **Boot troubleshooting**: Allocate 15-20 minutes maximum
- **Quick verification**: Use fast commands like `hostnamectl`, `lsblk`

Pitfalls to Avoid

- Don't spend excessive time on installation details during exam
- Remember to make configuration changes persistent
- Always verify system boots correctly after changes
- Check both current state and persistent configuration

Summary

Key Takeaways

- **Anaconda** is the RHEL installer program with graphical and text interfaces
- **Server base environment** is ideal for RHCSA practice and exam preparation
- **Standard partitioning** includes root (/) and swap at minimum, /boot recommended
- **Post-installation verification** ensures system is properly configured

Critical Commands to Remember

```
hostnamectl                # System information and hostname management
lsblk                      # Display block devices and partitions
systemctl status           # Check system and service status
```

Next Steps

- Continue to Module 02: File Management
- Practice installation in virtual environment using Vagrant
- Review related topics: Boot Process, Storage

Navigation: ← Exam Overview | Index | Next → File Management

2.4 02 - File Management & Text Processing

Navigation: ← System Installation | Index | Next → User Management

1. Executive Summary

Topic Scope: Essential file operations, text processing, archiving, and linking in RHEL 10

RHCSA Relevance: Critical foundation skill - file management appears in virtually every exam task

Exam Weight: High - File operations are fundamental to all system administration tasks

Prerequisites: Basic Linux command line familiarity

Related Topics: File Permissions, Storage & LVM, Troubleshooting

2. Conceptual Foundation

Core Theory

File management in Linux operates on the principle that "everything is a file." This includes:

- **Regular files**: Documents, scripts, configuration files
- **Directories**: Containers for organizing files and other directories
- **Links**: References to files that can be hard (same inode) or symbolic (pathname reference)
- **Special files**: Device files, pipes, sockets

Real-World Applications

- **Configuration management**: Editing system configuration files
- **Log analysis**: Processing and searching through system logs
- **Backup operations**: Creating and extracting archives
- **Documentation**: Creating and maintaining system documentation
- **Automation**: Writing and managing shell scripts

Common Misconceptions

- **Case sensitivity**: Linux is case-sensitive (`File.txt` ≠ `file.txt`)

- **File extensions**: Extensions are for humans; Linux determines file type by content
- **Hidden files**: Files starting with `.` are hidden from `ls` by default
- **Directory permissions**: Different from file permissions; affect access to directory contents

Key Terminology

- **Inode**: Index node containing file metadata and disk block locations
- **Hard link**: Multiple directory entries pointing to the same inode
- **Symbolic link**: File containing the pathname of another file
- **Archive**: Single file containing multiple files and directories
- **Compression**: Reducing file size using algorithms like gzip or bzip2
- **Glob patterns**: Wildcards for matching multiple files (`*`, `?`, `[]`)

---

3. Command Mastery

Essential File Operations

```
# Listing files and directories
ls -la                  # Long format with hidden files
ls -lh                  # Human-readable sizes
ls -lt                  # Sort by modification time
ls -lS                  # Sort by size
ls -R                   # Recursive listing

# Creating and removing
touch filename          # Create empty file or update timestamp
mkdir -p path/to/dir    # Create directory path recursively
rmdir directory         # Remove empty directory
rm -rf directory        # Remove directory and contents recursively

# Copying and moving
cp source destination   # Copy file
cp -r source dest       # Copy directory recursively
cp -p source dest       # Preserve permissions and timestamps
mv source destination   # Move/rename file or directory
```

Text Processing Commands

```
# Viewing file contents
cat filename             # Display entire file
less filename            # Page through file content
head -n 10 file          # First 10 lines
tail -n 10 file          # Last 10 lines
tail -f file             # Follow file changes (logs)

# Text manipulation
grep pattern file        # Search for pattern in file
grep -i pattern file     # Case-insensitive search
grep -r pattern dir      # Recursive search in directory
grep -v pattern file     # Invert match (exclude pattern)

# Text processing tools
sort file                # Sort lines alphabetically
sort -n file             # Sort numerically
uniq file                # Remove duplicate lines
wc -l file               # Count lines
cut -d: -f1 /etc/passwd  # Extract first field (delimiter :)
```

Archiving and Compression

```
# Tar archives
tar -czf archive.tar.gz directory/    # Create compressed archive
tar -xzf archive.tar.gz               # Extract compressed archive
tar -tzf archive.tar.gz               # List archive contents
tar -xzf archive.tar.gz file.txt      # Extract specific file

# Alternative compression
gzip file                # Compress file (creates file.gz)
gunzip file.gz           # Decompress file
zip archive.zip files    # Create zip archive
unzip archive.zip        # Extract zip archive
```

File Linking

```
# Hard links
ln source hardlink       # Create hard link
ls -li file*             # Show inode numbers

# Symbolic links
ln -s target symlink     # Create symbolic link
ls -l symlink            # Show link target
readlink symlink         # Display link target
```

Command Reference Table

Command	Purpose	Key Options	Example
`ls`	List directory contents	`-l`, `-a`, `-h`, `-t`	`ls -lah /home`
`find`	Search for files	`-name`, `-type`, `-size`	`find / -name "*.conf"`
`locate`	Quick file search	`-i`, `-c`	`locate passwd`
`grep`	Search text patterns	`-i`, `-r`, `-v`	`grep -r "error" /var/log`
`tar`	Archive files	`-c`, `-x`, `-z`, `-f`	`tar -czf backup.tar.gz /home`

---

4. Procedural Workflows

Standard Procedure: File Search and Analysis

1. **Initial search**: Use `locate` for quick filename searches

```
locate filename
updatedb  # Update locate database if needed
```

2. **Detailed search**: Use `find` for complex criteria

```
find /path -name "pattern" -type f -size +1M
```

3. **Content analysis**: Examine file contents

```
file filename                # Determine file type
less filename               # Review content
grep "pattern" filename     # Search within file
```

4. **Verification**: Confirm file properties

```
ls -l filename             # Check permissions and size
stat filename              # Detailed file information
```

Decision Tree: Archive Strategy

```
Archive Task
├── Backup entire directory? → tar -czf backup.tar.gz directory/
├── Selective file backup? → tar -czf backup.tar.gz file1 file2 file3
├── Cross-platform compatibility? → zip -r archive.zip directory/
└── Maximum compression? → tar -cjf backup.tar.bz2 directory/
```

Standard Procedure: Log Analysis Workflow

1. **Identify log location**: Common locations

```
ls /var/log/                # System logs
journalctl --list-boots     # Systemd journal
```

2. **Filter relevant entries**: Use grep patterns

```
grep -i "error\|fail\|warn" /var/log/messages
tail -f /var/log/secure     # Monitor authentication
```

3. **Time-based analysis**: Focus on specific periods

```
grep "$(date '+%b %d')" /var/log/messages  # Today's entries
journalctl --since "1 hour ago"           # Recent systemd logs
```

---

5. Configuration Deep Dive

File System Navigation

- `/etc/`: System configuration files
- `/var/log/`: System and application logs
- `/tmp/`: Temporary files (cleared on reboot)
- `/home/`: User home directories
- `/opt/`: Optional software packages

Glob Pattern Usage

```
# Wildcards
ls *.txt                # All .txt files
ls file?.log            # file1.log, file2.log, etc.
ls file[1-3].txt        # file1.txt, file2.txt, file3.txt
ls file[!1].txt         # All except file1.txt

# Complex patterns
find . -name "*.conf" -o -name "*.cfg"     # Multiple patterns
ls {*.txt,*.log}                           # Brace expansion
```

Archive Best Practices

```
# Include/exclude patterns
tar --exclude="*.tmp" -czf backup.tar.gz directory/
tar --exclude-from=exclude-list.txt -czf backup.tar.gz /

# Verification
tar -tzf archive.tar.gz | head -10        # List first 10 files
tar -df archive.tar.gz                    # Compare with filesystem
```

---

6. Hands-On Labs

Lab 6.1: File Operations Mastery (Asghar Ghori Style)

Objective: Master essential file operations and text processing

Steps:

1. **Create test environment**

```
mkdir -p ~/lab02/{documents,logs,archives}
cd ~/lab02
```

2. **Generate test files**

```
echo "System log entry 1" > logs/system.log
echo "Error message here" >> logs/system.log
echo "Normal operation" >> logs/system.log
echo "Configuration data" > documents/config.txt
```

3. **Practice file operations**

```
# Copy with different options
cp documents/config.txt documents/config.backup
cp -p logs/system.log logs/system.$(date +%Y%m%d)

# Search operations
find . -name "*.log" -type f
grep -r "Error" .
```

Verification:

```
ls -la logs/                    # Verify file creation
find . -name "*.backup"         # Check backup files
wc -l logs/system.log          # Count log entries
```

Lab 6.2: Advanced Text Processing (Sander van Vugt Style)

Objective: Master text processing and analysis techniques

Steps:

1. **Create sample data**

```
cp /etc/passwd ~/lab02/passwd.sample
cp /var/log/messages ~/lab02/messages.sample 2>/dev/null || \
journalctl > ~/lab02/messages.sample
```

2. **Text analysis tasks**

```
# User analysis
cut -d: -f1,3 ~/lab02/passwd.sample | sort -t: -k2 -n
grep -c "bash\|sh" ~/lab02/passwd.sample

# Log analysis
grep -i "error\|fail" ~/lab02/messages.sample | wc -l
tail -20 ~/lab02/messages.sample | grep -v "systemd"
```

3. **Advanced filtering**

```
# Complex grep patterns
grep -E "(error|fail|warn)" ~/lab02/messages.sample
awk '{print $1, $2, $3}' ~/lab02/messages.sample | head -10
```

Verification:

```
# Verify text processing results
cut -d: -f1 ~/lab02/passwd.sample | sort | head -5
grep -c ":" ~/lab02/passwd.sample
```

Lab 6.3: Archive and Link Management (Synthesis Challenge)

Objective: Master archiving, compression, and linking

Scenario: Create a backup system with different archive types and linking strategies

Requirements:

- Create compressed archives of different directories
- Implement hard and symbolic links
- Practice archive extraction and verification

Solution Steps:

1. **Prepare directory structure**

```
mkdir -p ~/lab02/backup-test/{dir1,dir2,dir3}
echo "File in dir1" > ~/lab02/backup-test/dir1/file1.txt
echo "File in dir2" > ~/lab02/backup-test/dir2/file2.txt
echo "Shared content" > ~/lab02/backup-test/shared.txt
```

2. **Create various archives**

```
# Different compression methods
tar -czf ~/lab02/archives/backup-gzip.tar.gz ~/lab02/backup-test/
tar -cjf ~/lab02/archives/backup-bzip2.tar.bz2 ~/lab02/backup-test/
zip -r ~/lab02/archives/backup.zip ~/lab02/backup-test/
```

3. **Implement linking strategy**

```
# Hard links
ln ~/lab02/backup-test/shared.txt ~/lab02/backup-test/dir1/shared-hard

# Symbolic links
ln -s ../shared.txt ~/lab02/backup-test/dir2/shared-soft
```

4. **Verification and analysis**

```
# Compare archive sizes
ls -lh ~/lab02/archives/

# Verify links
ls -li ~/lab02/backup-test/shared.txt ~/lab02/backup-test/dir1/shared-hard
ls -l ~/lab02/backup-test/dir2/shared-soft
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: "No such file or directory" errors

Symptoms:

- Commands fail with file not found errors
- Scripts cannot locate files

Diagnosis:

```
# Check current directory
pwd
# Verify file existence
ls -la filename
# Check path components
ls -ld /path/to/file
```

Resolution:

```
# Use absolute paths
ls /full/path/to/file
# Check spelling and case sensitivity
ls -la | grep -i filename
# Verify permissions on parent directories
ls -ld /path/to/
```

Prevention: Always use tab completion and absolute paths in scripts

Issue 2: Archive extraction failures

Symptoms:

- tar command fails with "not in gzip format" error
- Archive appears corrupted

Diagnosis:

```
# Check file type
file archive.tar.gz
# Verify archive integrity
tar -tzf archive.tar.gz >/dev/null
# Check available disk space
df -h .
```

Resolution:

```
# Use correct extraction flags
tar -xf archive.tar.gz    # Auto-detect compression
# Try without compression flag
tar -xf archive.tar
# Check for partial download
ls -l archive.tar.gz
```

Issue 3: Symbolic link problems

Symptoms:

- Symlinks point to non-existent files
- Permission denied accessing through symlinks

Diagnosis:

```
# Check link status
ls -l symlink
# Test link target
readlink symlink
# Verify target existence
ls -l $(readlink symlink)
```

Resolution:

```
# Fix broken link
ln -sf correct/target symlink
# Remove and recreate
rm symlink && ln -s new/target symlink
```

Diagnostic Command Sequence

```
# File system troubleshooting workflow
pwd                     # Confirm current location
ls -la                  # List all files with details
file suspicious_file    # Determine file type
stat filename          # Detailed file information
lsof filename          # Check if file is open
```

Log File Analysis

- `/var/log/messages`: General system messages
- `/var/log/secure`: Authentication and security events
- `/var/log/boot.log`: Boot process messages
- `journalctl`: Systemd journal entries

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# File operations
ls -lah                 # List all files with details
find / -name pattern    # Search for files
cp -r source dest       # Copy recursively
mv source dest          # Move/rename

# Text processing
grep pattern file       # Search in file
sort file               # Sort lines
uniq file              # Remove duplicates
wc -l file             # Count lines

# Archives
tar -czf archive.tar.gz dir/    # Create compressed archive
tar -xzf archive.tar.gz         # Extract archive
```

Key File Locations

- **Configuration**: `/etc/` directory
- **Logs**: `/var/log/` directory
- **User data**: `/home/username/`
- **Temporary**: `/tmp/` directory

Important Patterns

- **Hidden files**: Start with `.` (dot)
- **Backup files**: Often end with `~` or `.bak`
- **Log files**: Usually in `/var/log/` with `.log` extension

Verification Commands

```
# Quick file checks
ls -l filename          # File details
file filename           # File type
stat filename           # Complete file information
du -sh directory        # Directory size
```

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between hard links and symbolic links? **Answer**: Hard links point to the same inode (same file data), while symbolic links contain the pathname of another file. Hard links cannot cross filesystems or point to directories; symbolic links can. If the original file is deleted, hard links still access the data, but symbolic links become broken.
2. **Question**: Why might you use `tar` instead of `zip` for archiving? **Answer**: Tar preserves Unix file permissions, ownership, and metadata better than zip. It's the standard in Unix/Linux environments and integrates seamlessly with compression tools. Tar also handles symbolic links correctly and is more efficient for backing up entire directory structures.
3. **Question**: When would you use `find` versus `locate`? **Answer**: Use `locate` for quick filename searches across the entire system (faster, uses database). Use `find` for complex searches based on file attributes, content, or when you need real-time results. Find searches the actual filesystem; locate searches a database that may be outdated.

Practical Scenarios

1. **Scenario**: You need to find all configuration files modified in the last 24 hours. **Solution**:

```
find /etc -name "*.conf" -mtime -1 -type f
find /etc -name "*.cfg" -mtime -1 -type f
```

2. **Scenario**: Create a backup excluding temporary files and logs. **Solution**:

```
tar --exclude="*.tmp" --exclude="*.log" --exclude="/var/log/*" \
    -czf backup.tar.gz /home/user/
```

Command Challenges

1. **Challenge**: Write a command to find all files larger than 100MB in /var directory **Answer**: `find /var -type f -size +100M` **Explanation**: `-type f` ensures only regular files, `-size +100M` finds files larger than 100 megabytes
2. **Challenge**: Create a command to show the 10 largest files in the current directory **Answer**: `ls -lS | head -11 | tail -10` **Explanation**: `-S` sorts by size (largest first), `head -11` gets first 11 lines (including header), `tail -10` shows last 10 (excluding header)

---

10. Exam Strategy

Topic-Specific Tips

- Practice file operations until they're automatic - speed matters in the exam
- Master grep patterns as they're used throughout the exam
- Know the difference between absolute and relative paths
- Understand when to use different archive formats

Common Exam Scenarios

1. **Scenario**: Find and copy configuration files to a backup directory **Approach**: Use `find` with appropriate criteria, then `cp` with `-p` to preserve attributes
2. **Scenario**: Search log files for specific error patterns **Approach**: Combine `grep`, `tail`, and date filtering for targeted searches
3. **Scenario**: Create archives of user data with specific exclusions **Approach**: Use `tar` with `--exclude` patterns for clean backups

Time Management

- **File operations**: 2-3 minutes for basic tasks
- **Archive creation**: 3-5 minutes including verification
- **Text searching**: 2-4 minutes depending on complexity
- **Quick verification**: Always test your commands before moving on

Pitfalls to Avoid

- Don't forget the `-r` flag when copying directories
- Remember that Linux is case-sensitive
- Always verify archive contents before considering task complete
- Watch out for hidden files when copying directories
- Test symbolic links after creation

Summary

Key Takeaways

- **File management is fundamental** - these skills are used in every exam task
- **Master text processing tools** - grep, sort, and cut are essential for analysis
- **Archive operations are common** - know tar syntax and compression options
- **Practice makes perfect** - file operations must be automatic for exam success

Critical Commands to Remember

```
ls -la                          # List files with details
find /path -name "pattern"      # Search for files
grep -r "pattern" directory     # Search text recursively
tar -czf archive.tar.gz dir/    # Create compressed archive
ln -s target linkname           # Create symbolic link
```

Next Steps

- Continue to Module 03: User & Group Management
- Practice file operations in the Vagrant environment
- Review related topics: File Permissions, Storage

Navigation: ← System Installation | Index | Next → User Management

2.5 03 - User & Group Management

Navigation: ← File Management | Index | Next → File Permissions

1. Executive Summary

Topic Scope: User account creation, modification, deletion, group management, and password policies in RHEL 10

RHCSA Relevance: Critical exam topic - user management appears in multiple exam tasks

Exam Weight: High - Essential system administration skill tested frequently

Prerequisites: Understanding of Linux file system and basic command line operations

Related Topics: File Permissions, SELinux Management, SSH Configuration

2. Conceptual Foundation

Core Theory

User and group management in RHEL 10 is based on the traditional Unix model with modern enhancements:

- **User accounts**: Unique identities with UID, home directory, and shell
- **Groups**: Collections of users for permission management (primary and supplementary)
- **System accounts**: Special accounts for services and daemons (UID < 1000)
- **Regular users**: Human users with interactive login capabilities (UID ≥ 1000)
- **Password policies**: Rules governing password complexity and expiration

Real-World Applications

- **Multi-user environments**: Corporate servers with multiple administrators
- **Service accounts**: Running applications with specific privileges
- **Temporary access**: Creating accounts for contractors or temporary staff
- **Security compliance**: Implementing password policies for regulatory requirements
- **Resource management**: Controlling access to files and system resources

Common Misconceptions

- **Root is UID 0**: Root always has UID 0, but UID 0 doesn't always mean "root" name
- **Group membership**: Users can belong to multiple groups simultaneously
- **Home directories**: Not automatically deleted when users are removed
- **Shell access**: Users can exist without shell access (nologin)
- **Password expiration**: Affects login but not running processes

Key Terminology

- **UID**: User Identifier (numeric ID for user accounts)
- **GID**: Group Identifier (numeric ID for groups)
- **Primary group**: User's main group (stored in /etc/passwd)
- **Supplementary groups**: Additional groups a user belongs to
- **Shadow file**: Encrypted password storage with aging information
- **Login shell**: Program executed when user logs in
- **Home directory**: User's personal directory space
- **Skeleton directory**: Template for new user home directories

---

3. Command Mastery

User Management Commands

```
# Creating users
useradd username                    # Basic user creation
useradd -u 1500 -g users username   # Specify UID and primary group
useradd -G wheel,admin username     # Add to supplementary groups
useradd -s /bin/bash username       # Specify shell
useradd -d /custom/home username    # Custom home directory
useradd -m username                 # Create home directory
useradd -c "Full Name" username     # Add comment/description

# Modifying users
usermod -l newname oldname          # Change username
usermod -u 1600 username            # Change UID
usermod -g newgroup username        # Change primary group
usermod -aG group username          # Add to supplementary group
usermod -G group1,group2 username   # Set all supplementary groups
usermod -s /sbin/nologin username   # Change shell
usermod -L username                 # Lock account
usermod -U username                 # Unlock account
usermod -d /new/home username       # Change home directory

# Removing users
userdel username                    # Delete user (keep home)
userdel -r username                 # Delete user and home directory
```

Group Management Commands

```
# Creating groups
groupadd groupname                  # Basic group creation
groupadd -g 2000 groupname         # Specify GID
groupadd -r systemgroup            # Create system group

# Modifying groups
groupmod -n newname oldname         # Rename group
groupmod -g 2500 groupname         # Change GID

# Group membership
gpasswd -a username groupname       # Add user to group
gpasswd -d username groupname       # Remove user from group
gpasswd -A admin groupname          # Set group administrator

# Removing groups
groupdel groupname                  # Delete group
```

Password Management Commands

```
# Setting passwords
passwd username                    # Set/change password
passwd -l username                 # Lock password
passwd -u username                 # Unlock password
passwd -d username                 # Delete password (disable)
passwd -e username                 # Expire password (force change)

# Password aging
chage -M 90 username               # Max age 90 days
chage -m 7 username                # Min age 7 days
chage -W 14 username               # Warning 14 days before expiry
chage -I 30 username               # Inactive 30 days after expiry
chage -E 2024-12-31 username       # Account expires on date
chage -l username                  # List aging information
```

Information Commands

```
# User information
id username                        # Show UID, GID, and groups
groups username                    # Show group memberships
finger username                    # Detailed user information (if available)
who                               # Currently logged-in users
w                                 # Detailed who information
last username                      # Login history
lastb                             # Failed login attempts

# System information
cat /etc/passwd                    # All user accounts
cat /etc/group                     # All groups
cat /etc/shadow                    # Password information (root only)
getent passwd username             # Get user info from all sources
getent group groupname             # Get group info from all sources
```

Command Reference Table

Command	Purpose	Key Options	Example
`useradd`	Create user account	`-u`, `-g`, `-G`, `-s`, `-d`	`useradd -G wheel john`
`usermod`	Modify user account	`-aG`, `-L`, `-U`, `-s`	`usermod -aG admins john`
`userdel`	Delete user account	`-r`	`userdel -r john`
`groupadd`	Create group	`-g`, `-r`	`groupadd -g 2000 developers`
`passwd`	Manage passwords	`-l`, `-u`, `-e`	`passwd -e john`
`chage`	Password aging	`-M`, `-m`, `-W`, `-E`	`chage -M 90 john`

4. Procedural Workflows

Standard Procedure: Creating a New User

1. **Plan user requirements**

```
# Determine: UID, primary group, supplementary groups, shell, home directory
```

2. **Create the user account**

```
useradd -u 1500 -g users -G wheel,developers -s /bin/bash -m username
```

3. **Set initial password**

```
passwd username
# Force password change on first login
chage -d 0 username
```

4. **Configure password policy**

```
chage -M 90 -m 7 -W 14 username
```

5. **Verify account creation**

```
id username
ls -ld /home/username
getent passwd username
```

Standard Procedure: User Account Maintenance

1. **Regular account review**

```
# Check for unused accounts
last | grep username
# Review password aging
chage -l username
```

2. **Modify account as needed**

```
# Add to new group
usermod -aG newgroup username
# Change shell
usermod -s /bin/zsh username
```

3. **Handle account issues**

```
# Temporarily lock account
usermod -L username
# Force password change
passwd -e username
```

Decision Tree: Account Creation Strategy

```
New User Request
├── Regular user?
│   ├── Standard UID range (≥1000)
│   ├── Create home directory
│   └── Interactive shell
├── Service account?
│   ├── System UID range (<1000)
│   ├── No home directory
│   └── /sbin/nologin shell
└── Temporary user?
    ├── Set account expiration
    ├── Force password change
    └── Document removal date
```

Standard Procedure: Group Management

1. **Create group structure**

```
# Create functional groups
groupadd -g 2000 developers
groupadd -g 2001 admins
groupadd -g 2002 operations
```

2. **Assign users to groups**

```
# Add existing users
usermod -aG developers user1,user2
gpasswd -a user3 admins
```

3. **Verify group memberships**

```
# Check specific user
groups username
# Check specific group
getent group groupname
```

---

5. Configuration Deep Dive

Primary Configuration Files

/etc/passwd - User Account Information

```
# Format: username:password:UID:GID:comment:home:shell
root:x:0:0:root:/root:/bin/bash
john:x:1000:1000:John Doe:/home/john:/bin/bash
apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin
```

/etc/shadow - Password Information

```
# Format: username:password:lastchange:min:max:warn:inactive:expire:reserved
root:$6$encrypted$hash:19000:0:99999:7:::
john:$6$encrypted$hash:19000:7:90:14:30:19200:
```

/etc/group - Group Information

```
# Format: groupname:password:GID:members
root:x:0:
wheel:x:10:john,admin
developers:x:2000:john,jane,bob
```

/etc/gshadow - Group Password Information

```
# Format: groupname:password:admins:members
root:::
wheel:::john,admin
developers:!!::john,jane,bob
```

Default Configuration Files

/etc/default/useradd - Default User Settings

```
# Default values for useradd command
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

/etc/login.defs - Login Definitions

```
# Password aging controls
PASS_MAX_DAYS   90
PASS_MIN_DAYS   7
PASS_WARN_AGE   14

# User ID ranges
UID_MIN         1000
UID_MAX         60000
SYS_UID_MIN     201
SYS_UID_MAX     999

# Group ID ranges
GID_MIN         1000
GID_MAX         60000
SYS_GID_MIN     201
SYS_GID_MAX     999
```

/etc/skel/ - Skeleton Directory

```
# Template files copied to new user home directories
/etc/skel/.bash_logout
/etc/skel/.bash_profile
/etc/skel/.bashrc
```

Password Policy Configuration

System-wide Password Policies

```
# /etc/security/pwquality.conf
minlen = 8              # Minimum password length
dcredit = -1           # Require at least 1 digit
ucredit = -1           # Require at least 1 uppercase
lcredit = -1           # Require at least 1 lowercase
ocredit = -1           # Require at least 1 special character
```

---

6. Hands-On Labs

Lab 6.1: Basic User Management (Asghar Ghori Style)

Objective: Create, modify, and manage user accounts with various configurations

Steps:

1. **Create users with different specifications**

```
# Regular user with defaults
useradd alice
passwd alice

# User with custom UID and group
useradd -u 1500 -g wheel -s /bin/bash bob
passwd bob

# Service account
useradd -r -s /sbin/nologin -d /var/lib/webservice webservice
```

2. **Modify existing users**

```
# Add alice to additional groups
usermod -aG wheel,developers alice

# Change bob's shell
usermod -s /bin/zsh bob

# Lock webservice account
usermod -L webservice
```

3. **Configure password policies**

```
# Set password aging for alice
chage -M 60 -m 5 -W 10 alice

# Force password change for bob
passwd -e bob
```

Verification:

```
# Verify user creation and modifications
id alice
id bob
id webservice
groups alice
chage -l alice
getent passwd | grep -E "(alice|bob|webservice)"
```

Lab 6.2: Group Management and Membership (Sander van Vugt Style)

Objective: Create groups and manage complex membership scenarios

Steps:

1. **Create organizational groups**

```
# Create department groups
groupadd -g 2000 marketing
groupadd -g 2001 sales
groupadd -g 2002 engineering

# Create role-based groups
groupadd -g 3000 managers
groupadd -g 3001 leads
```

2. **Create users and assign group memberships**

```
# Marketing team
useradd -g marketing -G leads marketing_lead
useradd -g marketing marketing_user1
useradd -g marketing marketing_user2

# Engineering team
useradd -g engineering -G managers,leads engineering_lead
useradd -g engineering engineering_dev1
useradd -g engineering engineering_dev2

# Set passwords
echo "password123" | passwd --stdin marketing_lead
echo "password123" | passwd --stdin marketing_user1
echo "password123" | passwd --stdin engineering_lead
echo "password123" | passwd --stdin engineering_dev1
```

3. **Modify group memberships**

```
# Add cross-functional team members
usermod -aG sales marketing_lead
usermod -aG marketing engineering_lead

# Use gpasswd for group management
gpasswd -a marketing_user1 leads
gpasswd -A marketing_lead marketing
```

Verification:

```
# Verify group structure
getent group | grep -E "(marketing|sales|engineering|managers|leads)"
# Check user memberships
groups marketing_lead
groups engineering_lead
# Verify group administrators
getent gshadow | grep marketing
```

Lab 6.3: Advanced User Account Scenarios (Synthesis Challenge)

Objective: Handle complex user management scenarios combining both methodologies

Scenario: Set up a development environment with different user types and access requirements

Requirements:

- Create system service accounts
- Set up developer accounts with specific group memberships
- Implement password policies and account expiration
- Handle temporary contractor accounts

Solution Steps:

1. **Create service accounts for applications**

```
# Database service account
useradd -r -u 500 -g daemon -s /sbin/nologin -d /var/lib/database database

# Web service account
useradd -r -u 501 -g daemon -s /sbin/nologin -d /var/lib/webapp webapp

# Backup service account
useradd -r -u 502 -g daemon -s /bin/bash -d /var/lib/backup backup
```

2. **Create developer environment**

```
# Create developer groups
groupadd -g 5000 developers
groupadd -g 5001 senior_devs
groupadd -g 5002 devops

# Create developer accounts
useradd -g developers -G wheel -s /bin/bash -c "Senior Developer" senior_dev1
useradd -g developers -s /bin/bash -c "Junior Developer" junior_dev1
useradd -g developers -G devops,wheel -s /bin/bash -c "DevOps Engineer" devops1

# Set strong password policies for developers
for user in senior_dev1 junior_dev1 devops1; do
    passwd $user
    chage -M 30 -m 3 -W 5 $user
done
```

3. **Handle contractor accounts**

```
# Create temporary contractor account (expires in 90 days)
future_date=$(date -d "+90 days" +%Y-%m-%d)
useradd -g developers -s /bin/bash -c "Contractor" -e $future_date contractor1
passwd contractor1

# Force password change on first login
chage -d 0 contractor1

# Set shorter password validity
chage -M 14 -m 1 -W 3 contractor1
```

4. **Verification and documentation**

```
# Generate user report
echo "=== Service Accounts ===" > user_report.txt
getent passwd | awk -F: '$3 < 1000 && $3 != 0 {print $1, $3, $7}' >> user_report.txt

echo -e "\n=== Developer Accounts ===" >> user_report.txt
getent passwd | awk -F: '$3 >= 1000 {print $1, $3, $5}' >> user_report.txt

echo -e "\n=== Group Memberships ===" >> user_report.txt
for user in $(getent passwd | awk -F: '$3 >= 1000 {print $1}'); do
    echo "$user: $(groups $user | cut -d: -f2)" >> user_report.txt
done

# Check account expiration
echo -e "\n=== Account Expiration ===" >> user_report.txt
chage -l contractor1 | grep "Account expires" >> user_report.txt
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: User Cannot Login

Symptoms:

- Authentication failures
- Account locked messages
- Permission denied errors

Diagnosis:

```
# Check account status
passwd -S username
chage -l username
# Check login attempts
lastb username
# Verify home directory
ls -ld /home/username
# Check shell validity
grep username /etc/passwd
```

Resolution:

```
# Unlock account if locked
passwd -u username
usermod -U username
# Fix expired password
passwd -e username
# Correct home directory permissions
chown username:username /home/username
chmod 700 /home/username
# Fix invalid shell
usermod -s /bin/bash username
```

Prevention: Implement regular account audits and proper password policies

Issue 2: Group Permission Problems

Symptoms:

- Users cannot access group files
- "Permission denied" for group resources
- Inconsistent group memberships

Diagnosis:

```
# Check current group membership
groups username
id username
# Verify group exists
getent group groupname
# Check if user needs to re-login
# (group changes require new login)
```

Resolution:

```
# Add user to correct group
usermod -aG groupname username
# Or use gpasswd
gpasswd -a username groupname
# Verify group membership
getent group groupname
# User must logout and login again
```

Issue 3: UID/GID Conflicts

Symptoms:

- User creation fails with "UID already exists"
- File ownership shows numbers instead of names
- Permission inconsistencies

Diagnosis:

```
# Check for UID conflicts
getent passwd | sort -t: -k3 -n | uniq -D -f2
# Check for GID conflicts
getent group | sort -t: -k3 -n | uniq -D -f2
# Find files owned by numeric UIDs
find / -nouser -o -nogroup 2>/dev/null
```

Resolution:

```
# Change conflicting UID
usermod -u newuid username
# Change conflicting GID
groupmod -g newgid groupname
# Update file ownership
find /home/username -uid olduid -exec chown username {} \;
```

Diagnostic Command Sequence

```
# User account troubleshooting workflow
getent passwd username      # Verify account exists
id username                 # Check UID/GID and groups
chage -l username          # Check password aging
passwd -S username         # Check password status
ls -ld /home/username      # Verify home directory
last username              # Check login history
```

Log File Analysis

- `/var/log/secure`: Authentication events, login attempts
- `/var/log/messages`: General system messages including user management
- `/var/log/audit/audit.log`: SELinux denials related to user operations
- `journalctl -u systemd-logind`: Login service messages

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# User management
useradd -G wheel username   # Create user with sudo access
usermod -aG group username  # Add user to group
userdel -r username         # Delete user and home directory
passwd username             # Set password

# Group management
groupadd groupname          # Create group
gpasswd -a user group      # Add user to group
groupdel groupname         # Delete group

# Information
id username                # Show user/group IDs
groups username            # Show group memberships
chage -l username          # Show password aging info
```

Key File Locations

- **User accounts**: `/etc/passwd`
- **Password hashes**: `/etc/shadow`
- **Group information**: `/etc/group`
- **Group passwords**: `/etc/gshadow`
- **User defaults**: `/etc/default/useradd`
- **Login policies**: `/etc/login.defs`
- **Skeleton directory**: `/etc/skel/`

Important UID/GID Ranges

- **Root**: UID 0, GID 0
- **System accounts**: UID 1-999
- **Regular users**: UID ≥ 1000
- **System groups**: GID 1-999
- **Regular groups**: GID ≥ 1000

Password Aging Parameters

- **Maximum age**: `-M days` (default 99999)
- **Minimum age**: `-m days` (default 0)
- **Warning period**: `-W days` (default 7)
- **Inactive period**: `-I days` (account locked after password expires)

- **Expiration date**: `-E date` (account expires)

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between primary and supplementary groups? **Answer**: A primary group is a user's default group (stored in /etc/passwd, field 4) used for file creation. Supplementary groups are additional groups a user belongs to, providing access to resources owned by those groups. Users can have one primary group but multiple supplementary groups.
2. **Question**: Why might you use a system account instead of a regular user account? **Answer**: System accounts (UID < 1000) are designed for services and daemons. They typically don't have home directories, use /sbin/nologin as shell, and follow the principle of least privilege. This provides better security isolation and prevents interactive login for service accounts.
3. **Question**: What happens when you lock a user account with `usermod -L`? **Answer**: Account locking prepends an exclamation mark (!) to the password hash in /etc/shadow, preventing password authentication. However, the user might still login using SSH keys. For complete access blocking, also set shell to /sbin/nologin and consider expiring the account.

Practical Scenarios

1. **Scenario**: Create a contractor account that expires in 30 days and must change password every 14 days. **Solution**:

```
future_date=$(date -d "+30 days" +%Y-%m-%d)
useradd -e $future_date -s /bin/bash contractor
passwd contractor
chage -M 14 -m 1 -W 3 -d 0 contractor
```

2. **Scenario**: A user reports they can't access files owned by the "projects" group despite being added to it. **Solution**: The user needs to logout and login again for group membership changes to take effect, or use `newgrp projects` to switch to the new group in the current session.

Command Challenges

1. **Challenge**: Write a command to show all users with UID between 1000 and 2000. **Answer**: `getent passwd | awk -F: '$3 >= 1000 && $3 <= 2000 {print $1, $3}'` **Explanation**: Uses getent to get all passwd entries, awk to filter by UID range in field 3

2. **Challenge**: Create a user with no login shell, custom home directory, and specific UID. **Answer**: `useradd -u 1555 -d /opt/service -s /sbin/nologin -m serviceuser` **Explanation**: `-u` sets UID, `-d` sets custom home, `-s` sets shell, `-m` creates home directory

---

10. Exam Strategy

Topic-Specific Tips

- Always verify user creation with `id username` and `getent passwd username`
- Remember that group changes require logout/login or `newgrp` to take effect
- Use `chage -l` to verify password policies are correctly applied
- Practice creating users with multiple requirements in single commands

Common Exam Scenarios

1. **Scenario**: Create users with specific group memberships and password policies **Approach**: Use `useradd` with multiple options, then `chage` for password aging
2. **Scenario**: Troubleshoot user access problems **Approach**: Check account status, group memberships, and home directory permissions
3. **Scenario**: Set up service accounts for applications **Approach**: Use system UID range, /sbin/nologin shell, and appropriate group

Time Management

- **Basic user creation**: 2-3 minutes including verification
- **Complex user with groups and policies**: 4-5 minutes
- **Troubleshooting user issues**: 5-7 minutes depending on complexity
- **Always verify**: Use `id` and `groups` commands to confirm

Pitfalls to Avoid

- Don't forget `-m` flag when creating home directories with `useradd`
- Remember that `usermod -G` replaces all supplementary groups (use `-aG` to append)
- Always set passwords after creating users
- Verify group membership changes take effect (may need re-login)
- Check that service accounts have appropriate shells and home directories

---

Summary

Key Takeaways

- **User and group management is foundational** - required for virtually all system administration
- **Understand the difference between primary and supplementary groups** - critical for file permissions
- **Master password policies and account aging** - important for security compliance
- **System accounts vs. regular users** - different configuration requirements and security implications

Critical Commands to Remember

```
useradd -G wheel -s /bin/bash -m username    # Create user with sudo access
usermod -aG groupname username               # Add user to supplementary group
passwd username                              # Set password
chage -M 90 -m 7 -W 14 username             # Set password aging policy
id username                                  # Verify user configuration
```

Next Steps

- Continue to Module 04: File Permissions
- Practice user management in the Vagrant environment
- Review related topics: SELinux, SSH Configuration

Navigation: ← File Management | Index | Next → File Permissions

2.6 04 - File Permissions & Access Control

Navigation: ← User Management | Index | Next → Process Management

1. Executive Summary

Topic Scope: File permissions, ownership, and umask configuration in RHEL 10

RHCSA Relevance: Critical security topic - file permissions are fundamental to Linux security model

Exam Weight: High - Permission management appears in multiple exam scenarios

RHEL 10 Exam Note: ACLs (setfacl/getfacl) and special permissions (setuid/setgid/sticky bit) are **no longer RHCSA exam objectives** as of RHEL 10. They are retained below as supplementary reference material.

Prerequisites: Understanding of users, groups, and basic file operations

Related Topics: User Management, SELinux, Security

2. Conceptual Foundation

Core Theory

Linux file permissions operate on a three-tier model:

- **Owner (user)**: The file/directory owner's permissions
- **Group**: Permissions for the file's group members
- **Other**: Permissions for all other users
- **Permission types**: Read (r), Write (w), Execute (x)

Real-World Applications

- **System security**: Protecting sensitive configuration files
- **Collaboration**: Shared directories for team projects
- **Service accounts**: Restricting application access to specific files
- **Backup systems**: Ensuring backup files are readable only by authorized users
- **Web servers**: Setting appropriate permissions for web content

Common Misconceptions

- **Directory permissions**: Execute permission on directories means "traverse" not "run"
- **Group permissions**: Group permission applies to primary group, not all user's groups
- **Root override**: Root can read/write most files regardless of permissions (but not execute)

Key Terminology

- **Octal notation**: Numeric representation of permissions (755, 644, etc.)
- **Symbolic notation**: Letter-based permission representation (rwxr-xr-x)
- **umask**: Default permission mask for new files and directories

---

3. Command Mastery

Basic Permission Commands

```
# View permissions
ls -l file                    # Show detailed permissions
ls -ld directory              # Show directory permissions
stat file                    # Detailed file information including permissions

# Change permissions (symbolic)
chmod u+x file                # Add execute for owner
chmod g-w file                # Remove write for group
chmod o=r file                # Set other to read-only
chmod a+r file                # Add read for all (user, group, other)
chmod u=rwx,g=rx,o=r file     # Set specific permissions for each

# Change permissions (octal)
chmod 755 file                # rwxr-xr-x
chmod 644 file                # rw-r--r--
chmod 600 file                # rw-------
chmod 777 file                # rwxrwxrwx (dangerous!)
```

Ownership Commands

```
# Change ownership
chown user file               # Change owner only
chown user:group file         # Change owner and group
chown :group file             # Change group only
chgrp group file              # Change group (alternative method)

# Recursive ownership changes
chown -R user:group directory # Change ownership recursively
chmod -R 755 directory        # Change permissions recursively
```

umask Configuration

```
# View current umask
umask                          # Show current umask in octal
umask -S                       # Show current umask symbolically

# Set umask
umask 022                      # Set umask to 022 (default for many systems)
umask 027                      # More restrictive umask
umask u=rwx,g=rx,o=           # Symbolic umask setting
```

Finding Files by Permissions

```
# Find by permission patterns
find / -perm 777               # Find world-writable files
find / -perm -o+w              # Find other-writable files

# Find by ownership
find / -user username          # Find files owned by user
find / -group groupname        # Find files owned by group
find / -nouser                 # Find files with no valid owner
find / -nogroup                # Find files with no valid group
```

Command Reference Table

Command	Purpose	Key Options	Example
`chmod`	Change file permissions	`u+x` , `g-w` , `755` , `-R`	`chmod 644 file.txt`
`chown`	Change file ownership	`user:group` , `-R`	`chown alice:staff file`
`chgrp`	Change group ownership	`-R`	`chgrp developers file`
`umask`	Set default permissions	`-S`	`umask 022`

---

4. Procedural Workflows

Standard Procedure: Setting Up Secure File Permissions

1. **Determine access requirements**

```
# Identify who needs what access:
# - Owner: full control
# - Group: read/execute
# - Others: no access
```

2. **Set basic permissions**

```
chmod 750 file_or_directory
# or symbolically:
chmod u=rwx,g=rx,o= file_or_directory
```

3. **Set appropriate ownership**

```
chown owner:group file_or_directory
```

4. **Verify permissions**

```
ls -l file_or_directory
stat file_or_directory
```

Standard Procedure: Shared Directory Setup

1. **Create directory with appropriate permissions**

```
mkdir /shared/project
chmod 775 /shared/project
chown :projectteam /shared/project
```

2. **Test directory functionality**

```
# Test as different users
touch /shared/project/testfile
ls -l /shared/project/testfile
```

Decision Tree: Permission Strategy

```
Permission Requirements
├── Simple user/group/other? → Use chmod with octal notation
├── Shared directory? → Use group permissions + chmod
└── System service? → Restrict to specific user/group only
```

Standard Procedure: Security Audit

1. **Find potentially dangerous permissions**

```
# World-writable files
find / -type f -perm -002 2>/dev/null

# Files with no owner/group
find / \( -nouser -o -nogroup \) 2>/dev/null
```

2. **Review critical system files**

```
ls -l /etc/passwd /etc/shadow /etc/group
ls -l /etc/sudoers
ls -ld /tmp /var/tmp
```

3. **Check home directory permissions**

```
ls -ld /home/*
find /home -type d -perm -002 2>/dev/null
```

---

5. Configuration Deep Dive

Permission Calculation

Octal Permission Values

```
# Read (r) = 4, Write (w) = 2, Execute (x) = 1

# Common permission combinations:
755 = rwxr-xr-x    # Standard executable/directory
644 = rw-r--r--    # Standard file
600 = rw-------    # Private file
777 = rwxrwxrwx    # Full permissions (dangerous)
000 = ---------    # No permissions
```

Special Permission Values

```
# Special permissions (added to regular permissions):
4000 = setuid bit
2000 = setgid bit
1000 = sticky bit

# Combined examples:
4755 = rwsr-xr-x   # setuid + 755
2755 = rwxr-sr-x   # setgid + 755
1755 = rwxr-xr-t   # sticky + 755
6755 = rwsr-sr-x   # setuid + setgid + 755
```

umask Configuration Files

System-wide umask

```
# /etc/bashrc or /etc/profile
umask 022          # Default for most users

# /root/.bashrc
umask 027          # More restrictive for root
```

Per-user umask

```
# ~/.bashrc or ~/.bash_profile
umask 077          # Very restrictive (user-only access)
```

---

6. Hands-On Labs

Lab 6.1: Basic Permission Management (Asghar Ghori Style)

Objective: Master fundamental permission operations and special bits

Steps:

1. **Create test environment**

```
mkdir ~/permissions_lab
cd ~/permissions_lab
touch file1 file2 file3
mkdir dir1 dir2 dir3
```

2. **Practice basic permissions**

```
# Set different permission combinations
chmod 644 file1          # Standard file permissions
chmod 755 dir1           # Standard directory permissions
chmod 600 file2          # Private file
chmod 700 dir2           # Private directory

# Use symbolic notation
chmod u=rw,g=r,o= file3  # Owner: rw, Group: r, Other: none
chmod u+x dir3           # Add execute for owner
```

Verification:

```
ls -la                      # Check all permissions
stat file1 file2 file3      # Detailed permission info
```

Lab 6.2: Ownership and umask Configuration

Objective: Practice ownership changes and understand umask

Steps:

1. **Practice ownership changes**

```
mkdir ~/ownership_lab
cd ~/ownership_lab
touch file1 file2
mkdir dir1

# Change ownership (requires root or owning the files)
chown :users file1
chgrp users dir1
```

2. **Understand umask effects**

```
# Check current umask
umask
umask -S

# Set restrictive umask and test
umask 077
touch private_file
mkdir private_dir
ls -l private_file         # Should be rw-------
ls -ld private_dir         # Should be rwx------

# Set collaborative umask
umask 002
touch shared_file
ls -l shared_file          # Should be rw-rw-r--
```

3. **Configure persistent umask**

```
# Add umask to ~/.bashrc for persistence
echo "umask 027" >> ~/.bashrc
```

Verification:

```
ls -la ~/ownership_lab/
stat ~/ownership_lab/private_file
umask
```

Lab 6.3: Shared Directory Setup (Synthesis Challenge)

Objective: Create a collaborative workspace using standard permissions

Scenario: Set up a project directory where a team can collaborate.

Requirements:

- Project team members: read/write access
- Others: no access

Solution Steps:

1. **Create directory structure**

```
sudo mkdir -p /projects/webapp
sudo groupadd developers

# Add users to group (assuming users exist)
# sudo usermod -aG developers alice
# sudo usermod -aG developers bob
```

2. **Set permissions and ownership**

```
sudo chown :developers /projects/webapp
sudo chmod 770 /projects/webapp
```

3. **Verify**

```
ls -ld /projects/webapp/
# Test as a member of the developers group
touch /projects/webapp/testfile
ls -l /projects/webapp/testfile
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Permission Denied Errors

Symptoms:

- Users cannot access files they should be able to read
- Applications fail with permission errors
- "Permission denied" messages in logs

Diagnosis:

```
# Check file permissions and ownership
ls -la filename
stat filename

# Check user's group memberships
id username
groups username

# Check parent directory permissions
ls -ld /path/to/
ls -ld /path/

# Check for ACLs
getfacl filename
```

Resolution:

```
# Fix basic permissions
chmod 644 filename          # For regular files
chmod 755 directoryname     # For directories

# Fix ownership
chown user:group filename

# Add user to appropriate group
usermod -aG groupname username

# Set ACLs if needed
setfacl -m u:username:r-- filename
```

Prevention: Always verify permissions after creating files and directories

Diagnostic Command Sequence

```
# Permission troubleshooting workflow
ls -la filename             # Check basic permissions
stat filename               # Detailed permission info
id username                 # Check user context
groups username             # Check group memberships
lsattr filename             # Check extended attributes
```

Log File Analysis

- `/var/log/messages`: General permission-related errors
- `/var/log/secure`: Authentication and access control events
- `/var/log/audit/audit.log`: SELinux and detailed access events
- **Application logs**: Specific permission errors from services

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Permissions
chmod 755 file              # Standard executable/directory
chmod 644 file              # Standard file
chown user:group file       # Change ownership
chgrp group file            # Change group
umask 022                   # Set default permission mask
```

Octal Permission Reference

- **755**: rwxr-xr-x (directories, executables)
- **644**: rw-r--r-- (regular files)
- **600**: rw------- (private files)
- **777**: rwxrwxrwx (dangerous, avoid)
- **000**: --------- (no permissions)

Common umask Values

- **022**: Default (644 for files, 755 for directories)
- **027**: Group-friendly (640 for files, 750 for directories)
- **077**: Private (600 for files, 700 for directories)

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What is the difference between octal and symbolic permission notation? **Answer**: Octal notation uses numbers (e.g., `755` = rwxr-xr-x) where each digit represents user/group/other permissions (r=4, w=2, x=1). Symbolic notation uses letters (e.g., `u+x`, `g=rw`, `o-w`) to add, set, or remove specific permissions.
2. **Question**: What does the execute permission mean on a directory? **Answer**: On a directory, execute (x) means "traverse" — the ability to cd into the directory and access files within it. Without execute on a directory, you cannot access its contents even if you have read permission (which only lets you list filenames).
3. **Question**: How does umask affect new file and directory permissions? **Answer**: umask subtracts from the default permissions. New files start at 666 (no execute) and directories at 777. With umask 022, files become 644 (rw-r--r--) and directories become 755 (rwxr-xr-x).

Practical Scenarios

1. **Scenario**: Create a directory where only members of the "project" group can access files. **Solution**:

```
mkdir /project
chown :project /project
chmod 770 /project
```

2. **Scenario**: A user creates files that are world-readable by default. Make them private. **Solution**: Set a restrictive umask:

```
umask 077
# Or add to ~/.bashrc for persistence
echo "umask 077" >> ~/.bashrc
```

Command Challenges

1. **Challenge**: Change ownership of all files in /data to user "admin" and group "staff" recursively. **Answer**: `chown -R admin:staff /data` **Explanation**: `-R` applies the change recursively to all files and subdirectories.
2. **Challenge**: Find all files owned by a user who no longer exists on the system. **Answer**: `find / -nouser 2>/dev/null` **Explanation**: `-nouser` finds files whose numeric UID doesn't match any user in /etc/passwd.

10. Exam Strategy

Topic-Specific Tips

- Master octal notation — it's faster than symbolic for complex permissions
- Always verify permissions after setting them with `ls -l`
- Know how umask affects default permissions for new files and directories
- Practice chmod, chown, chgrp until they are second nature

Common Exam Scenarios

1. **Scenario**: Set up a shared directory for a group **Approach**: Create group, set group ownership with `chown :group dir`, set `chmod 770` or `chmod 775`
2. **Scenario**: Set appropriate permissions on configuration files **Approach**: Restrictive permissions like `chmod 600` for sensitive files, `chmod 644` for readable configs

Time Management

- **Basic permission tasks**: 2-3 minutes including verification
- **Ownership changes**: 1-2 minutes
- **Always verify**: Use `ls -l` and `stat` to confirm settings

Pitfalls to Avoid

- Don't forget that directory execute permission is needed for traversal
- Remember that changing group membership requires logout/login to take effect
- Don't use 777 permissions unless absolutely necessary (security risk)
- Remember umask subtracts from defaults: files start at 666, directories at 777

Summary

Key Takeaways

- **File permissions are the foundation of Linux security** — master chmod, chown, chgrp
- **umask controls default permissions** — understand its impact on file creation
- **Ownership determines access** — proper user:group assignment is critical

Critical Commands to Remember

```
chmod 755 directory                      # Standard directory permissions
chmod 644 file                          # Standard file permissions
chown user:group file                   # Change ownership
chgrp group file                       # Change group
umask 022                              # Set default permissions
find / -nouser                         # Find orphaned files
```

Next Steps

- Continue to Module 05: Process & Service Management
- Practice permission scenarios in the Vagrant environment
- Review related topics: User Management, SELinux

Supplementary Reference: ACLs (Not on RHEL 10 Exam)

Note: Access Control Lists (ACLs) are no longer an RHCSA exam objective as of RHEL 10. This section is retained for reference only.

ACL Commands

```
# View ACLs
getfacl file                  # Show ACL information
getfacl -R directory          # Recursive ACL display

# Set ACLs
setfacl -m u:username:rwx file        # Set user ACL
setfacl -m g:groupname:rx file        # Set group ACL
setfacl -m d:u:username:rwx directory # Set default user ACL

# Remove ACLs
setfacl -x u:username file            # Remove user ACL
setfacl -b file                       # Remove all ACLs
setfacl -k directory                  # Remove default ACLs

# Copy ACLs
getfacl file1 | setfacl --set-file=- file2  # Copy ACLs between files
```

Supplementary Reference: Special Permissions (Not on RHEL 10 Exam)

Note: setuid, setgid, and sticky bit are no longer RHCSA exam objectives as of RHEL 10. This section is retained for reference only.

Special Permission Commands

```
# setuid (4000) — execute file with owner's privileges
chmod u+s file               # Add setuid bit
chmod 4755 file              # Set permissions with setuid

# setgid (2000) — execute with group's privileges / inherit group on directories
chmod g+s directory          # Set group inheritance on directory
chmod 2755 directory         # Set permissions with setgid

# Sticky bit (1000) — prevent deletion by non-owners
chmod +t directory           # Add sticky bit to directory
chmod 1755 directory         # Set permissions with sticky bit
```

Special Permission Values

- **4000**: setuid bit
- **2000**: setgid bit
- **1000**: sticky bit

Finding Special Permission Files

```
find / -perm -4000 2>/dev/null   # Find setuid files
find / -perm -2000 2>/dev/null   # Find setgid files
find / -perm -1000 2>/dev/null   # Find sticky bit directories
```

Navigation: ← User Management | Index | Next → Process Management

2.7 05 - Process & Service Management

Navigation: ← File Permissions | Index | Next → Package Management

1. Executive Summary

Topic Scope: Process monitoring, control, systemd service management, and system targets in RHEL 10

RHCSA Relevance: Critical operational skill - process and service management is essential for system administration

Exam Weight: High - Service management and process control appear in multiple exam scenarios

Prerequisites: Basic understanding of Linux command line and system concepts

Related Topics: Boot Process, Logging, Troubleshooting

2. Conceptual Foundation

Core Theory

RHEL 10 uses systemd as the init system and service manager, which fundamentally changed how processes and services are managed:

- **Process hierarchy**: All processes descend from PID 1 (systemd)
- **Service units**: Standardized configuration for system services
- **Targets**: Groups of services that define system states
- **Dependencies**: Services can depend on other services or system states
- **Process control**: Signal-based communication for process management

Real-World Applications

- **Web server management**: Starting, stopping, and monitoring Apache/Nginx
- **Database operations**: Managing MySQL/PostgreSQL service lifecycle
- **System maintenance**: Controlling system services during maintenance windows
- **Performance troubleshooting**: Identifying resource-intensive processes
- **Service reliability**: Ensuring critical services restart automatically

Common Misconceptions

- **systemctl vs service**: Old `service` command still works but `systemctl` is preferred
- **Enable vs start**: Services must be both enabled (auto-start) and started (running now)
- **Process vs service**: Not all processes are services; services are managed processes
- **Kill vs terminate**: Different signals have different effects on processes
- **Targets vs runlevels**: systemd targets are more flexible than traditional runlevels

Key Terminology

- **Process**: Running instance of a program with unique PID
- **Service**: Long-running process managed by systemd
- **Unit**: systemd configuration object (service, target, socket, etc.)
- **Target**: Group of units that define a system state
- **Signal**: Inter-process communication mechanism
- **Daemon**: Background process providing system services
- **Job**: systemd task for starting/stopping units
- **Slice**: Resource management group for processes

---

3. Command Mastery

Process Monitoring Commands

```
# Basic process listing
ps aux                      # All processes, detailed format
ps -ef                      # All processes, full format
ps -eLf                     # Include threads
ps -o pid,ppid,user,comm    # Custom output format

# Dynamic process monitoring
top                         # Interactive process viewer
htop                        # Enhanced interactive viewer (if installed)
top -u username             # Filter by user
top -p PID                  # Monitor specific process

# Process hierarchy
pstree                      # Process tree view
pstree -p                   # Include PIDs
pstree username             # User's process tree

# Process information
pgrep processname           # Find process IDs by name
pidof processname           # Alternative to pgrep
ps -C processname           # Show processes by command name
lsof -p PID                 # Files opened by process
```

Process Control Commands

```
# Process signals
kill PID                    # Send TERM signal (graceful termination)
kill -9 PID                 # Send KILL signal (force termination)
kill -HUP PID               # Send HUP signal (often reload config)
kill -USR1 PID              # Send USR1 signal (user-defined)
killall processname         # Kill all processes by name
pkill -u username           # Kill processes by user

# Job control
nohup command &             # Run command immune to hangups
command &                   # Run in background
jobs                        # List active jobs
fg %jobnumber              # Bring job to foreground
bg %jobnumber              # Send job to background
disown %jobnumber          # Remove job from shell's job table

# Process priority
nice -n 10 command          # Start with adjusted priority
renice 5 PID               # Change priority of running process
ionice -c 2 -n 7 PID       # Set I/O priority
```

systemd Service Management

```
# Service status and control
systemctl status servicename        # Show service status
systemctl start servicename         # Start service
systemctl stop servicename          # Stop service
systemctl restart servicename       # Restart service
systemctl reload servicename        # Reload configuration
systemctl enable servicename        # Enable auto-start at boot
systemctl disable servicename       # Disable auto-start
systemctl mask servicename          # Prevent service from starting
systemctl unmask servicename        # Remove mask

# Service information
systemctl is-active servicename     # Check if running
systemctl is-enabled servicename    # Check if enabled
systemctl is-failed servicename     # Check if failed
systemctl list-units --type=service # List all services
systemctl list-unit-files --type=service # List service files

# Service dependencies
systemctl list-dependencies servicename    # Show dependencies
systemctl list-dependencies --reverse servicename # Reverse dependencies
```

System Targets

```
# Target management
systemctl get-default               # Show default target
systemctl set-default target       # Set default target
systemctl isolate target           # Switch to target immediately
systemctl list-units --type=target # List active targets

# Common targets
systemctl isolate rescue.target    # Single-user mode
systemctl isolate multi-user.target # Multi-user (no GUI)
systemctl isolate graphical.target  # Multi-user with GUI
systemctl isolate poweroff.target   # Shutdown system
systemctl isolate reboot.target     # Restart system
```

Advanced systemd Commands

```
# Unit file management
systemctl daemon-reload             # Reload systemd configuration
systemctl edit servicename          # Edit service override
systemctl cat servicename           # Show service unit file
systemctl show servicename          # Show all service properties

# System control
systemctl poweroff                  # Shutdown system
systemctl reboot                    # Restart system
systemctl suspend                   # Suspend to RAM
systemctl hibernate                 # Suspend to disk
systemctl hybrid-sleep              # Suspend to both RAM and disk
```

Command Reference Table

Command	Purpose	Key Options	Example
`ps`	List processes	`aux` , `-ef` , `-C`	`ps aux | grep httpd`
`top`	Monitor processes	`-u` , `-p`	`top -u apache`
`kill`	Send signals	`-9` , `-HUP` , `-USR1`	`kill -HUP 1234`
`systemctl`	Manage services	`start` , `stop` , `enable` , `status`	`systemctl enable httpd`
`pgrep`	Find processes	`-u` , `-f`	`pgrep -u root sshd`
`nohup`	Run immune to hangups	`&`	`nohup ./script.sh &`

---

4. Procedural Workflows

Standard Procedure: Service Installation and Configuration

1. **Install and enable service**

```
dnf install servicename
systemctl enable servicename
systemctl start servicename
```

2. **Verify service status**

```
systemctl status servicename
systemctl is-active servicename
systemctl is-enabled servicename
```

3. **Configure service**

```
# Edit main configuration
vim /etc/servicename/config.conf

# Or create systemd override
systemctl edit servicename
```

4. **Apply changes**

```
systemctl daemon-reload
systemctl restart servicename
systemctl status servicename
```

Standard Procedure: Process Troubleshooting

1. **Identify problematic process**

```
top                    # Look for high CPU/memory usage
ps aux --sort=-%cpu    # Sort by CPU usage
ps aux --sort=-%mem    # Sort by memory usage
```

2. **Gather process information**

```
ps -p PID -o pid,ppid,user,cmd
lsof -p PID           # Files opened by process
strace -p PID         # System calls (use carefully)
```

3. **Take appropriate action**

```
# Graceful termination
kill PID

# Force termination if needed
kill -9 PID

# Or restart associated service
systemctl restart servicename
```

Decision Tree: Process Management Strategy

```
Process Issue
├── High CPU usage?
│   ├── Expected behavior? → Monitor and document
│   └── Unexpected? → Investigate cause, consider restart
├── High memory usage?
│   ├── Memory leak suspected? → Restart service
│   └── Normal operation? → Check system capacity
├── Process not responding?
│   ├── Try graceful termination → kill PID
│   └── If unsuccessful → kill -9 PID
└── Service keeps failing?
    ├── Check logs → journalctl -u servicename
    ├── Verify configuration
    └── Check dependencies
```

Standard Procedure: System Target Management

1. **Check current target**

```
systemctl get-default
who -r               # Alternative method
```

2. **Change target temporarily**

```
systemctl isolate multi-user.target
```

3. **Change default target**

```
systemctl set-default graphical.target
```

4. **Verify target change**

```
systemctl get-default
systemctl list-units --type=target
```

---

5. Configuration Deep Dive

systemd Unit Files

Service Unit Structure

```
# /etc/systemd/system/myservice.service
[Unit]
Description=My Custom Service
After=network.target
Requires=network.target

[Service]
Type=simple
User=myuser
Group=mygroup
ExecStart=/usr/local/bin/myservice
ExecStop=/bin/kill -TERM $MAINPID
Restart=always
RestartSec=30

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

Common Unit File Sections

```
[Unit]
Description=Service description
Documentation=man:service(8)
After=dependency.service
Before=dependent.service
Requires=hard.dependency
Wants=soft.dependency
Conflicts=conflicting.service

[Service]
Type=simple|forking|oneshot|notify|idle
ExecStart=/path/to/executable
ExecStop=/path/to/stop/command
ExecReload=/path/to/reload/command
User=username
Group=groupname
Restart=no|on-success|on-failure|on-abnormal|on-watchdog|always
RestartSec=seconds

[Install]
WantedBy=target.target
RequiredBy=target.target
Also=other.service
```

Process Priority and Nice Values

Understanding Priority

```
# Nice values range from -20 (highest priority) to 19 (lowest priority)
# Default nice value is 0
# Only root can set negative nice values

# Examples:
nice -n 10 ./cpu-intensive-task     # Lower priority
nice -n -10 ./critical-task         # Higher priority (root only)
renice 5 1234                       # Change running process priority
```

Resource Control with systemd

Systemd Slices and Resource Limits

```
# Create custom slice for resource management
# /etc/systemd/system/myapp.slice
[Unit]
Description=My Application Slice
Before=slices.target

[Slice]
CPUQuota=50%
MemoryLimit=1G
TasksMax=100
```

Service Resource Limits

```
# In service unit file [Service] section:
CPUQuota=50%                # Limit CPU usage to 50%
MemoryLimit=512M            # Limit memory to 512MB
TasksMax=50                 # Limit number of tasks
IOWeight=100                # I/O priority weight
```

---

6. Hands-On Labs

Lab 6.1: Process Monitoring and Control (Asghar Ghori Style)

Objective: Master process identification, monitoring, and control techniques

Steps:

1. **Start background processes for testing**

```
# Create some test processes
sleep 300 &
PID1=$!
dd if=/dev/zero of=/dev/null &
PID2=$!
find / -name "*.log" > /dev/null 2>&1 &
PID3=$!

echo "Started processes: $PID1 $PID2 $PID3"
```

2. **Practice process monitoring**

```
# View all processes
ps aux | head -20

# Find specific processes
ps aux | grep sleep
pgrep sleep
pidof sleep

# Monitor resource usage
top -p $PID1,$PID2,$PID3
# Press 'q' to quit top

# View process tree
pstree $$    # Show tree from current shell
```

3. **Practice process control**

```
# Send different signals
kill -USR1 $PID1        # User signal (sleep will ignore)
kill -STOP $PID2        # Suspend process
kill -CONT $PID2        # Resume process

# Check process status
ps -o pid,state,comm -p $PID1,$PID2,$PID3

# Terminate processes
kill $PID1              # Graceful termination
kill -9 $PID2           # Force termination
killall find            # Kill by name
```

4. **Practice job control**

```
# Start job in foreground
sleep 100
# Press Ctrl+Z to suspend

# Manage jobs
jobs                    # List jobs
bg %1                   # Send to background
fg %1                   # Bring to foreground
# Press Ctrl+C to terminate
```

Verification:

```
# Verify no test processes remain
ps aux | grep -E "(sleep|dd|find)" | grep -v grep
jobs                    # Should show no jobs
```

Lab 6.2: systemd Service Management (Sander van Vugt Style)

Objective: Master systemd service lifecycle and configuration

Steps:

1. **Explore existing services**

```
# List all services
systemctl list-units --type=service --all

# Check specific service status
systemctl status sshd
systemctl is-active sshd
systemctl is-enabled sshd

# View service dependencies
systemctl list-dependencies sshd
systemctl list-dependencies --reverse sshd
```

2. **Practice service control**

```
# Work with chronyd (time synchronization)
systemctl status chronyd
systemctl stop chronyd
systemctl status chronyd
systemctl start chronyd
systemctl reload chronyd    # If reload is supported
systemctl restart chronyd
```

3. **Configure service startup**

```
# Check and modify service enablement
systemctl is-enabled chronyd
systemctl disable chronyd
systemctl is-enabled chronyd
systemctl enable chronyd
systemctl is-enabled chronyd
```

4. **Explore service configuration**

```
# View service unit file
systemctl cat chronyd

# Show all service properties
systemctl show chronyd | head -20

# Create service override (don't actually modify)
systemctl edit chronyd --drop-in=custom
# This would open an editor, but let's skip actual changes
```

Verification:

```
# Verify service is properly configured
systemctl status chronyd
systemctl is-active chronyd
systemctl is-enabled chronyd
```

Lab 6.3: Custom Service Creation (Synthesis Challenge)

Objective: Create and manage a custom systemd service

Scenario: Create a custom web log monitor service that watches for specific patterns in web server logs

Requirements:

- Service runs as non-root user
- Automatically restarts if it fails
- Starts after network is available
- Can be stopped and started with systemctl

Solution Steps:

1. **Create the monitoring script**

```
sudo mkdir -p /opt/logmonitor
sudo tee /opt/logmonitor/weblog-monitor.sh << 'EOF'
#!/bin/bash
# Simple web log monitor

LOGFILE="/var/log/httpd/access_log"
MONITOR_LOG="/var/log/logmonitor.log"

# Create log file if it doesn't exist
touch "$MONITOR_LOG"

echo "$(date): Web log monitor started" >> "$MONITOR_LOG"

while true; do
    # Monitor for 404 errors (customize as needed)
    if tail -n 1 "$LOGFILE" 2>/dev/null | grep -q " 404 "; then
        echo "$(date): 404 error detected" >> "$MONITOR_LOG"
    fi
    sleep 5
done
EOF

sudo chmod +x /opt/logmonitor/weblog-monitor.sh
```

2. **Create service user**

```
sudo useradd -r -s /sbin/nologin -d /opt/logmonitor logmonitor
sudo chown -R logmonitor:logmonitor /opt/logmonitor
sudo touch /var/log/logmonitor.log
sudo chown logmonitor:logmonitor /var/log/logmonitor.log
```

3. **Create systemd service unit**

```
sudo tee /etc/systemd/system/weblog-monitor.service << 'EOF'
[Unit]
Description=Web Log Monitor Service
Documentation=man:tail(1)
After=network.target httpd.service
Wants=network.target

[Service]
Type=simple
User=logmonitor
Group=logmonitor
ExecStart=/opt/logmonitor/weblog-monitor.sh
ExecStop=/bin/kill -TERM $MAINPID
Restart=always
RestartSec=30
StandardOutput=journal
StandardError=journal

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

4. **Enable and test the service**

```
# Reload systemd configuration
sudo systemctl daemon-reload

# Enable and start the service
sudo systemctl enable weblog-monitor
sudo systemctl start weblog-monitor

# Check service status
sudo systemctl status weblog-monitor

# Test service functionality
sudo systemctl stop weblog-monitor
sudo systemctl start weblog-monitor
sudo systemctl restart weblog-monitor
```

5. **Monitor and verify service**

```
# Check service logs
sudo journalctl -u weblog-monitor -f --no-pager
# Press Ctrl+C to stop following

# Verify service is running as correct user
ps aux | grep weblog-monitor

# Check if service auto-restarts (simulate crash)
sudo pkill -f weblog-monitor.sh
sleep 35  # Wait for restart
sudo systemctl status weblog-monitor
```

Verification:

```
# Complete service verification
sudo systemctl is-active weblog-monitor
sudo systemctl is-enabled weblog-monitor
sudo journalctl -u weblog-monitor --no-pager -n 10
ls -l /var/log/logmonitor.log
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Service Won't Start

Symptoms:

- `systemctl start` fails with error
- Service shows "failed" status
- Application not responding

Diagnosis:

```
# Check service status and errors
systemctl status servicename
systemctl --failed

# View detailed logs
journalctl -u servicename --no-pager
journalctl -xe

# Check service dependencies
systemctl list-dependencies servicename
```

Resolution:

```
# Fix common issues
systemctl daemon-reload      # Reload if unit file changed
systemctl reset-failed      # Clear failed status

# Check and fix configuration
systemctl cat servicename   # Review unit file
vim /etc/servicename/config  # Fix application config

# Restart dependencies if needed
systemctl restart dependency.service
systemctl start servicename
```

Prevention: Always test configuration changes before applying to production

Issue 2: High System Load

Symptoms:

- System responds slowly
- High load average
- Applications timing out

Diagnosis:

```
# Check system load
uptime
top
htop (if available)

# Identify resource-intensive processes
ps aux --sort=-%cpu | head -10
ps aux --sort=-%mem | head -10

# Check I/O activity
iotop (if available)
iostat 1 5
```

Resolution:

```
# Adjust process priorities
renice 10 PID              # Lower priority
ionice -c 3 PID           # Idle I/O class

# Restart problematic services
systemctl restart servicename

# Kill runaway processes if necessary
kill PID
kill -9 PID  # If graceful doesn't work
```

Issue 3: Process Won't Terminate

Symptoms:

- Process ignores TERM signal
- `kill` command has no effect
- Process shows as zombie

Diagnosis:

```
# Check process state
ps -o pid,ppid,state,comm -p PID

# Check what process is doing
lsof -p PID
strace -p PID (use carefully)

# Check for zombie processes
ps aux | grep -E '<defunct>|<zombie>'
```

Resolution:

```
# Try escalating signals
kill PID           # TERM signal
sleep 5
kill -QUIT PID     # QUIT signal
sleep 5
kill -9 PID        # KILL signal (last resort)

# For zombie processes, kill parent
ps -o pid,ppid -p PID
kill PPID
```

Diagnostic Command Sequence

```
# Service troubleshooting workflow
systemctl status servicename    # Check service status
journalctl -u servicename       # Check service logs
systemctl cat servicename       # Review unit file
ps aux | grep servicename       # Check if process running
lsof -i :port                   # Check port usage
```

Log File Analysis

- `journalctl`: Primary systemd log viewer
- `/var/log/messages`: General system messages
- `/var/log/secure`: Authentication and security events
- **Service-specific logs**: Application logs in `/var/log/`

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Process monitoring
ps aux                          # List all processes
top                            # Interactive process monitor
pgrep processname              # Find process by name
kill PID                       # Terminate process

# Service management
systemctl start servicename    # Start service
systemctl stop servicename     # Stop service
systemctl enable servicename   # Enable auto-start
systemctl status servicename   # Check service status

# System targets
systemctl get-default          # Show default target
systemctl isolate target       # Switch to target
```

Important Signals

- **TERM (15)**: Graceful termination (default for `kill`)
- **KILL (9)**: Force termination (cannot be caught)
- **HUP (1)**: Hangup (often used to reload config)
- **STOP (19)**: Suspend process
- **CONT (18)**: Resume suspended process

Common systemd Targets

- **poweroff.target**: Shutdown system
- **rescue.target**: Single-user mode
- **multi-user.target**: Multi-user, no GUI
- **graphical.target**: Multi-user with GUI
- **reboot.target**: Restart system

Process States

- **R**: Running or runnable
- **S**: Sleeping (waiting for event)
- **D**: Uninterruptible sleep (usually I/O)
- **T**: Stopped (by signal)
- **Z**: Zombie (terminated but not cleaned up)

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between `systemctl start` and `systemctl enable`? **Answer**: `start` immediately begins running the service, while `enable` configures the service to start automatically at boot. A service can be enabled but not running, or running but not enabled. For complete setup, you typically need both commands.
2. **Question**: Why might a process become a zombie? **Answer**: A zombie process occurs when a child process has finished executing but its parent hasn't read its exit status yet. The process entry remains in the process table until the parent calls wait(). If the parent never calls wait() or terminates, the zombie persists.
3. **Question**: What happens when you send SIGKILL (-9) to a process? **Answer**: SIGKILL cannot be caught or ignored by the process - the kernel immediately terminates it. This bypasses any cleanup code, potentially leaving files open, shared memory segments allocated, or other resources in an inconsistent state. Use only as a last resort.

Practical Scenarios

1. **Scenario**: A web service keeps crashing and needs to restart automatically. **Solution**: Configure the systemd service with `Restart=always` and `RestartSec=30` in the `[Service]` section of the unit file.
2. **Scenario**: You need to temporarily stop a service without preventing future automatic starts. **Solution**: Use `systemctl stop servicename`. This stops the service but leaves it enabled, so it will still start automatically at next boot.

Command Challenges

1. **Challenge**: Find all processes owned by the user "apache" and show their CPU usage. **Answer**: `ps -u apache -o pid,user,%cpu,comm` **Explanation**: `-u apache` filters by user, `-o` specifies custom output format
2. **Challenge**: Create a command to monitor the top 5 CPU-consuming processes, updating every 2 seconds. **Answer**: `top -n 0 -d 2 | head -12 | tail -5` or use `watch -n 2 "ps aux --sort=-%cpu | head -5"`

---

10. Exam Strategy

Topic-Specific Tips

- Always use `systemctl status` to verify service operations
- Remember to both `enable` and `start` services for complete setup

- Practice signal usage - know when to use graceful vs force termination
- Understand systemd dependencies - services may fail if dependencies aren't met

Common Exam Scenarios

1. **Scenario**: Configure a service to start automatically at boot **Approach**: Use `systemctl enable servicename` then `systemctl start servicename`
2. **Scenario**: Troubleshoot a service that won't start **Approach**: Check `systemctl status`, review `journalctl -u servicename`, verify dependencies
3. **Scenario**: Find and terminate a runaway process **Approach**: Use `top` or `ps` to identify, then `kill` with appropriate signal

Time Management

- **Basic service operations**: 2-3 minutes including verification
- **Process troubleshooting**: 5-7 minutes depending on complexity
- **Custom service creation**: 8-10 minutes for complete setup
- **Always verify**: Check service status after changes

Pitfalls to Avoid

- Don't forget to `systemctl daemon-reload` after editing unit files
- Remember that stopping a service doesn't disable it (still starts at boot)
- Avoid `kill -9` unless absolutely necessary - try graceful termination first
- Check service dependencies - some services require others to be running
- Verify both that service is running AND enabled for boot

Summary

Key Takeaways

- **systemd is the modern service manager** - master its commands and concepts
- **Process management requires understanding signals** - different signals have different effects
- **Service dependencies matter** - services may fail if dependencies aren't met
- **Always verify changes** - check status after making service modifications

Critical Commands to Remember

```
systemctl start servicename                # Start service now
systemctl enable servicename               # Start service at boot
systemctl status servicename               # Check service status
ps aux                                      # List all processes
kill PID                                    # Graceful process termination
journalctl -u servicename                  # View service logs
```

Next Steps

- Continue to Module 06: Package Management
- Practice service management in the Vagrant environment
- Review related topics: Boot Process, Logging

Navigation: ← File Permissions | Index | Next → Package Management

2.8 06 - Package Management

Navigation: ← Process Management | Index | Next → Storage & LVM

1. Executive Summary

Topic Scope: DNF package manager, RPM operations, repository management, and software installation in RHEL 10

RHCSA Relevance: Essential system administration skill - package management is fundamental for maintaining RHEL systems

Exam Weight: Medium-High - Package operations appear in various exam scenarios

Prerequisites: Basic understanding of Linux file system and command line operations

Related Topics: System Installation, Service Management, Security

2. Conceptual Foundation

Core Theory

RHEL 10 uses DNF (Dandified YUM) as the primary package manager, which provides:

- **Dependency resolution**: Automatic handling of package dependencies
- **Repository management**: Centralized software distribution points
- **Transaction safety**: Rollback capability for failed installations
- **Metadata caching**: Improved performance through local metadata storage
- **Modular content**: Support for application streams and modules

Real-World Applications

- **System maintenance**: Installing security updates and patches
- **Software deployment**: Installing applications and development tools
- **Environment setup**: Configuring development or production environments
- **Security compliance**: Keeping systems updated with latest security fixes
- **Custom repositories**: Managing internal software distributions

Common Misconceptions

- **DNF vs YUM**: DNF is the successor to YUM with better dependency resolution
- **Package vs RPM**: Packages are distributed as RPMs, but package managers handle dependencies
- **Repository priority**: Higher numbers mean lower priority (opposite of what you might expect)
- **Clean vs remove**: Clean removes cache, remove uninstalls packages
- **Modules vs packages**: Modules provide different versions/streams of software

Key Terminology

- **Package**: Software bundle with metadata, dependencies, and installation scripts
- **Repository**: Collection of packages available for installation
- **Metadata**: Information about packages, dependencies, and repositories
- **Transaction**: Complete package operation (install, update, remove)
- **Dependency**: Required packages for software to function
- **Module**: Collection of packages representing different versions of software
- **Stream**: Specific version of a module
- **Profile**: Set of packages within a module for specific use cases

---

3. Command Mastery

Basic DNF Operations

```
# Package installation and removal
dnf install packagename              # Install package
dnf install package1 package2       # Install multiple packages
dnf remove packagename               # Remove package
dnf autoremove                       # Remove unneeded dependencies
dnf reinstall packagename            # Reinstall package

# Package updates
dnf update                           # Update all packages
dnf update packagename               # Update specific package
dnf check-update                     # Check for available updates
dnf upgrade                          # Alias for update

# Package information
dnf info packagename                 # Show package details
dnf list installed                   # List installed packages
dnf list available                   # List available packages
dnf list updates                     # List available updates
dnf search keyword                   # Search packages by keyword
dnf provides */filename              # Find package providing file
```

Advanced DNF Operations

```
# Package groups
dnf grouplist                        # List available groups
dnf groupinstall "Group Name"        # Install package group
dnf groupremove "Group Name"         # Remove package group
dnf groupinfo "Group Name"           # Show group information

# Transaction history
dnf history                          # Show transaction history
dnf history info transaction-id      # Details of specific transaction
dnf history undo transaction-id      # Undo transaction
dnf history redo transaction-id      # Redo transaction

# Local package installation
dnf localinstall package.rpm         # Install local RPM
dnf install /path/to/package.rpm     # Alternative syntax

# Download operations
dnf download packagename             # Download package without installing
dnf download --resolve packagename   # Download with dependencies
```

Repository Management

```
# Repository operations
dnf repolist                         # List enabled repositories
dnf repolist --all                   # List all repositories
dnf repolist --disabled              # List disabled repositories
dnf config-manager --add-repo URL    # Add repository
dnf config-manager --enable repo     # Enable repository
dnf config-manager --disable repo    # Disable repository

# Cache management
dnf makecache                        # Download repository metadata
dnf clean all                        # Clean all cache
dnf clean packages                   # Clean downloaded packages
dnf clean metadata                   # Clean repository metadata
dnf clean expire-cache               # Clean expired cache
```

Module Management

```
# Module operations
dnf module list                      # List available modules
dnf module list --installed          # List installed modules
dnf module info modulename           # Show module information
dnf module install modulename:stream # Install specific stream
dnf module enable modulename:stream  # Enable module stream
dnf module disable modulename        # Disable module
dnf module reset modulename          # Reset module state
```

RPM Commands

```
# Package information
rpm -qa                              # List all installed packages
rpm -qi packagename                  # Query package information
rpm -ql packagename                  # List package files
rpm -qd packagename                  # List package documentation
rpm -qc packagename                  # List configuration files
rpm -qf /path/to/file               # Find package owning file

# Package verification
rpm -V packagename                   # Verify package integrity
rpm -Va                             # Verify all packages

# Package installation (not recommended, use DNF instead)
rpm -ivh package.rpm                # Install package
rpm -Uvh package.rpm                # Upgrade package
rpm -e packagename                  # Erase package
```

Command Reference Table

Command	Purpose	Key Options	Example
`dnf install`	Install packages	`-y`, `--nogpgcheck`	`dnf install -y httpd`
`dnf remove`	Remove packages	`-y`	`dnf remove -y packagename`
`dnf update`	Update packages	`-y`, `--security`	`dnf update -y`
`dnf search`	Search packages		`dnf search web server`
`dnf info`	Package information		`dnf info httpd`
`dnf history`	Transaction history	`info`, `undo`, `redo`	`dnf history undo 5`

---

4. Procedural Workflows

Standard Procedure: Software Installation

1. **Search for package**

```
dnf search keyword
dnf info packagename
```

2. **Install package**

```
dnf install -y packagename
```

3. **Verify installation**

```
dnf list installed | grep packagename
rpm -qi packagename
```

4. **Configure and start if it's a service**

```
systemctl enable --now servicename
systemctl status servicename
```

Standard Procedure: System Updates

1. **Check for updates**

```
dnf check-update
dnf list updates
```

2. **Review security updates**

```
dnf updateinfo list security
dnf updateinfo info security
```

3. **Apply updates**

```
# Test updates first
dnf update --downloadonly

# Apply all updates
dnf update -y

# Or security only
dnf update --security -y
```

4. **Verify and reboot if needed**

```
dnf history info last
# Reboot if kernel updated
needs-restarting -r
```

Decision Tree: Package Management Strategy

```
Package Task
├── Installing new software?
│   ├── Available in repositories? → dnf install
│   └── Local RPM file? → dnf localinstall
├── System maintenance?
│   ├── Security updates? → dnf update --security
│   └── Full update? → dnf update
├── Removing software?
│   ├── Remove package only? → dnf remove
│   └── Remove dependencies too? → dnf autoremove
└── Troubleshooting?
    ├── Package conflicts? → Check dnf history
    ├── Dependency issues? → dnf check
    └── Corrupted packages? → rpm -V
```

Standard Procedure: Repository Management

1. **Add new repository**

```
# Method 1: Using config-manager
dnf config-manager --add-repo https://example.com/repo

# Method 2: Manual file creation
cat > /etc/yum.repos.d/custom.repo << 'EOF'
[custom-repo]
name=Custom Repository
baseurl=https://example.com/repo
enabled=1
gpgcheck=1
gpgkey=https://example.com/repo/RPM-GPG-KEY
EOF
```

2. **Update repository metadata**

```
dnf makecache
```

3. **Verify repository**

```
dnf repolist
dnf repoinfo custom-repo
```

---

5. Configuration Deep Dive

DNF Configuration Files

Main Configuration

```
# /etc/dnf/dnf.conf
[main]
gpgcheck=1                    # Verify package signatures
installonly_limit=3           # Keep 3 kernels maximum
clean_requirements_on_remove=True  # Clean unused dependencies
best=True                     # Use best available package versions
skip_if_unavailable=False     # Fail if repository unavailable
```

Repository Configuration

```
# /etc/yum.repos.d/example.repo
[repository-id]
name=Human Readable Repository Name
baseurl=https://example.com/repo/
        file:///path/to/local/repo
mirrorlist=https://example.com/mirrors
metalink=https://example.com/metalink.xml
enabled=1                     # 1=enabled, 0=disabled
gpgcheck=1                    # 1=check signatures, 0=don't check
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-example
       https://example.com/RPM-GPG-KEY
priority=1                    # Lower number = higher priority
cost=1000                     # Repository cost (lower = preferred)
includepkgs=package1,package2 # Only include these packages
excludepkgs=package3,package4 # Exclude these packages
```

Package Groups Configuration

Common Package Groups

```
# Development tools
dnf groupinstall "Development Tools"

# Web server
dnf groupinstall "Web Server"

# Virtualization
dnf groupinstall "Virtualization Hypervisor"
dnf groupinstall "Virtualization Client"

# Desktop environments
dnf groupinstall "GNOME Desktop Environment"
dnf groupinstall "KDE Plasma Workspaces"
```

Module Configuration

Module Stream Management

```
# List available streams for a module
dnf module list nodejs

# Enable specific stream
dnf module enable nodejs:14

# Install module with specific profile
dnf module install nodejs:14/development

# Switch to different stream
dnf module reset nodejs
dnf module enable nodejs:16
dnf module install nodejs:16/minimal
```

---

6. Hands-On Labs

Lab 6.1: Basic Package Operations (Asghar Ghori Style)

Objective: Master fundamental DNF package management operations

Steps:

1. **Explore package information**

```
# Search for web server packages
dnf search "web server"
dnf search apache

# Get detailed information
dnf info httpd
dnf info nginx

# Check what's installed
dnf list installed | grep -i web
```

2. **Install and configure packages**

```
# Install web server
dnf install -y httpd

# Install additional packages
dnf install -y wget curl

# Verify installations
dnf list installed | grep -E "(httpd|wget|curl)"
rpm -qi httpd
```

3. **Manage package groups**

```
# List available groups
dnf grouplist | head -20

# Get information about development tools
dnf groupinfo "Development Tools"

# Install development group (if not already installed)
dnf groupinstall -y "Development Tools"

# List installed groups
dnf grouplist --installed
```

4. **Practice package removal**

```
# Remove a package
dnf remove -y wget

# Check for orphaned dependencies
dnf autoremove

# Reinstall package
dnf install -y wget
```

Verification:

```
# Verify package operations
dnf history | head -10
dnf list installed | grep -E "(httpd|curl)"
systemctl status httpd
```

Lab 6.2: Repository Management (Sander van Vugt Style)

Objective: Configure and manage software repositories

Steps:

1. **Explore existing repositories**

```
# List current repositories
dnf repolist
dnf repolist --all

# Get detailed repository information
dnf repoinfo baseos
dnf repoinfo appstream

# Check repository configuration files
ls /etc/yum.repos.d/
cat /etc/yum.repos.d/redhat.repo
```

2. **Add EPEL repository (Extra Packages for Enterprise Linux)**

```
# Install EPEL release package
dnf install -y epel-release

# Verify EPEL repository is added
dnf repolist | grep epel

# Search for packages in EPEL
dnf search --enablerepo=epel htop
dnf info --enablerepo=epel htop
```

3. **Practice repository management**

```
# Disable a repository temporarily
dnf config-manager --disable epel
dnf repolist | grep epel

# Enable repository
dnf config-manager --enable epel
dnf repolist | grep epel

# Update repository metadata
dnf makecache
dnf clean expire-cache
```

4. **Work with repository priorities**

```
# View repository configuration
cat /etc/yum.repos.d/epel.repo

# Install package from specific repository
dnf install --enablerepo=epel -y htop

# Verify installation
which htop
htop --version
```

Verification:

```
# Verify repository configuration
dnf repolist
dnf list installed | grep epel
ls -la /etc/yum.repos.d/
```

Lab 6.3: Advanced Package Management (Synthesis Challenge)

Objective: Handle complex package scenarios including modules, local packages, and troubleshooting

Scenario: Set up a development environment with specific software versions and handle package conflicts

Requirements:

- Install development tools
- Configure specific module streams
- Install local packages
- Handle dependency conflicts
- Document the configuration

Solution Steps:

1. **Set up development environment**

```
# Install base development tools
dnf groupinstall -y "Development Tools"

# Install additional development packages
dnf install -y git vim-enhanced tree

# List installed development packages
dnf groupinfo "Development Tools" | grep "Installed Packages"
```

2. **Work with modules for specific versions**

```
# List available modules
dnf module list | head -20

# Work with Node.js module (example)
dnf module list nodejs

# Enable specific stream and install
dnf module enable -y nodejs:16
dnf module install -y nodejs:16/development

# Verify module installation
node --version
npm --version
```

3. **Handle local package installation**

```
# Create a directory for downloaded packages
mkdir ~/packages
cd ~/packages

# Download a package without installing
dnf download --resolve tree

# List downloaded packages
ls -la *.rpm

# Reinstall from local file
dnf remove -y tree
dnf localinstall -y tree-*.rpm
```

4. **Troubleshoot package issues**

```
# Check for package problems
dnf check

# Verify package integrity
rpm -Va | head -10

# Check transaction history
dnf history | head -10
dnf history info last

# Clean up if needed
dnf autoremove -y
dnf clean all
```

5. **Document the environment**

```
# Create environment documentation
cat > ~/development-environment.md << 'EOF'
# Development Environment Setup

## Installed Components
- Development Tools group
- Node.js version 16.x with development profile
- Git, Vim, Tree utilities

## Repository Configuration
- BaseOS and AppStream (default RHEL repositories)
- EPEL repository for additional packages

## Module Configuration
- nodejs:16 stream enabled with development profile

## Package Verification Commands
```bash
dnf grouplist --installed
dnf module list --installed
node --version && npm --version
```
EOF

Create package list backup
dnf list installed > ~/installed-packages-$(date +%Y%m%d).txt
```

Verification:

```
# Complete environment verification
dnf grouplist --installed | grep -i development
dnf module list --installed
node --version
npm --version
git --version
tree --version
cat ~/development-environment.md
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Package Installation Failures

Symptoms:

- "Nothing to do" message when installing
- Dependency conflicts
- Repository errors

Diagnosis:

```
# Check if package exists
dnf search packagename
dnf info packagename

# Check repository status
dnf repolist
dnf makecache

# Check for conflicts
dnf check
```

Resolution:

```
# Update repository metadata
dnf clean expire-cache
dnf makecache

# Try alternative package name
dnf search keyword

# Install with specific repository
dnf install --enablerepo=repository packagename

# Force reinstall if corrupted
dnf reinstall packagename
```

Prevention: Regular repository metadata updates and system maintenance

Issue 2: Dependency Hell

Symptoms:

- Circular dependency errors
- "Package does not exist" for dependencies
- Transaction test failures

Diagnosis:

```
# Check package dependencies
dnf deplist packagename
rpm -qR packagename

# Review transaction history
dnf history info problematic-transaction
```

Resolution:

```
# Reset transaction
dnf history undo problematic-transaction

# Use different installation method
dnf shell
> install package1
> install package2
> run

# Exclude problematic packages temporarily
dnf install packagename --exclude=problematic-package
```

Issue 3: Repository Problems

Symptoms:

- "Repository not found" errors
- GPG signature failures
- Slow or failed downloads

Diagnosis:

```
# Check repository configuration
cat /etc/yum.repos.d/problematic.repo

# Test repository connectivity
curl -I repository-baseurl

# Check GPG keys
rpm -qa gpg-pubkey*
```

Resolution:

```
# Fix repository URL
vim /etc/yum.repos.d/problematic.repo

# Import missing GPG keys
rpm --import /path/to/RPM-GPG-KEY

# Disable GPG check temporarily (not recommended)
dnf install --nogpgcheck packagename

# Update repository metadata
dnf clean all
dnf makecache
```

Diagnostic Command Sequence

```
# Package troubleshooting workflow
dnf check                    # Check for problems
dnf repolist                 # Verify repositories
dnf history                  # Check recent transactions
rpm -Va | head -20          # Verify package integrity
df -h                       # Check disk space
```

Log File Analysis

- `/var/log/dnf.log`: DNF transaction logs
- `/var/log/dnf.librepo.log`: Repository access logs
- `/var/log/dnf.rpm.log`: RPM transaction logs
- `/var/log/hawkey.log`: Dependency resolution logs

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Basic operations
dnf install packagename      # Install package
dnf remove packagename       # Remove package
dnf update                   # Update all packages
dnf search keyword           # Search packages

# Information
dnf info packagename         # Package details
dnf list installed          # Installed packages
dnf provides filename       # Find package providing file

# Repository management
dnf repolist                 # List repositories
dnf makecache               # Update metadata
dnf clean all               # Clean cache
```

Common Package Groups

- **"Development Tools"**: Compilers, build tools
- **"Web Server"**: Apache HTTP server and related
- **"Virtualization Host"**: KVM and virtualization tools
- **"Security Tools"**: Security-related packages

RPM Query Options

- `-qa`: List all installed packages
- `-qi`: Package information
- `-ql`: List package files
- `-qf`: Find package owning file
- `-qd`: List documentation files
- `-qc`: List configuration files

DNF History Operations

- `dnf history`: Show transaction history
- `dnf history info ID`: Transaction details
- `dnf history undo ID`: Undo transaction
- `dnf history redo ID`: Redo transaction

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between `dnf remove` and `dnf autoremove`? **Answer**: `dnf remove` removes specified packages and their dependencies that are no longer needed by other packages. `dnf autoremove` removes packages that were installed as dependencies but are no longer required by any installed packages. Use `autoremove` to clean up orphaned dependencies.
2. **Question**: Why would you use modules instead of regular packages? **Answer**: Modules provide different versions (streams) of software that aren't available as separate packages. For example, you can choose Node.js 14, 16, or 18 streams. Modules also offer different profiles (minimal, development, etc.) with different sets of packages for specific use cases.
3. **Question**: When should you use `rpm` commands instead of `dnf`? **Answer**: Use `rpm` for querying information about installed packages, verifying package integrity, and installing local packages when you don't need dependency resolution. Use `dnf` for installation, updates, and dependency management. Never use `rpm -e` to remove packages - use `dnf remove` instead.

Practical Scenarios

1. **Scenario**: You need to install a specific version of Python that's not available in the default repositories. **Solution**: Check for Python modules with `dnf module list python*`, enable the desired stream with `dnf module enable python39:3.9`, then install with `dnf module install python39:3.9`.
2. **Scenario**: A package installation failed halfway through and the system is in an inconsistent state. **Solution**: Use `dnf history` to find the failed transaction, then `dnf history undo transaction-id` to roll back the changes.

Command Challenges

1. **Challenge**: Find which package provides the `netstat` command. **Answer**: `dnf provides */netstat` or `dnf whatprovides netstat` **Explanation**: The `provides` subcommand searches for packages that provide a specific file or command
2. **Challenge**: Install all available security updates without installing other updates. **Answer**: `dnf update --security` **Explanation**: The `--security` flag limits updates to only security-related packages

---

10. Exam Strategy

Topic-Specific Tips

- Master the difference between `dnf` and `rpm` - use the right tool for each task
- Practice repository management - know how to add, enable, and disable repos
- Understand package groups - they're often used in exam scenarios
- Remember that modules provide version flexibility

Common Exam Scenarios

1. **Scenario**: Install software development tools **Approach**: Use `dnf groupinstall "Development Tools"` for comprehensive setup
2. **Scenario**: Configure custom repository **Approach**: Create repository file in `/etc/yum.repos.d/` or use `dnf config-manager --add-repo`
3. **Scenario**: Troubleshoot failed package installation **Approach**: Check `dnf history`, use `dnf check`, verify repository configuration

Time Management

- **Package installation**: 2-3 minutes including verification
- **Repository configuration**: 4-5 minutes for complete setup
- **Package troubleshooting**: 5-7 minutes depending on issue complexity
- **Always verify**: Check installation with `dnf list installed` or `rpm -q`

Pitfalls to Avoid

- Don't mix `rpm` and `dnf` operations (use `dnf` for dependency management)
- Remember to enable repositories after adding them
- Always update metadata (`dnf makecache`) after adding repositories
- Don't forget to enable services after installing server packages
- Use `dnf history` to track and potentially undo problematic changes

Summary

Key Takeaways

- **DNF is the modern package manager** - it replaces YUM with better dependency resolution
- **Repository management is crucial** - proper repository configuration enables software installation

- **Modules provide version flexibility** - use them for software requiring specific versions
- **Package groups simplify installation** - use them for installing related software collections

Critical Commands to Remember

```
dnf install packagename                    # Install software
dnf update                                 # Update system
dnf groupinstall "Group Name"              # Install package group
dnf repolist                               # List repositories
dnf search keyword                         # Find packages
dnf history                                # View transaction history
```

Next Steps

- Continue to Module 07: Storage & LVM
- Practice package management in the Vagrant environment
- Review related topics: System Installation, Service Management

---

Navigation: ← Process Management | Index | Next → Storage & LVM

2.9 07 - Storage & LVM Management

Navigation: ← Package Management | Index | Next → Network Configuration

1. Executive Summary

Topic Scope: Disk partitioning, LVM (Logical Volume Management), filesystem creation, mounting, and swap management in RHEL 10

RHCSA Relevance: Critical exam topic - storage management is a major component of RHCSA certification

Exam Weight: High - Storage tasks are complex and carry significant point values

Prerequisites: Understanding of Linux file system hierarchy and basic command line operations

Related Topics: System Installation, File Management, Boot Process

2. Conceptual Foundation

Core Theory

Storage management in RHEL 10 involves multiple layers:

- **Physical storage**: Hard drives, SSDs, network storage
- **Partitions**: Logical divisions of physical storage
- **Volume management**: LVM provides flexibility between partitions and filesystems
- **Filesystems**: Data organization structures (XFS, ext4, etc.)
- **Mount points**: Directory locations where filesystems are accessible

Real-World Applications

- **Database servers**: Managing storage for database files with growth requirements
- **Web servers**: Organizing storage for logs, content, and temporary files
- **Development systems**: Creating isolated storage areas for different projects
- **Backup systems**: Managing storage for backup data with retention policies
- **Virtual environments**: Providing flexible storage to virtual machines

Common Misconceptions

- **LVM complexity**: LVM adds flexibility, not just complexity
- **XFS vs ext4**: XFS is better for large files, ext4 for small files
- **Partition vs LVM**: LVM provides better flexibility for production systems
- **Swap size**: Modern systems need less swap than traditional recommendations
- **Online resizing**: XFS can only grow, ext4 can shrink and grow

Key Terminology

- **Physical Volume (PV)**: Physical storage device or partition used by LVM
- **Volume Group (VG)**: Collection of physical volumes acting as single storage pool
- **Logical Volume (LV)**: Virtual partition created from volume group space
- **Physical Extent (PE)**: Smallest unit of space allocation in LVM
- **Logical Extent (LE)**: Mapping unit from logical volume to physical extents
- **Mount point**: Directory where filesystem is attached to directory tree
- **Thin pool**: LVM storage pool for thin provisioning with on-demand allocation
- **Thin volume**: Logical volume that draws space from a thin pool as data is written
- **fstab**: Configuration file for automatic filesystem mounting
- **UUID**: Universally Unique Identifier for devices and filesystems

---

3. Command Mastery

Disk and Partition Management

```
# View disk information
lsblk                       # List block devices in tree format
lsblk -f                    # Include filesystem information
fdisk -l                    # List all partitions on all disks
fdisk -l /dev/sda           # List partitions on specific disk
blkid                       # Show device UUIDs and filesystem types
df -h                       # Show mounted filesystem usage
du -sh /path                # Show directory space usage

# Partition management with fdisk
fdisk /dev/sdb              # Interactive partition editor
# Common fdisk commands:
# n - create new partition
# d - delete partition
# t - change partition type
# w - write changes and exit
# q - quit without saving

# Partition management with parted
parted /dev/sdb             # Interactive partition editor
parted /dev/sdb print       # Show partition table
parted /dev/sdb mklabel gpt  # Create GPT partition table
parted /dev/sdb mkpart primary 1MB 100%  # Create partition

# Update kernel partition table
partprobe /dev/sdb          # Update without reboot
```

LVM Management Commands

```
# Physical Volume (PV) management
pvcreate /dev/sdb1           # Create physical volume
pvdisplay                    # Show PV details
pvs                          # Show PV summary
pvremove /dev/sdb1           # Remove PV (must be unused)

# Volume Group (VG) management
vgcreate vgname /dev/sdb1    # Create volume group
vgcreate vgname /dev/sdb1 /dev/sdc1  # VG with multiple PVs
vgdisplay                    # Show VG details
vgs                          # Show VG summary
vgextend vgname /dev/sdd1    # Add PV to VG
vgreduce vgname /dev/sdd1    # Remove PV from VG
vgremove vgname              # Remove VG (must be empty)

# Logical Volume (LV) management
lvcreate -L 2G -n lvname vgname        # Create LV with specific size
lvcreate -l 50%VG -n lvname vgname     # Create LV with percentage
lvcreate -l 100%FREE -n lvname vgname  # Use all free space
lvdisplay                              # Show LV details
lvs                                    # Show LV summary
lvextend -L +1G /dev/vgname/lvname     # Extend LV by 1GB
lvextend -L 5G /dev/vgname/lvname      # Extend LV to 5GB total
lvreduce -L -1G /dev/vgname/lvname     # Reduce LV by 1GB
lvremove /dev/vgname/lvname            # Remove LV
```

LVM Thin Provisioning

```
# Create a thin pool (allocates actual storage)
lvcreate --type thin-pool -L 5G -n mythinpool vgname

# Create thin volumes (virtual size, allocated from pool on demand)
lvcreate --virtualsize 10G --thin -n thinlv1 vgname/mythinpool
lvcreate --virtualsize 10G --thin -n thinlv2 vgname/mythinpool

# Monitor thin pool usage
lvs -o+lv_size,pool_lv,data_percent vgname
lvs -a vgname                          # Show all LVs including pool metadata

# Extend thin pool when running low
lvextend -L +5G vgname/mythinpool

# Create filesystem and mount thin volume (same as regular LV)
mkfs.xfs /dev/vgname/thinlv1
mount /dev/vgname/thinlv1 /mnt/thin1
```

Filesystem Management

```
# Create filesystems
mkfs.xfs /dev/vgname/lvname           # Create XFS filesystem
mkfs.ext4 /dev/vgname/lvname          # Create ext4 filesystem
mkfs -t xfs /dev/sdb1                 # Alternative syntax

# Filesystem information
blkid /dev/vgname/lvname              # Show filesystem UUID and type
xfs_info /dev/vgname/lvname           # XFS filesystem information
tune2fs -l /dev/vgname/lvname         # ext4 filesystem information

# Filesystem resizing
xfs_growfs /mountpoint                # Grow XFS (can only grow)
resize2fs /dev/vgname/lvname          # Resize ext4 (grow or shrink)
resize2fs /dev/vgname/lvname 3G       # Resize ext4 to specific size

# Filesystem checking and repair
xfs_repair /dev/vgname/lvname         # Repair XFS (unmounted)
fsck.ext4 /dev/vgname/lvname          # Check/repair ext4 (unmounted)
```

Mount Management

```
# Mounting filesystems
mount /dev/vgname/lvname /mnt/data    # Mount filesystem
mount -t xfs /dev/sdb1 /mnt/backup    # Mount with specific type
mount -o ro /dev/sdb1 /mnt/readonly   # Mount read-only
umount /mnt/data                      # Unmount filesystem
umount -f /mnt/data                   # Force unmount

# Persistent mounting
vim /etc/fstab                        # Edit fstab for permanent mounts
mount -a                              # Mount all fstab entries
findmnt --verify                      # Verify fstab syntax

# View mounted filesystems
mount                                 # Show all mounted filesystems
mount | column -t                     # Formatted output
findmnt                              # Tree view of mounts
df -h                                # Mounted filesystem usage
```

Swap Management

```
# Create and manage swap
mkswap /dev/vgname/swaplv              # Create swap filesystem
swapon /dev/vgname/swaplv              # Enable swap
swapoff /dev/vgname/swaplv             # Disable swap
swapon --show                          # Show active swap
swapon -a                              # Enable all swap in fstab

# Swap file creation
dd if=/dev/zero of=/swapfile bs=1M count=1024  # Create 1GB file
chmod 600 /swapfile                    # Secure permissions
mkswap /swapfile                       # Format as swap
swapon /swapfile                       # Enable swap file
```

Command Reference Table

Command	Purpose	Key Options	Example
`lsblk`	List block devices	`-f`	`lsblk -f`
`fdisk`	Partition editor	`-l`	`fdisk /dev/sdb`
`pvcreate`	Create physical volume		`pvcreate /dev/sdb1`
`vgcreate`	Create volume group		`vgcreate datavg /dev/sdb1`
`lvcreate`	Create logical volume	`-L`, `-n`	`lvcreate -L 2G -n datalv datavg`
`mkfs.xfs`	Create XFS filesystem		`mkfs.xfs /dev/datavg/datalv`

---

4. Procedural Workflows

Standard Procedure: Complete LVM Setup

1. **Prepare physical storage**

```
# Create partition (if not using whole disk)
fdisk /dev/sdb
# Create partition, set type to Linux LVM (8e)
partprobe /dev/sdb
```

2. **Create LVM structure**

```
# Create physical volume
pvcreate /dev/sdb1

# Create volume group
vgcreate datavg /dev/sdb1

# Create logical volume
lvcreate -L 2G -n datalv datavg
```

3. **Create and mount filesystem**

```
# Create filesystem
mkfs.xfs /dev/datavg/datalv

# Create mount point
mkdir -p /data

# Mount filesystem
mount /dev/datavg/datalv /data
```

4. **Make mount permanent**

```
# Add to fstab
echo "/dev/datavg/datalv /data xfs defaults 0 2" >> /etc/fstab

# Verify fstab
mount -a
df -h /data
```

Standard Procedure: Extending LVM Storage

1. **Add new physical volume**

```
# Prepare new disk/partition
pvcreate /dev/sdc1

# Add to existing volume group
vgextend datavg /dev/sdc1

# Verify VG size increased
vgs datavg
```

2. **Extend logical volume**

```
# Extend logical volume
lvextend -L +5G /dev/datavg/datalv

# Or extend to use all available space
lvextend -l +100%FREE /dev/datavg/datalv
```

3. **Extend filesystem**

```
# For XFS filesystems
xfs_growfs /data

# For ext4 filesystems
resize2fs /dev/datavg/datalv

# Verify new size
df -h /data
```

Standard Procedure: LVM Thin Provisioning Setup

Thin provisioning allows over-committing storage — thin volumes can have a combined virtual size larger than the physical pool. Space is allocated only as data is written.

1. **Create thin pool from volume group**

```
# Create a thin pool (actual physical storage)
lvcreate --type thin-pool -L 5G -n thinpool datavg
```

2. **Create thin volumes**

```
# Virtual size can exceed pool size (overprovisioning)
lvcreate --virtualsize 10G --thin -n app1 datavg/thinpool
lvcreate --virtualsize 10G --thin -n app2 datavg/thinpool
```

3. **Create filesystems and mount**

```
mkfs.xfs /dev/datavg/app1
mkfs.xfs /dev/datavg/app2
mkdir -p /srv/{app1,app2}
mount /dev/datavg/app1 /srv/app1
mount /dev/datavg/app2 /srv/app2
```

4. **Monitor pool usage and extend when needed**

```
# Check how much of the pool is actually used
lvs -o+data_percent datavg/thinpool

# Extend pool before it fills up
lvextend -L +5G datavg/thinpool
```

5. **Make mounts persistent**

```
echo "/dev/datavg/app1 /srv/app1 xfs defaults 0 2" >> /etc/fstab
echo "/dev/datavg/app2 /srv/app2 xfs defaults 0 2" >> /etc/fstab
mount -a
```

Decision Tree: Storage Strategy Selection

```
Storage Requirements
├── Simple single-disk setup?
│   ├── Basic partitioning → fdisk + mkfs + mount
│   └── Future growth expected → Use LVM even for single disk
├── Multiple disks?
│   ├── Need flexibility? → LVM with multiple PVs
│   ├── Performance priority? → RAID + LVM
│   └── Simple aggregation? → LVM spanning
├── Overprovisioned / on-demand storage?
│   └── LVM thin provisioning → thin pool + thin volumes
└── Specific use case?
    ├── Database storage → XFS on LVM for large files
    ├── Boot partition → ext4 on regular partition
    └── Swap space → LVM logical volume or swap file
```

Standard Procedure: Filesystem Migration

1. **Prepare new storage**

```
# Create new LVM structure
pvcreate /dev/sdd1
vgcreate newvg /dev/sdd1
lvcreate -L 10G -n newlv newvg
mkfs.xfs /dev/newvg/newlv
```

2. **Copy data safely**

```
# Mount new filesystem temporarily
mkdir /mnt/newdata
mount /dev/newvg/newlv /mnt/newdata

# Copy data with rsync
rsync -avxHAX /olddata/ /mnt/newdata/

# Verify data integrity
diff -r /olddata /mnt/newdata
```

3. **Switch to new storage**

```
# Update fstab
sed -i 's|/dev/oldvg/oldlv|/dev/newvg/newlv|g' /etc/fstab

# Unmount old, remount new
umount /olddata
umount /mnt/newdata
mount /dev/newvg/newlv /olddata
```

---

5. Configuration Deep Dive

/etc/fstab Configuration

fstab Entry Format

```
# Device/UUID  Mount Point  Filesystem  Options  Dump  Pass
/dev/datavg/datalv  /data  xfs  defaults  0  2
UUID=abc123-def456  /home  ext4  defaults,noatime  1  2
/dev/datavg/swaplv  swap   swap defaults  0  0
```

Common fstab Options

```
# Performance options
defaults,noatime        # Don't update access times (performance)
defaults,relatime       # Update access time relatively (compromise)

# Security options
defaults,noexec         # Prevent execution of binaries
defaults,nosuid         # Ignore suid bits
defaults,nodev          # Don't interpret device files

# Reliability options
defaults,_netdev        # Network device (wait for network)
defaults,nofail         # Continue boot if device unavailable
```

LVM Configuration Files

LVM Configuration

```
# /etc/lvm/lvm.conf
devices {
    scan = [ "/dev" ]           # Directories to scan for devices
    filter = [ "a/sda/", "r/.*/" ]  # Device filter (accept/reject)
}

activation {
    volume_list = [ "vg1", "@*" ]   # Limit activated VGs
}

backup {
    backup = 1                  # Enable metadata backups
    archive = 1                 # Enable metadata archives
}
```

Volume Group Backup and Recovery

```
# Backup VG metadata
vgcfgbackup vgname
vgcfgbackup -f /backup/vgname.conf vgname

# Restore VG metadata
vgcfgrestore -f /backup/vgname.conf vgname

# List backups
vgcfgrestore -l vgname
```

Filesystem-Specific Configuration

XFS Configuration

```
# XFS filesystem options in fstab
/dev/datavg/datalv /data xfs defaults,noatime,logbsize=256k 0 2

# XFS maintenance commands
xfs_fsr /data                   # Defragment XFS filesystem
xfs_db -r /dev/datavg/datalv    # XFS debugger (read-only)
```

ext4 Configuration

```
# ext4 tuning
tune2fs -o acl,user_xattr /dev/datavg/datalv    # Enable ACLs
tune2fs -c 50 /dev/datavg/datalv                # Check every 50 mounts
tune2fs -i 180d /dev/datavg/datalv              # Check every 180 days
```

---

6. Hands-On Labs

Lab 6.1: Basic LVM Setup (Asghar Ghori Style)

Objective: Create complete LVM storage solution from scratch

Prerequisites: Additional disk (/dev/sdb) available for testing

Steps:

1. **Explore current storage**

```
# View current storage configuration
lsblk
df -h
pvs
vgs
lvs
```

2. **Create partition for LVM**

```
# Create partition on /dev/sdb
fdisk /dev/sdb
# Commands in fdisk:
# n (new partition)
# p (primary)
# 1 (partition number)
# Enter (default start)
# Enter (default end, use whole disk)
# t (change type)
# 8e (Linux LVM)
# w (write and exit)

# Update kernel partition table
partprobe /dev/sdb

# Verify partition
lsblk /dev/sdb
```

3. **Create LVM components**

```
# Create physical volume
pvcreate /dev/sdb1
pvdisplay /dev/sdb1

# Create volume group
vgcreate labtesting /dev/sdb1
vgdisplay labtesting

# Create logical volumes
lvcreate -L 1G -n data labtesting
lvcreate -L 500M -n logs labtesting
lvcreate -L 256M -n swap labtesting

# Verify LVM structure
lvdisplay
```

4. **Create filesystems and swap**

```
# Create filesystems
mkfs.xfs /dev/labtesting/data
mkfs.ext4 /dev/labtesting/logs
mkswap /dev/labtesting/swap

# Verify filesystem creation
blkid | grep labtesting
```

5. **Mount and configure**

```
# Create mount points
mkdir -p /lab/{data,logs}

# Mount filesystems
mount /dev/labtesting/data /lab/data
mount /dev/labtesting/logs /lab/logs
swapon /dev/labtesting/swap

# Verify mounts
df -h /lab/data /lab/logs
swapon --show
```

Verification:

```
# Complete verification
lsblk
pvs && vgs && lvs
df -h | grep lab
swapon --show | grep labtesting
```

Lab 6.2: LVM Extension and Management (Sander van Vugt Style)

Objective: Practice extending and managing existing LVM infrastructure

Prerequisites: Lab 6.1 completed, additional disk (/dev/sdc) available

Steps:

1. **Add storage to existing VG**

```
# Prepare new disk
fdisk /dev/sdc
# Create partition, set type to Linux LVM (8e)
partprobe /dev/sdc

# Add to LVM
pvcreate /dev/sdc1
vgextend labtesting /dev/sdc1

# Verify VG growth
vgs labtesting
vgdisplay labtesting
```

2. **Extend logical volumes**

```
# Extend data LV by 2GB
lvextend -L +2G /dev/labtesting/data

# Extend logs LV to use remaining space
lvextend -l +100%FREE /dev/labtesting/logs

# Verify LV sizes
lvs labtesting
```

3. **Resize filesystems**

```
# Resize XFS filesystem (data)
xfs_growfs /lab/data

# Resize ext4 filesystem (logs)
resize2fs /dev/labtesting/logs

# Verify filesystem sizes
df -h /lab/data /lab/logs
```

4. **Practice LV management operations**

```
# Create snapshot of data LV
lvcreate -L 500M -s -n data-snapshot /dev/labtesting/data

# Create some test data
echo "Test file content" > /lab/data/testfile.txt
ls -la /lab/data/

# Mount and examine snapshot
mkdir /mnt/snapshot
mount /dev/labtesting/data-snapshot /mnt/snapshot
ls -la /mnt/snapshot/

# Clean up snapshot
umount /mnt/snapshot
lvremove -f /dev/labtesting/data-snapshot
```

Verification:

```
# Verify final state
pvs && vgs && lvs
df -h | grep lab
lsblk | grep labtesting
```

Lab 6.3: Complete Storage Migration (Synthesis Challenge)

Objective: Migrate existing storage to new LVM configuration with minimal downtime

Scenario: Migrate a production-like data directory to new storage with better organization

Requirements:

- Create new VG with better naming convention
- Migrate data safely without corruption
- Update system configuration appropriately
- Document the migration process

Solution Steps:

1. **Prepare new storage infrastructure**

```
# Use remaining space or additional disk
if lsblk | grep -q sdd; then
    NEWDISK=/dev/sdd
else
    # Use remaining space in existing VG
    NEWDISK="extend_existing"
fi

if [ "$NEWDISK" != "extend_existing" ]; then
    pvcreate ${NEWDISK}1
    vgcreate production ${NEWDISK}1
else
    # Extend existing VG and create new LVs
    vgextend labtesting /dev/sdc1 2>/dev/null || true
fi

# Create production-style LVM layout
if vgs production >/dev/null 2>&1; then
    lvcreate -L 3G -n app-data production
    lvcreate -L 1G -n app-logs production
    lvcreate -L 500M -n app-backup production
else
    lvcreate -L 1G -n app-data labtesting
    lvcreate -L 500M -n app-logs labtesting
    lvcreate -L 256M -n app-backup labtesting
fi
```

2. **Prepare new filesystems**

```
# Determine which VG to use
if vgs production >/dev/null 2>&1; then
    VG=production
else
    VG=labtesting
fi

# Create filesystems with appropriate options
mkfs.xfs -L app-data /dev/${VG}/app-data
mkfs.ext4 -L app-logs /dev/${VG}/app-logs
mkfs.ext4 -L app-backup /dev/${VG}/app-backup

# Create mount points
mkdir -p /app/{data,logs,backup}
```

3. Migrate existing data

```
# Create some test data in old location
echo "Important application data" > /lab/data/app.conf
echo "$(date): Application started" > /lab/logs/app.log
mkdir -p /lab/data/important
echo "Critical data" > /lab/data/important/critical.txt

# Mount new filesystems temporarily
mkdir -p /mnt/migration/{data,logs,backup}
mount /dev/${VG}/app-data /mnt/migration/data
mount /dev/${VG}/app-logs /mnt/migration/logs
mount /dev/${VG}/app-backup /mnt/migration/backup

# Migrate data safely with rsync
rsync -avxHAX /lab/data/ /mnt/migration/data/
rsync -avxHAX /lab/logs/ /mnt/migration/logs/

# Create backup of migration
tar -czf /mnt/migration/backup/migration-backup-$(date +%Y%m%d).tar.gz \
    -C /mnt/migration data logs

# Verify data integrity
diff -r /lab/data /mnt/migration/data
echo "Data integrity check: $?"
```

4. Update system configuration

```
# Prepare new fstab entries
cat >> /tmp/new-fstab-entries << EOF
/dev/${VG}/app-data  /app/data    xfs   defaults,noatime     0 2
/dev/${VG}/app-logs  /app/logs    ext4  defaults,noatime     0 2
/dev/${VG}/app-backup /app/backup ext4  defaults,noexec      0 2
EOF

# Show what will be added
echo "New fstab entries:"
cat /tmp/new-fstab-entries

# Add to fstab (in real migration, this would be done during maintenance window)
cat /tmp/new-fstab-entries >> /etc/fstab

# Switch to new storage
umount /mnt/migration/data /mnt/migration/logs /mnt/migration/backup
mount /dev/${VG}/app-data /app/data
mount /dev/${VG}/app-logs /app/logs
mount /dev/${VG}/app-backup /app/backup
```

5. **Verify and document migration**

```
# Create migration report
cat > /app/backup/migration-report-$(date +%Y%m%d).txt << EOF
Storage Migration Report
Date: $(date)

Old Storage:
- /lab/data ($(du -sh /lab/data | cut -f1))
- /lab/logs ($(du -sh /lab/logs | cut -f1))

New Storage:
Volume Group: ${VG}
- /app/data: /dev/${VG}/app-data (XFS, $(df -h /app/data | tail -1 | awk '{print $2}'))
- /app/logs: /dev/${VG}/app-logs (ext4, $(df -h /app/logs | tail -1 | awk '{print $2}'))
- /app/backup: /dev/${VG}/app-backup (ext4, $(df -h /app/backup | tail -1 | awk '{print
$2}'))

Migration Status: COMPLETED
Data Verification: PASSED
Backup Created: migration-backup-$(date +%Y%m%d).tar.gz
EOF

# Display final configuration
echo "=== Migration Complete ==="
lsblk | grep -E "(${VG}|labtesting)"
df -h | grep app
cat /app/backup/migration-report-$(date +%Y%m%d).txt
```

Verification:

```
# Complete post-migration verification
mount | grep "/app"
ls -la /app/data/
ls -la /app/logs/
ls -la /app/backup/
cat /app/data/app.conf
cat /app/logs/app.log
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: LV Won't Mount After Reboot

Symptoms:

- Filesystem not available after system restart
- "Device not found" errors during boot
- Services fail to start due to missing storage

Diagnosis:

```
# Check if LVM volumes are active
lvs
vgs
pvs

# Check fstab entries
cat /etc/fstab | grep -v "^#"

# Check what's actually mounted
mount | grep mapper
```

Resolution:

```
# Activate volume groups
vgchange -ay

# Scan for LVM volumes
pvscan
vgscan
lvscan

# Fix fstab if needed (use UUID instead of device names)
blkid /dev/vgname/lvname
# Replace device path with UUID in fstab

# Test mount
mount -a
```

Prevention: Use UUIDs in fstab, ensure LVM service is enabled

Issue 2: Cannot Extend Filesystem

Symptoms:

- LV extends successfully but filesystem size unchanged
- "No space left on device" despite LV extension
- Applications still report old filesystem size

Diagnosis:

```
# Check LV vs filesystem size
lvs
df -h /mountpoint

# Check filesystem type
mount | grep /mountpoint
blkid /dev/vgname/lvname
```

Resolution:

```
# For XFS filesystems
xfs_growfs /mountpoint

# For ext4 filesystems
resize2fs /dev/vgname/lvname

# Verify filesystem size
df -h /mountpoint
```

Issue 3: VG Shows as Inactive

Symptoms:

- Volume group not visible or inactive
- Cannot access logical volumes
- LVM commands show no VGs

Diagnosis:

```
# Check PV status
pvdisplay -C
pvscan

# Check for VG metadata issues
vgscan
vgdisplay -A
```

Resolution:

```
# Activate volume group
vgchange -ay vgname

# If metadata corrupted, restore from backup
vgcfgrestore -l vgname
vgcfgrestore -f /etc/lvm/archive/vgname_XXXX.vg vgname

# Rescan LVM components
pvscan
vgscan
lvscan
```

Diagnostic Command Sequence

```
# Storage troubleshooting workflow
lsblk                        # Overview of block devices
df -h                        # Mounted filesystem status
mount | column -t            # Mount point details
pvs && vgs && lvs           # LVM component status
blkid                       # Device UUIDs and filesystems
cat /etc/fstab              # Persistent mount configuration
```

Log File Analysis

- `/var/log/messages`: General storage and LVM errors
- `/var/log/boot.log`: Boot-time storage issues
- `dmesg`: Kernel messages about storage devices
- `journalctl -u lvm2*`: LVM service messages

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# LVM creation workflow
pvcreate /dev/sdb1           # Create physical volume
vgcreate vgname /dev/sdb1    # Create volume group
lvcreate -L 2G -n lvname vgname  # Create logical volume
mkfs.xfs /dev/vgname/lvname  # Create filesystem
mount /dev/vgname/lvname /mnt # Mount filesystem

# Extension workflow
vgextend vgname /dev/sdc1    # Add space to VG
lvextend -L +1G /dev/vgname/lvname  # Extend LV
xfs_growfs /mnt              # Grow XFS filesystem

# Thin provisioning workflow
lvcreate --type thin-pool -L 5G -n pool vgname  # Create thin pool
lvcreate --virtualsize 10G --thin -n lv1 vgname/pool  # Create thin LV
lvs -o+data_percent vgname/pool  # Monitor pool usage
```

fstab Entry Examples

```
# Using device path (not recommended)
/dev/datavg/datalv  /data  xfs  defaults  0  2

# Using UUID (recommended)
UUID=abc123-def456  /data  xfs  defaults,noatime  0  2

# Swap entry
/dev/datavg/swaplv  swap  swap  defaults  0  0
```

LVM Size Specifications

- **Absolute sizes**: `1G`, `500M`, `2T`
- **Percentage of VG**: `50%VG`, `100%VG`
- **Percentage of free space**: `50%FREE`, `100%FREE`
- **Relative changes**: `+1G`, `-500M`, `+50%FREE`

Filesystem Types

- **XFS**: Best for large files, can only grow
- **ext4**: General purpose, can grow and shrink
- **swap**: Virtual memory extension

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the advantage of LVM over traditional partitioning? **Answer**: LVM provides flexibility to resize storage without repartitioning disks. You can extend logical volumes across multiple physical devices, create snapshots, and resize filesystems online. It separates physical storage from logical organization, making storage management much more flexible.
2. **Question**: Why can't you shrink an XFS filesystem? **Answer**: XFS is designed for performance and large-scale storage. Its metadata structure and allocation algorithms are optimized for forward growth. Shrinking would require complex metadata reorganization that could compromise performance and reliability, so XFS developers chose to support only growth operations.
3. **Question**: When would you use a swap file instead of swap partition? **Answer**: Swap files are easier to manage dynamically - you can create, resize, or remove them without repartitioning. They're useful when you need to add swap temporarily, when using cloud instances with limited partitioning options, or when you want to adjust swap size based on changing workload requirements.

Practical Scenarios

1. **Scenario**: Database server running out of space for transaction logs. **Solution**: Extend the LV containing the logs (`lvextend -L +10G /dev/vgname/logslv`), then grow the filesystem (`xfs_growfs /var/lib/mysql/logs`), assuming XFS filesystem.
2. **Scenario**: Need to migrate data from failing disk to new storage. **Solution**: Create new LVM structure on new disk, use `rsync -avxHAX` to copy data while old disk still works, then update fstab and remount on new storage.

Command Challenges

1. **Challenge**: Create a 5GB logical volume using exactly 50% of volume group space. **Answer**: `lvcreate -l 50%VG -n mylv vgname` **Explanation**: `-l` uses extents/percentages, `50%VG` means half of total VG space
2. **Challenge**: Find all LVM logical volumes and their mount points. **Answer**: `findmnt | grep mapper` or `mount | grep mapper` **Explanation**: LVM device paths contain `/dev/mapper/` prefix when mounted

10. Exam Strategy

Topic-Specific Tips

- Always verify disk space before creating partitions or LVs
- Use `lsblk` to visualize storage hierarchy before making changes
- Remember that XFS can only grow, not shrink
- Practice the complete workflow: PV → VG → LV → filesystem → mount → fstab
- For thin provisioning: monitor pool usage (`lvs -o+data_percent`) and extend before full

Common Exam Scenarios

1. **Scenario**: Add storage to existing system **Approach**: Create PV, extend VG, extend LV, resize filesystem
2. **Scenario**: Create swap space **Approach**: Create LV, format as swap, enable swap, add to fstab
3. **Scenario**: Set up persistent mounts **Approach**: Add appropriate entries to /etc/fstab, test with `mount -a`

Time Management

- **Basic LVM setup**: 8-10 minutes for complete PV→VG→LV→filesystem→mount
- **Storage extension**: 5-7 minutes for extend LV and resize filesystem
- **Partition creation**: 3-4 minutes using fdisk
- **Always verify**: Check with `df -h` and `mount` after each step

Pitfalls to Avoid

- Don't forget `partprobe` after creating partitions with fdisk
- Remember to resize filesystem after extending LV
- Use UUIDs in fstab for reliability
- Always verify fstab with `mount -a` before rebooting
- Don't try to shrink XFS - it's not supported
- Ensure adequate space before creating LVs (don't use 100% VG space)

Summary

Key Takeaways

- **LVM provides storage flexibility** - essential for production systems

- **Understand the hierarchy**: Physical Volume → Volume Group → Logical Volume → Filesystem
- **XFS vs ext4 have different capabilities** - choose based on requirements
- **fstab configuration is critical** - systems won't boot properly without correct entries

Critical Commands to Remember

```
pvcreate /dev/sdb1                       # Create physical volume
vgcreate vgname /dev/sdb1                # Create volume group
lvcreate -L 2G -n lvname vgname          # Create logical volume
mkfs.xfs /dev/vgname/lvname              # Create XFS filesystem
lvextend -L +1G /dev/vgname/lvname       # Extend logical volume
xfs_growfs /mountpoint                   # Grow XFS filesystem
```

Next Steps

- Continue to Module 08: Network Configuration
- Practice storage management in the Vagrant environment with multiple disks
- Review related topics: System Installation, Boot Process

Navigation: ← Package Management | Index | Next → Network Configuration

2.10 08 - Network Configuration

Navigation: ← Storage & LVM | Index | Next → SELinux Management

1. Executive Summary

Topic Scope: Network interface configuration, static IP assignment, hostname management, and network troubleshooting in RHEL 10

RHCSA Relevance: Essential system administration skill - network configuration is fundamental for server management

Exam Weight: Medium-High - Network tasks appear frequently in exam scenarios

Prerequisites: Basic understanding of TCP/IP networking and Linux command line operations

Related Topics: System Installation, SSH Configuration, Firewall

2. Conceptual Foundation

Core Theory

RHEL 10 uses NetworkManager as the primary network management service:

- **NetworkManager**: Modern network configuration service replacing traditional networking scripts
- **Connection profiles**: Persistent network configurations stored by NetworkManager
- **Device vs Connection**: Devices are physical/virtual interfaces; connections are configurations applied to devices
- **Static vs DHCP**: Manual IP assignment versus automatic configuration
- **Network namespaces**: Isolated network environments (advanced topic)

Real-World Applications

- **Server deployment**: Configuring static IPs for production servers
- **Network troubleshooting**: Diagnosing connectivity issues in enterprise environments
- **Remote management**: Ensuring SSH access through proper network configuration
- **Service hosting**: Setting up networks for web servers, databases, and applications
- **Network isolation**: Separating different types of traffic for security

Common Misconceptions

- **NetworkManager vs network scripts**: RHEL 10 uses NetworkManager, not legacy scripts
- **Interface naming**: Modern systems use predictable names (ens3, enp0s3) not eth0
- **Connection vs device state**: A device can be up but connection down
- **DNS configuration**: Managed by NetworkManager, not directly in /etc/resolv.conf
- **Gateway terminology**: Default route and default gateway are the same concept

Key Terminology

- **Interface**: Physical or virtual network device (ens3, enp0s3, virbr0)
- **Connection**: NetworkManager configuration profile applied to interface
- **Profile**: Persistent network configuration including IP, DNS, gateway
- **DHCP**: Dynamic Host Configuration Protocol for automatic IP assignment
- **Static IP**: Manually configured IP address that doesn't change
- **Gateway**: Router that connects local network to other networks
- **DNS**: Domain Name System for resolving hostnames to IP addresses
- **MTU**: Maximum Transmission Unit - largest packet size for interface

---

3. Command Mastery

NetworkManager Commands (nmcli)

```
# Connection management
nmcli connection show                    # List all connections
nmcli connection show --active           # Show active connections
nmcli connection show "connection-name"  # Show specific connection details
nmcli connection up "connection-name"    # Activate connection
nmcli connection down "connection-name"  # Deactivate connection
nmcli connection delete "connection-name" # Delete connection
nmcli connection reload                  # Reload connection files

# Device information
nmcli device status                      # Show device status
nmcli device show                        # Show all device details
nmcli device show ens3                   # Show specific device details
nmcli device connect ens3                # Connect device to auto connection
nmcli device disconnect ens3             # Disconnect device

# Creating connections
nmcli connection add type ethernet con-name "static-ens3" ifname ens3 \
  ipv4.addresses 192.168.1.100/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8 8.8.4.4" \
  ipv4.method manual

nmcli connection add type ethernet con-name "dhcp-ens3" ifname ens3 \
  ipv4.method auto

# Modifying connections
nmcli connection modify "static-ens3" ipv4.addresses 192.168.1.150/24
nmcli connection modify "static-ens3" ipv4.dns "8.8.8.8,1.1.1.1"
nmcli connection modify "static-ens3" connection.autoconnect yes
```

Traditional Network Commands

```
# Interface information
ip addr show                    # Show all interface addresses
ip addr show ens3               # Show specific interface
ip link show                    # Show link layer information
ip route show                   # Show routing table
ip route show default           # Show default route only

# Legacy commands (still useful for information)
ifconfig                        # Show interface configuration (if installed)
route -n                        # Show routing table (if installed)
netstat -rn                     # Show routing table
ss -tuln                        # Show listening sockets (replaces netstat)
```

Network Testing Commands

```
# Connectivity testing
ping -c 4 8.8.8.8              # Test internet connectivity
ping -c 4 192.168.1.1          # Test gateway connectivity
traceroute google.com          # Trace route to destination
tracepath google.com           # Alternative to traceroute

# DNS testing
nslookup google.com            # Basic DNS lookup
dig google.com                 # Detailed DNS query
host google.com                # Simple DNS lookup

# Port and service testing
telnet host 80                 # Test TCP port connectivity
nc -zv host 80                 # Test port with netcat
ss -tuln | grep :80            # Check if service listening on port
```

Hostname Management

```
# Hostname commands
hostnamectl                     # Show hostname information
hostnamectl set-hostname server1.example.com  # Set hostname
hostnamectl status             # Detailed hostname status

# Legacy hostname commands
hostname                       # Show current hostname
hostname server1               # Set hostname (temporary)
```

Network Configuration Files

```
# NetworkManager connection files
ls /etc/NetworkManager/system-connections/  # Connection profiles
cat /etc/NetworkManager/NetworkManager.conf # NetworkManager config

# System network files
cat /etc/hosts                 # Local hostname resolution
cat /etc/resolv.conf           # DNS configuration (managed by NetworkManager)
cat /etc/nsswitch.conf         # Name resolution order
```

Command Reference Table

Command	Purpose	Key Options	Example
`nmcli con show`	List connections	`--active`	`nmcli con show --active`
`nmcli con add`	Create connection	`type` , `con-name` , `ifname`	`nmcli con add type ethernet con-name static ifname ens3`
`nmcli con modify`	Modify connection	`ipv4.addresses` , `ipv4.method`	`nmcli con modify static ipv4.addresses 192.168.1.100/24`
`ip addr show`	Show IP addresses	`dev`	`ip addr show dev ens3`
`ping`	Test connectivity	`-c`	`ping -c 4 8.8.8.8`
`hostnamectl`	Manage hostname	`set-hostname`	`hostnamectl set-hostname server1`

---

4. Procedural Workflows

Standard Procedure: Configure Static IP Address

1. **Identify available interface**

```
nmcli device status
ip link show
```

2. **Create static connection**

```
nmcli connection add type ethernet \
  con-name "static-connection" \
  ifname ens3 \
  ipv4.addresses 192.168.1.100/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8 8.8.4.4" \
  ipv4.method manual \
  connection.autoconnect yes
```

3. **Activate connection**

```
nmcli connection up "static-connection"
```

4. **Verify configuration**

```
ip addr show ens3
ip route show
cat /etc/resolv.conf
ping -c 2 8.8.8.8
```

Standard Procedure: Switch from DHCP to Static

1. **Check current configuration**

```
nmcli connection show --active
nmcli device show ens3
```

2. **Modify existing connection**

```
# Get current connection name
CON_NAME=$(nmcli -t -f NAME connection show --active | head -1)

# Modify to static
nmcli connection modify "$CON_NAME" \
  ipv4.method manual \
  ipv4.addresses 192.168.1.100/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8,8.8.4.4"
```

3. **Apply changes**

```
nmcli connection down "$CON_NAME"
nmcli connection up "$CON_NAME"
```

4. **Verify changes**

```
ip addr show
ping -c 2 google.com
```

Decision Tree: Network Configuration Strategy

```
Network Configuration Need
├── New system setup?
│   ├── DHCP available? → Use auto method
│   └── Static required? → Configure manual method
├── Change existing config?
│   ├── Modify current connection → nmcli con modify
│   └── Create new connection → nmcli con add
├── Troubleshooting?
│   ├── Check device status → nmcli device status
│   ├── Check connection status → nmcli con show --active
│   └── Test connectivity → ping, traceroute
└── Advanced needs?
    ├── Multiple IPs? → Add secondary addresses
    ├── VLANs? → Create VLAN connections
    └── Bonding? → Create bond connections
```

Standard Procedure: Network Troubleshooting

1. **Check physical connectivity**

```
nmcli device status
ip link show
# Look for "connected" state and "UP" flags
```

2. **Check IP configuration**

```
ip addr show
nmcli connection show --active
```

3. **Test network layers**

```
# Layer 3 - IP connectivity
ping -c 2 127.0.0.1        # Loopback
ping -c 2 $(ip route | grep default | awk '{print $3}')  # Gateway
ping -c 2 8.8.8.8          # External IP

# Layer 7 - DNS resolution
nslookup google.com
ping -c 2 google.com
```

4. **Check services and ports**

```
ss -tuln                    # Listening services
systemctl status NetworkManager
```

---

5. Configuration Deep Dive

NetworkManager Connection Files

Static IP Connection

```
# /etc/NetworkManager/system-connections/static-ens3.nmconnection
[connection]
id=static-ens3
uuid=12345678-1234-1234-1234-123456789012
type=ethernet
interface-name=ens3
autoconnect=true

[ethernet]

[ipv4]
address1=192.168.1.100/24,192.168.1.1
dns=8.8.8.8;8.8.4.4;
method=manual

[ipv6]
addr-gen-mode=eui64
method=auto

[proxy]
```

DHCP Connection

```
# /etc/NetworkManager/system-connections/dhcp-ens3.nmconnection
[connection]
id=dhcp-ens3
uuid=87654321-4321-4321-4321-210987654321
type=ethernet
interface-name=ens3
autoconnect=true

[ethernet]

[ipv4]
method=auto

[ipv6]
addr-gen-mode=eui64
method=auto

[proxy]
```

Advanced Network Configuration

Multiple IP Addresses

```
# Add secondary IP to existing connection
nmcli connection modify "static-ens3" \
  +ipv4.addresses 192.168.1.101/24

# Or specify multiple IPs during creation
nmcli connection add type ethernet con-name "multi-ip" ifname ens3 \
  ipv4.addresses "192.168.1.100/24,192.168.1.101/24" \
  ipv4.gateway 192.168.1.1 \
  ipv4.method manual
```

DNS Configuration

```
# Set specific DNS servers
nmcli connection modify "connection-name" \
  ipv4.dns "8.8.8.8,8.8.4.4,1.1.1.1"

# Set DNS search domains
nmcli connection modify "connection-name" \
  ipv4.dns-search "example.com,lab.local"

# Ignore auto DNS (use manual only)
nmcli connection modify "connection-name" \
  ipv4.ignore-auto-dns yes
```

Network Interface Naming

Predictable Network Interface Names

```
# Modern naming scheme (RHEL 10):
# ens3    - Ethernet slot 3
# enp0s3  - Ethernet PCI bus 0, slot 3
# enp1s0f0 - Ethernet PCI bus 1, slot 0, function 0
# wls1    - Wireless LAN slot 1

# View interface naming details
udevadm info /sys/class/net/ens3
```

---

6. Hands-On Labs

Lab 6.1: Basic Network Configuration (Asghar Ghori Style)

Objective: Configure static IP address and test connectivity

Steps:

1. **Explore current network configuration**

```
# Check current interfaces and connections
nmcli device status
nmcli connection show

# Show detailed interface information
ip addr show
ip route show

# Check current connectivity
ping -c 2 8.8.8.8
```

2. **Create static IP configuration**

```
# Create new static connection (adjust IP range for your environment)
nmcli connection add type ethernet \
  con-name "lab-static" \
  ifname ens3 \
  ipv4.addresses 192.168.1.150/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8 8.8.4.4" \
  ipv4.method manual \
  connection.autoconnect yes

# Activate the new connection
nmcli connection up "lab-static"
```

3. **Verify static configuration**

```
# Check new IP configuration
ip addr show ens3
ip route show
cat /etc/resolv.conf

# Test connectivity
ping -c 3 192.168.1.1        # Gateway
ping -c 3 8.8.8.8            # External IP
ping -c 3 google.com         # DNS resolution
```

4. **Test connection management**

```
# Bring connection down and up
nmcli connection down "lab-static"
ip addr show ens3            # Should show no IP

nmcli connection up "lab-static"
ip addr show ens3            # Should show static IP

# Show connection details
nmcli connection show "lab-static"
```

Verification:

```
# Complete verification
nmcli connection show --active | grep lab-static
ping -c 2 google.com
nslookup google.com
ip route show | grep default
```

Lab 6.2: Advanced Network Management (Sander van Vugt Style)

Objective: Modify existing connections and manage multiple network configurations

Steps:

1. **Analyze existing network setup**

```
# Document current configuration
nmcli connection show > /tmp/original-connections.txt
nmcli device show > /tmp/original-devices.txt
ip addr show > /tmp/original-ips.txt
```

2. **Create DHCP backup connection**

```
# Create DHCP connection for same interface
nmcli connection add type ethernet \
  con-name "lab-dhcp-backup" \
  ifname ens3 \
  ipv4.method auto \
  connection.autoconnect no

# Test switching between static and DHCP
nmcli connection down "lab-static"
nmcli connection up "lab-dhcp-backup"

# Check what IP was assigned by DHCP
ip addr show ens3
```

3. **Modify connection properties**

```
# Switch back to static and modify it
nmcli connection down "lab-dhcp-backup"
nmcli connection up "lab-static"

# Add secondary IP address
nmcli connection modify "lab-static" \
  +ipv4.addresses 192.168.1.151/24

# Change DNS servers
nmcli connection modify "lab-static" \
  ipv4.dns "1.1.1.1,8.8.8.8"

# Apply changes
nmcli connection down "lab-static"
nmcli connection up "lab-static"
```

4. **Verify multiple IPs and DNS changes**

```
# Check multiple IP addresses
ip addr show ens3 | grep inet

# Verify DNS changes
cat /etc/resolv.conf

# Test connectivity from both IPs
ping -c 2 -I 192.168.1.150 google.com
ping -c 2 -I 192.168.1.151 google.com
```

Verification:

```
# Document final configuration
nmcli connection show "lab-static"
ip addr show ens3
nslookup google.com
```

Lab 6.3: Network Troubleshooting Scenario (Synthesis Challenge)

Objective: Diagnose and resolve complex network connectivity issues

Scenario: A server has lost network connectivity and needs systematic troubleshooting

Requirements:

- Systematically diagnose network issues
- Test connectivity at multiple network layers
- Document findings and resolution steps
- Restore full network functionality

Solution Steps:

1. **Create a "broken" network scenario**

```
# Simulate network problems (choose one or more):

# Problem 1: Wrong gateway
nmcli connection modify "lab-static" ipv4.gateway 192.168.1.99

# Problem 2: Wrong DNS
nmcli connection modify "lab-static" ipv4.dns "192.168.1.99"

# Problem 3: Wrong IP range
nmcli connection modify "lab-static" ipv4.addresses 10.0.0.100/24

# Apply one of these problematic configs
nmcli connection down "lab-static"
nmcli connection up "lab-static"

# Verify the problem exists
ping -c 2 google.com  # This should fail
```

2. Systematic network troubleshooting

```
# Step 1: Check physical and link layer
echo "=== LAYER 1 & 2 DIAGNOSTICS ==="
nmcli device status
ip link show ens3

# Step 2: Check network layer (IP configuration)
echo "=== LAYER 3 DIAGNOSTICS ==="
ip addr show ens3
ip route show

# Step 3: Test connectivity at each level
echo "=== CONNECTIVITY TESTS ==="

# Test loopback
ping -c 1 127.0.0.1 && echo "Loopback: OK" || echo "Loopback: FAIL"

# Test local IP
LOCAL_IP=$(ip addr show ens3 | grep 'inet ' | awk '{print $2}' | cut -d'/' -f1)
ping -c 1 $LOCAL_IP && echo "Local IP: OK" || echo "Local IP: FAIL"

# Test gateway
GATEWAY=$(ip route show default | awk '{print $3}')
ping -c 1 $GATEWAY && echo "Gateway: OK" || echo "Gateway: FAIL"

# Test external IP
ping -c 1 8.8.8.8 && echo "External IP: OK" || echo "External IP: FAIL"

# Test DNS resolution
nslookup google.com > /dev/null 2>&1 && echo "DNS Resolution: OK" || echo "DNS Resolution:
FAIL"
```

3. **Diagnose and fix the specific problem**

```
# Based on the test results, fix the issue:

# If gateway test failed:
echo "Fixing gateway configuration..."
nmcli connection modify "lab-static" ipv4.gateway 192.168.1.1

# If DNS resolution failed but external IP worked:
echo "Fixing DNS configuration..."
nmcli connection modify "lab-static" ipv4.dns "8.8.8.8,8.8.4.4"

# If external IP failed but gateway worked (wrong IP range):
echo "Fixing IP address configuration..."
nmcli connection modify "lab-static" ipv4.addresses 192.168.1.150/24

# Apply the fix
nmcli connection down "lab-static"
nmcli connection up "lab-static"
```

4. Verify resolution and document

```
# Re-run connectivity tests
echo "=== POST-FIX VERIFICATION ==="
ping -c 2 127.0.0.1      # Loopback
ping -c 2 192.168.1.1    # Gateway
ping -c 2 8.8.8.8        # External IP
ping -c 2 google.com     # DNS resolution

# Create troubleshooting report
cat > /tmp/network-troubleshooting-report.txt << EOF
Network Troubleshooting Report
Date: $(date)

Problem Description:
- Network connectivity was lost
- Systematic diagnostics performed

Diagnostics Performed:
1. Physical layer check: nmcli device status
2. IP configuration check: ip addr show, ip route show
3. Connectivity tests: loopback, gateway, external IP, DNS

Issue Found:
$(if ping -c 1 google.com > /dev/null 2>&1; then echo "Issue resolved successfully"; else
echo "Issue still present"; fi)

Resolution Applied:
- Modified connection: lab-static
- Updated configuration parameters
- Reactivated network connection

Final Configuration:
$(nmcli connection show "lab-static" | grep ipv4)

Post-Fix Tests:
- Gateway connectivity: $(ping -c 1 192.168.1.1 > /dev/null 2>&1 && echo "PASS" || echo
"FAIL")
- External connectivity: $(ping -c 1 8.8.8.8 > /dev/null 2>&1 && echo "PASS" || echo "FAIL")
- DNS resolution: $(ping -c 1 google.com > /dev/null 2>&1 && echo "PASS" || echo "FAIL")
EOF

# Display the report
cat /tmp/network-troubleshooting-report.txt
```

Verification:

```
# Final comprehensive verification
echo "=== FINAL NETWORK STATUS ==="
nmcli connection show --active
ip addr show ens3
ip route show | grep default
ping -c 2 google.com
nslookup google.com
cat /tmp/network-troubleshooting-report.txt
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: No Network Connectivity After Configuration

Symptoms:

- Cannot ping gateway or external hosts
- New IP configuration not applied
- Connection shows as activated but no connectivity

Diagnosis:

```
# Check connection and device status
nmcli connection show --active
nmcli device status

# Verify IP configuration applied
ip addr show interface-name
ip route show

# Check NetworkManager logs
journalctl -u NetworkManager --since "10 minutes ago"
```

Resolution:

```
# Restart NetworkManager service
systemctl restart NetworkManager

# Reload connection configuration
nmcli connection reload

# Manually bring connection down and up
nmcli connection down "connection-name"
nmcli connection up "connection-name"

# Check for conflicting connections
nmcli connection show | grep interface-name
```

Prevention: Always verify configuration before applying, test in stages

Issue 2: DNS Resolution Not Working

Symptoms:

- Can ping IP addresses but not hostnames
- /etc/resolv.conf has wrong or no DNS servers
- nslookup/dig commands fail

Diagnosis:

```
# Check DNS configuration
cat /etc/resolv.conf
nmcli connection show active-connection | grep dns

# Test DNS directly
nslookup google.com 8.8.8.8
dig @8.8.8.8 google.com

# Check DNS service
systemctl status systemd-resolved  # If using resolved
```

Resolution:

```
# Fix DNS in connection configuration
nmcli connection modify "connection-name" \
  ipv4.dns "8.8.8.8,8.8.4.4"

# Ignore auto-assigned DNS if needed
nmcli connection modify "connection-name" \
  ipv4.ignore-auto-dns yes

# Restart connection
nmcli connection down "connection-name"
nmcli connection up "connection-name"
```

Issue 3: Interface Name Changes After Reboot

Symptoms:

- Network interface has different name after reboot
- Connection tied to specific interface fails
- Previous interface name no longer exists

Diagnosis:

```
# Check current interfaces
nmcli device status
ip link show

# Check for renamed interfaces
dmesg | grep -i "renamed network interface"
journalctl -b | grep "renamed network interface"
```

Resolution:

```
# Update connection to use correct interface
nmcli connection modify "connection-name" \
  connection.interface-name new-interface-name

# Or create new connection with correct interface
nmcli connection clone "old-connection" "new-connection"
nmcli connection modify "new-connection" \
  connection.interface-name new-interface-name
```

Diagnostic Command Sequence

```
# Network troubleshooting workflow
nmcli device status              # Check device status
nmcli connection show --active   # Check active connections
ip addr show                     # Check IP assignments
ip route show                    # Check routing table
ping -c 2 gateway-ip            # Test gateway connectivity
ping -c 2 8.8.8.8               # Test external connectivity
nslookup google.com              # Test DNS resolution
```

Log File Analysis

- `journalctl -u NetworkManager`: NetworkManager service logs
- `/var/log/messages`: General system messages including network events
- `dmesg`: Kernel messages about network interfaces
- `journalctl -f`: Real-time log monitoring during troubleshooting

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Connection management
nmcli con show                    # List connections
nmcli con up "name"              # Activate connection
nmcli con down "name"            # Deactivate connection

# Static IP setup
nmcli con add type ethernet con-name "static" ifname ens3 \
  ipv4.addresses 192.168.1.100/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8" \
  ipv4.method manual

# Network information
ip addr show                     # Show IP addresses
ip route show                    # Show routing table
ping -c 2 host                   # Test connectivity
```

NetworkManager Configuration Structure

```
# Connection properties:
connection.id                    # Connection name
connection.interface-name        # Interface to use
connection.autoconnect          # Auto-connect on boot

# IPv4 properties:
ipv4.method                      # auto (DHCP) or manual (static)
ipv4.addresses                   # IP address/netmask
ipv4.gateway                     # Default gateway
ipv4.dns                         # DNS servers
```

Common Network Ranges

- **Private Class A**: 10.0.0.0/8 (10.0.0.0 - 10.255.255.255)
- **Private Class B**: 172.16.0.0/12 (172.16.0.0 - 172.31.255.255)
- **Private Class C**: 192.168.0.0/16 (192.168.0.0 - 192.168.255.255)
- **Loopback**: 127.0.0.0/8 (127.0.0.1 is localhost)

DNS Servers

- **Google**: 8.8.8.8, 8.8.4.4
- **Cloudflare**: 1.1.1.1, 1.0.0.1
- **Quad9**: 9.9.9.9, 149.112.112.112

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between a network device and a network connection in NetworkManager? **Answer**: A device is a physical or virtual network interface (like ens3), while a connection is a configuration profile that can be applied to a device. One device can have multiple connection profiles, but only one can be active at a time.
2. **Question**: Why might /etc/resolv.conf show different DNS servers than what you configured? **Answer**: NetworkManager dynamically manages /etc/resolv.conf. If you have multiple connections or DHCP is providing DNS servers, NetworkManager combines them. Use `nmcli connection show` to see the actual DNS configuration for each connection.

3. **Question**: What happens when you set ipv4.method to "auto" versus "manual"? **Answer**: "auto" uses DHCP to automatically obtain IP address, gateway, and DNS servers from a DHCP server. "manual" requires you to explicitly specify all network parameters and creates a static configuration.

Practical Scenarios

1. **Scenario**: Server needs to be accessible from multiple subnets but only has one interface. **Solution**: Add multiple IP addresses to the same connection:

```
nmcli con modify "connection" ipv4.addresses "192.168.1.100/24,10.0.1.100/24"
```

2. **Scenario**: Need to quickly switch between office and home network configurations. **Solution**: Create two connection profiles for the same interface and switch between them:

```
nmcli con add type ethernet con-name "office" ifname ens3 ipv4.method manual ...
nmcli con add type ethernet con-name "home" ifname ens3 ipv4.method auto
# Switch: nmcli con up "office" or nmcli con up "home"
```

Command Challenges

1. **Challenge**: Create a connection that gets IP via DHCP but uses custom DNS servers. **Answer**:

```
nmcli con add type ethernet con-name "dhcp-custom-dns" ifname ens3 \
  ipv4.method auto \
  ipv4.dns "8.8.8.8,8.8.4.4" \
  ipv4.ignore-auto-dns yes
```

2. **Challenge**: Find all interfaces that are up but don't have an IP address. **Answer**:
`ip link show | grep "state UP" -A1 | grep -B1 "NO-CARRIER\|state UP" | grep "^[0-9]" | cut -d: -f2`

---

10. Exam Strategy

Topic-Specific Tips

- Always use `nmcli` for configuration - it's the modern RHEL 10 way
- Verify configuration with both `nmcli` and `ip` commands
- Remember that connections must be activated after creation
- Test connectivity at multiple levels (gateway, external, DNS)

Common Exam Scenarios

1. **Scenario**: Configure static IP address on server **Approach**: Use `nmcli con add` with manual method, specify all required parameters
2. **Scenario**: Fix server that lost network connectivity **Approach**: Check device status, connection status, IP configuration, test connectivity systematically
3. **Scenario**: Change hostname of server **Approach**: Use `hostnamectl set-hostname` and verify with `hostnamectl status`

Time Management

- **Static IP configuration**: 5-7 minutes including verification
- **Network troubleshooting**: 8-10 minutes for systematic diagnosis
- **Hostname changes**: 2-3 minutes
- **Always verify**: Test connectivity after any network changes

Pitfalls to Avoid

- Don't forget to activate connections after creating them
- Remember that interface names may not be eth0 (use `nmcli device status` to find correct names)
- Always test both IP connectivity and DNS resolution
- Don't modify /etc/resolv.conf directly - use NetworkManager
- Verify changes persist after reboot if required

Summary

Key Takeaways

- **NetworkManager is the standard** in RHEL 10 - master `nmcli` commands
- **Connections are profiles** applied to devices - understand this relationship
- **Systematic troubleshooting** saves time - test connectivity at each network layer
- **Always verify configuration** with multiple commands and connectivity tests

Critical Commands to Remember

```
nmcli con add type ethernet con-name "static" ifname ens3 \
  ipv4.addresses 192.168.1.100/24 \
  ipv4.gateway 192.168.1.1 \
  ipv4.dns "8.8.8.8" \
  ipv4.method manual              # Create static IP connection

nmcli con show --active           # Show active connections
ip addr show                      # Show interface IP addresses
ping -c 2 gateway-ip             # Test connectivity
hostnamectl set-hostname name     # Set system hostname
```

Next Steps

- Continue to Module 09: SELinux Management
- Practice network configuration in the Vagrant environment
- Review related topics: Firewall Configuration, SSH Setup

Navigation: ← Storage & LVM | Index | Next → SELinux Management

2.11 09 - SELinux Management

Navigation: ← Network Configuration | Index | Next → Firewall Configuration

1. Executive Summary

Topic Scope: SELinux configuration, contexts, booleans, troubleshooting, and security policy management in RHEL 10

RHCSA Relevance: Critical security topic - SELinux is a major component of RHEL security and frequently tested

Exam Weight: High - SELinux tasks are complex and carry significant point values on the exam

Prerequisites: Understanding of Linux file permissions, processes, and network services

Related Topics: File Permissions, Process Management, Firewall

2. Conceptual Foundation

Core Theory

SELinux (Security-Enhanced Linux) is a Mandatory Access Control (MAC) system that provides an additional layer of security beyond traditional discretionary access controls:

- **Mandatory Access Control**: System enforces security policies that users cannot override
- **Security contexts**: Every process and file has a security context (labels)
- **Policy enforcement**: Rules determine what processes can access what resources
- **Multi-level security**: Supports classification levels and compartmentalization
- **Least privilege**: Processes run with minimal required permissions

Real-World Applications

- **Web server security**: Preventing web applications from accessing system files
- **Database protection**: Isolating database processes from unauthorized file access
- **Container security**: Providing isolation between containers and host system
- **Compliance**: Meeting security requirements for government and enterprise environments
- **Threat mitigation**: Limiting damage from compromised applications

Common Misconceptions

- **SELinux breaks things**: Properly configured SELinux enhances security without breaking functionality
- **Just disable it**: Disabling SELinux removes a critical security layer
- **Too complex**: Understanding basic concepts makes SELinux manageable
- **Performance impact**: Modern SELinux has minimal performance overhead
- **All or nothing**: SELinux can be selectively configured for different services

Key Terminology

- **Security Context**: Label format (user:role:type:level) assigned to processes and files
- **Type Enforcement**: Primary SELinux access control mechanism using type labels
- **Domain**: Security context type assigned to processes
- **Type**: Security context label for files and directories
- **Boolean**: Runtime switch to enable/disable specific SELinux policy rules
- **Policy**: Set of rules defining allowed interactions between domains and types
- **AVC (Access Vector Cache)**: SELinux access decisions and denials
- **Confined/Unconfined**: Whether a process is subject to SELinux restrictions

---

3. Command Mastery

SELinux Status and Mode Commands

```
# Check SELinux status
getenforce                    # Show current enforcement mode
sestatus                      # Detailed SELinux status
sestatus -v                   # Verbose status with policy details

# Change SELinux mode
setenforce 0                  # Set to permissive (temporary)
setenforce 1                  # Set to enforcing (temporary)
setenforce Enforcing          # Alternative syntax
setenforce Permissive         # Alternative syntax

# Permanent mode changes (requires reboot)
# Edit /etc/selinux/config:
# SELINUX=enforcing|permissive|disabled
```

Context Management Commands

```
# View file contexts
ls -Z file                                  # Show file security context
ls -lZ /path/                               # Show directory contents with contexts
stat filename                               # Includes SELinux context information

# View process contexts
ps -eZ                                      # All processes with contexts
ps -eZ | grep processname                   # Specific process contexts
ps auxZ                                     # Alternative format with contexts

# Change file contexts
chcon -t httpd_exec_t /path/file            # Change type only
chcon -R -t httpd_config_t /path/dir/       # Recursive type change
chcon --reference=/etc/passwd /path/file    # Copy context from reference

# Restore default contexts
restorecon /path/file                       # Restore single file
restorecon -R /path/dir/                    # Restore directory recursively
restorecon -Rv /path/dir/                   # Verbose recursive restore
```

Context Policy Management

```
# View context policies
semanage fcontext -l            # List all file context policies
semanage fcontext -l | grep httpd  # Show HTTP-related contexts

# Add new context policies
semanage fcontext -a -t httpd_exec_t "/opt/webapp/bin/.*"
semanage fcontext -a -t httpd_config_t "/opt/webapp/conf(/.*)?"

# Modify existing context policies
semanage fcontext -m -t httpd_log_t "/var/log/webapp/.*"

# Delete context policies
semanage fcontext -d "/opt/webapp/bin/.*"

# Apply context policies to files
semanage fcontext -a -t httpd_exec_t "/opt/webapp/bin/.*"
restorecon -Rv /opt/webapp/bin/
```

Boolean Management

```
# List SELinux booleans
getsebool -a                                # All booleans
getsebool -a | grep httpd                   # HTTP-related booleans
getsebool boolean_name                      # Specific boolean status

# Set booleans (temporary)
setsebool httpd_can_network_connect on
setsebool httpd_enable_homedirs off

# Set booleans (persistent)
setsebool -P httpd_can_network_connect on
setsebool -P httpd_enable_homedirs off

# Common useful booleans
setsebool -P httpd_can_network_connect on      # HTTP proxy/external connections
setsebool -P httpd_can_network_relay on        # HTTP as reverse proxy
setsebool -P httpd_enable_homedirs on          # Access user home directories
setsebool -P ftpd_anon_write on                # Anonymous FTP uploads
setsebool -P samba_enable_home_dirs on         # Samba home directory access
```

Troubleshooting Commands

```
# View SELinux denials
ausearch -m AVC -ts recent                     # Recent access vector cache denials
ausearch -m AVC -ts today                      # Today's denials
ausearch -m AVC -c httpd                       # Denials for specific command/process

# SELinux log analysis
sealert -a /var/log/audit/audit.log            # Analyze all denials with suggestions
tail -f /var/log/audit/audit.log | grep AVC    # Watch for real-time denials

# Generate policy modules for denials
ausearch -m AVC -ts recent | audit2allow -M mypolicy
semodule -i mypolicy.pp                        # Install generated policy module

# Port context management
semanage port -l                               # List port contexts
semanage port -l | grep http                   # HTTP-related ports
semanage port -a -t http_port_t -p tcp 8080    # Add custom HTTP port
semanage port -d -t http_port_t -p tcp 8080    # Delete custom port context
```

User and Role Management

```
# SELinux users (different from Linux users)
semanage user -l                    # List SELinux users
semanage login -l                   # List user login mappings

# Map Linux users to SELinux users
semanage login -a -s user_u john    # Map john to user_u
semanage login -m -s staff_u jane   # Change jane's mapping
semanage login -d john              # Delete mapping

# Role management
semanage user -a -R "staff_r sysadm_r" myuser_u  # Add roles to SELinux user
```

Command Reference Table

Command	Purpose	Key Options	Example
`getenforce`	Check SELinux mode		`getenforce`
`setenforce`	Change mode temporarily	`0` (permissive), `1` (enforcing)	`setenforce 0`
`ls -Z`	Show file contexts	`-R` (recursive)	`ls -lZ /var/www/`
`restorecon`	Restore file contexts	`-R`, `-v`	`restorecon -Rv /var/www/`
`semanage fcontext`	Manage context policies	`-a`, `-m`, `-d`, `-l`	`semanage fcontext -a -t httpd_exec_t "/opt/bin/.*"`
`setsebool`	Set SELinux booleans	`-P` (persistent)	`setsebool -P httpd_can_network_connect on`
`ausearch`	Search audit logs	`-m AVC`, `-ts`	`ausearch -m AVC -ts recent`

---

4. Procedural Workflows

Standard Procedure: Configure Custom Application for SELinux

1 - Install application in non-standard location

```
# Example: Installing custom web application
mkdir -p /opt/webapp/{bin,conf,logs}
# Copy application files to /opt/webapp/
```

2 - **Set appropriate file contexts**

```
# Define context policies for the application
semanage fcontext -a -t httpd_exec_t "/opt/webapp/bin(/.*)?"
semanage fcontext -a -t httpd_config_t "/opt/webapp/conf(/.*)?"
semanage fcontext -a -t httpd_log_t "/opt/webapp/logs(/.*)?"

# Apply the contexts to existing files
restorecon -Rv /opt/webapp/
```

3 - **Configure required booleans**

```
# Enable booleans for application functionality
setsebool -P httpd_can_network_connect on
setsebool -P httpd_can_network_relay on
```

4 - **Test and troubleshoot**

```
# Start the service and test
systemctl start httpd

# Monitor for denials
ausearch -m AVC -ts recent

# If denials occur, analyze and fix
sealert -a /var/log/audit/audit.log
```

Standard Procedure: SELinux Troubleshooting

1 - **Identify the problem**

```
# Check if SELinux is causing the issue
getenforce

# Temporarily set to permissive to test
setenforce 0
# Test if application works now
# If yes, SELinux is the cause
setenforce 1
```

2 - **Find specific denials**

```
# Search for recent AVC denials
ausearch -m AVC -ts recent

# Or monitor in real-time
tail -f /var/log/audit/audit.log | grep AVC
```

3 - **Analyze denial messages**

```
# Get human-readable analysis
sealert -a /var/log/audit/audit.log

# Look for specific patterns in denial
ausearch -m AVC -c httpd -ts recent
```

4 - **Apply appropriate fix**

```
# Option 1: Set boolean (if suggested)
setsebool -P boolean_name on

# Option 2: Fix file context
semanage fcontext -a -t correct_type_t "/path/to/file"
restorecon -v /path/to/file

# Option 3: Add port context (for network services)
semanage port -a -t http_port_t -p tcp 8080

# Option 4: Generate custom policy module (last resort)
ausearch -m AVC -ts recent | audit2allow -M mypolicy
semodule -i mypolicy.pp
```

Decision Tree: SELinux Problem Resolution

```
SELinux Access Denied
├── File access denied?
│   ├── Wrong file context? → Use semanage fcontext + restorecon
│   ├── File in wrong location? → Move file or create custom policy
│   └── Process running as wrong domain? → Check process context
├── Network access denied?
│   ├── Port not allowed? → Use semanage port
│   ├── Network connection blocked? → Set network booleans
│   └── Proxy/relay blocked? → Set appropriate booleans
├── Service functionality disabled?
│   ├── Feature disabled by boolean? → Use setsebool -P
│   └── Policy restriction? → Generate custom policy
└── Unknown issue?
    ├── Check ausearch -m AVC → Analyze denial messages
    ├── Use sealert → Get recommendations
    └── Test in permissive mode → Confirm SELinux cause
```

Standard Procedure: Custom Web Service SELinux Setup

1 - Create service structure

```
# Create custom web service directory
mkdir -p /srv/mywebapp/{htdocs,cgi-bin,logs}

# Create sample content
echo "<h1>My Web App</h1>" > /srv/mywebapp/htdocs/index.html
```

2 - Configure SELinux contexts

```
# Set context policies for custom directories
semanage fcontext -a -t httpd_config_t "/srv/mywebapp(/.*)?"
semanage fcontext -a -t httpd_exec_t "/srv/mywebapp/cgi-bin(/.*)?"
semanage fcontext -a -t httpd_log_t "/srv/mywebapp/logs(/.*)?"

# Apply contexts
restorecon -Rv /srv/mywebapp/

# Verify contexts
ls -lZ /srv/mywebapp/
```

3 - **Configure Apache for custom location**

```
# Create Apache virtual host configuration
cat > /etc/httpd/conf.d/mywebapp.conf << 'EOF'
<VirtualHost *:8080>
    DocumentRoot /srv/mywebapp/htdocs
    ErrorLog /srv/mywebapp/logs/error.log
    CustomLog /srv/mywebapp/logs/access.log combined

    <Directory /srv/mywebapp/htdocs>
        AllowOverride None
        Require all granted
    </Directory>
</VirtualHost>
EOF
```

4 - **Configure SELinux for custom port**

```
# Add port 8080 to HTTP port context
semanage port -a -t http_port_t -p tcp 8080

# Verify port context
semanage port -l | grep 8080
```

---

5. Configuration Deep Dive

SELinux Configuration Files

Main Configuration

```
# /etc/selinux/config
SELINUX=enforcing          # enforcing|permissive|disabled
SELINUXTYPE=targeted       # targeted|minimum|mls
```

Policy Files Location

```
# SELinux policy files
/etc/selinux/targeted/                              # Targeted policy files
/etc/selinux/targeted/contexts/files/file_contexts  # Default file contexts
/etc/selinux/targeted/modules/active/               # Active policy modules
```

Understanding SELinux Contexts

Context Format: user:role:type:level

```
# Example contexts:
system_u:object_r:admin_home_t:s0           # Administrative home directory
system_u:object_r:httpd_exec_t:s0           # Apache executable
unconfined_u:unconfined_r:unconfined_t:s0   # Unconfined user process
system_u:system_r:httpd_t:s0                # Apache daemon process
```

Common Types and Their Uses

```
# File types:
httpd_config_t      # Apache configuration files
httpd_log_t         # Apache log files
httpd_exec_t        # Apache executable files
admin_home_t        # Administrator home directories
user_home_t         # Regular user home directories
etc_t               # System configuration files
bin_t               # System binaries

# Process domains:
httpd_t             # Apache web server process
sshd_t              # SSH daemon process
unconfined_t        # Unconfined user processes
kernel_t            # Kernel processes
```

SELinux Booleans Categories

Web Server Booleans

```
httpd_can_network_connect=off      # HTTP proxy connections
httpd_can_network_relay=off        # HTTP reverse proxy
httpd_enable_homedirs=off          # Access user home directories
httpd_builtin_scripting=on         # Built-in scripting support
httpd_enable_cgi=on                # CGI script execution
httpd_can_sendmail=off             # Send email from web apps
```

File Sharing Booleans

```
samba_enable_home_dirs=off         # Samba access to home directories
use_samba_home_dirs=off            # Users access Samba homes
ftpd_anon_write=off                # Anonymous FTP uploads
ftpd_use_nfs=off                   # FTP access to NFS mounts
```

Port Context Management

Common Port Contexts

```
# Web services
http_port_t: 80, 81, 443, 488, 8008, 8009, 8443, 9000
# SSH
ssh_port_t: 22
# FTP
ftp_port_t: 21, 989, 990
# DNS
dns_port_t: 53
# SMTP
smtp_port_t: 25, 465, 587
```

---

6. Hands-On Labs

Lab 6.1: Basic SELinux Configuration (Asghar Ghori Style)

Objective: Understand SELinux modes, contexts, and basic troubleshooting

Steps:

1 - Explore current SELinux status

```
# Check SELinux enforcement status
getenforce
sestatus
sestatus -v

# View SELinux configuration
cat /etc/selinux/config

# Check current process contexts
ps -eZ | head -10

# Check file contexts in common directories
ls -lZ /etc/ | head -5
ls -lZ /var/www/html/
```

2 - **Practice mode changes**

```
# Change to permissive mode temporarily
setenforce 0
getenforce

# Change back to enforcing
setenforce 1
getenforce

# View current boolean settings
getsebool -a | grep httpd | head -5
getsebool httpd_can_network_connect
```

3 - **Work with file contexts**

```
# Create test files
mkdir -p /tmp/selinux-test
touch /tmp/selinux-test/testfile.txt
echo "Test content" > /tmp/selinux-test/testfile.txt

# Check current context
ls -lZ /tmp/selinux-test/

# Change context manually
chcon -t admin_home_t /tmp/selinux-test/testfile.txt
ls -lZ /tmp/selinux-test/testfile.txt

# Restore default context
restorecon -v /tmp/selinux-test/testfile.txt
ls -lZ /tmp/selinux-test/testfile.txt
```

4 - **Practice boolean management**

```
# Check current boolean status
getsebool httpd_enable_homedirs

# Set boolean temporarily
setsebool httpd_enable_homedirs on
getsebool httpd_enable_homedirs

# Set boolean permanently
setsebool -P httpd_enable_homedirs off
getsebool httpd_enable_homedirs
```

Verification:

```
# Verify SELinux is working properly
getenforce
sestatus
ps -eZ | grep httpd || echo "Apache not running"
getsebool -a | grep httpd | head -3
```

Lab 6.2: Advanced Context Management (Sander van Vugt Style)

Objective: Configure custom file contexts and manage SELinux policies

Steps:

1 - **Create custom web application directory**

```
# Create directory structure for custom web app
sudo mkdir -p /srv/webapp/{html,cgi-bin,logs}

# Create sample files
echo "<h1>Custom Web Application</h1>" | sudo tee /srv/webapp/html/index.html
echo "#!/bin/bash" | sudo tee /srv/webapp/cgi-bin/test.cgi
echo "echo 'Content-type: text/html'" | sudo tee -a /srv/webapp/cgi-bin/test.cgi
echo "echo '<h2>CGI Test</h2>'" | sudo tee -a /srv/webapp/cgi-bin/test.cgi
sudo chmod +x /srv/webapp/cgi-bin/test.cgi

# Check initial contexts
ls -lZ /srv/webapp/
ls -lZ /srv/webapp/html/
ls -lZ /srv/webapp/cgi-bin/
```

2 - **Configure appropriate SELinux contexts**

```
# Add file context policies for custom directories
sudo semanage fcontext -a -t httpd_config_t "/srv/webapp(/.*)?"
sudo semanage fcontext -a -t httpd_exec_t "/srv/webapp/cgi-bin(/.*)?"
sudo semanage fcontext -a -t httpd_log_t "/srv/webapp/logs(/.*)?"

# Apply the contexts to existing files
sudo restorecon -Rv /srv/webapp/

# Verify new contexts
ls -lZ /srv/webapp/
ls -lZ /srv/webapp/html/
ls -lZ /srv/webapp/cgi-bin/
```

3 - **Configure custom port context**

```
# Check current port contexts for HTTP
semanage port -l | grep http_port_t

# Add custom port for web application
sudo semanage port -a -t http_port_t -p tcp 8080

# Verify port was added
semanage port -l | grep 8080
```

4 - **Test context policies**

```
# Create new file in managed directory
echo "<p>New file</p>" | sudo tee /srv/webapp/html/newfile.html

# Check if it got the correct context automatically
ls -lZ /srv/webapp/html/newfile.html

# Create file with wrong context and fix it
sudo touch /tmp/wrongcontext.html
sudo mv /tmp/wrongcontext.html /srv/webapp/html/
ls -lZ /srv/webapp/html/wrongcontext.html

# Restore correct context
sudo restorecon -v /srv/webapp/html/wrongcontext.html
ls -lZ /srv/webapp/html/wrongcontext.html
```

Verification:

```
# Verify all context policies are working
sudo semanage fcontext -l | grep "/srv/webapp"
ls -lZ /srv/webapp/ /srv/webapp/html/ /srv/webapp/cgi-bin/
semanage port -l | grep 8080
```

Lab 6.3: SELinux Troubleshooting Scenario (Synthesis Challenge)

Objective: Diagnose and resolve SELinux access denials in a realistic scenario

Scenario: A custom web application is experiencing SELinux denials and needs systematic troubleshooting

Requirements:

- Set up a scenario that generates SELinux denials
- Use proper troubleshooting methodology
- Resolve issues with appropriate SELinux configurations
- Document the troubleshooting process

Solution Steps:

1 - **Create problematic scenario**

```
# Create a web service that will have SELinux issues
sudo mkdir -p /opt/customapp/{bin,data,config}

# Create executable that will cause SELinux denial
cat << 'EOF' | sudo tee /opt/customapp/bin/webapp.sh
#!/bin/bash
# Simple web service that writes to non-standard location
echo "Starting custom web application..."
echo "$(date): Web app started" > /opt/customapp/data/app.log
echo "Config loaded" >> /opt/customapp/config/status.conf
python3 -m http.server 9090 --directory /opt/customapp/data
EOF

sudo chmod +x /opt/customapp/bin/webapp.sh

# Create initial files with wrong contexts
echo "Initial log entry" | sudo tee /opt/customapp/data/app.log
echo "config=loaded" | sudo tee /opt/customapp/config/status.conf

# Check contexts (these will be wrong)
ls -lZ /opt/customapp/bin/
ls -lZ /opt/customapp/data/
ls -lZ /opt/customapp/config/
```

2 - **Attempt to run and identify SELinux denials**

```
# Try to run the application (this may cause SELinux denials)
sudo /opt/customapp/bin/webapp.sh &
WEBAPP_PID=$!
sleep 2
sudo kill $WEBAPP_PID 2>/dev/null

# Check for recent SELinux denials
sudo ausearch -m AVC -ts recent

# If no denials yet, try accessing the files as web service would
sudo -u apache cat /opt/customapp/config/status.conf 2>/dev/null || echo
"Access denied (this is expected)"

# Check for denials again
sudo ausearch -m AVC -ts recent
```

3 - **Systematic troubleshooting**

```
# Step 1: Identify the denial details
echo "=== SELINUX DENIAL ANALYSIS ==="
sudo ausearch -m AVC -ts recent | head -20

# Step 2: Get human-readable analysis
sudo sealert -a /var/log/audit/audit.log | tail -50

# Step 3: Check current contexts
echo "=== CURRENT CONTEXTS ==="
ls -lZ /opt/customapp/bin/webapp.sh
ls -lZ /opt/customapp/data/app.log
ls -lZ /opt/customapp/config/status.conf

# Step 4: Check what contexts should be
echo "=== CHECKING POLICY FOR SIMILAR PATHS ==="
sudo semanage fcontext -l | grep -E "(httpd|web)" | head -5
```

4 - **Apply systematic fixes**

```
# Fix 1: Set appropriate file contexts for web application
echo "=== APPLYING CONTEXT FIXES ==="

# Define appropriate contexts for our custom app
sudo semanage fcontext -a -t httpd_exec_t "/opt/customapp/bin(/.*)?"
sudo semanage fcontext -a -t httpd_config_t "/opt/customapp/config(/.*)?"
sudo semanage fcontext -a -t httpd_log_t "/opt/customapp/data(/.*)?"

# Apply contexts to existing files
sudo restorecon -Rv /opt/customapp/

# Fix 2: Add custom port if needed
if ! semanage port -l | grep -q "9090"; then
    echo "Adding custom port 9090 to HTTP context..."
    sudo semanage port -a -t http_port_t -p tcp 9090
fi

# Fix 3: Enable necessary booleans
echo "=== SETTING REQUIRED BOOLEANS ==="
sudo setsebool -P httpd_can_network_connect on
sudo setsebool -P httpd_builtin_scripting on

# Verify contexts after fixes
echo "=== CONTEXTS AFTER FIXES ==="
ls -lZ /opt/customapp/bin/
ls -lZ /opt/customapp/data/
ls -lZ /opt/customapp/config/
```

5 - **Test resolution and document**

```
# Test the application again
echo "=== TESTING AFTER FIXES ==="
sudo /opt/customapp/bin/webapp.sh &
WEBAPP_PID=$!
sleep 3

# Test if it's working (try to access the service)
curl -s http://localhost:9090/ >/dev/null && echo "Web service accessible" || echo "Web service
not accessible"

# Clean up the test
sudo kill $WEBAPP_PID 2>/dev/null

# Check for new denials
NEW_DENIALS=$(sudo ausearch -m AVC -ts recent | wc -l)
if [ "$NEW_DENIALS" -eq 0 ]; then
    echo "SUCCESS: No new SELinux denials found"
else
    echo "ISSUE: Still getting SELinux denials"
    sudo ausearch -m AVC -ts recent | tail -5
fi

# Create troubleshooting documentation
cat > /tmp/selinux-troubleshooting-report.md << EOF
# SELinux Troubleshooting Report
Date: $(date)

## Problem Description
Custom web application experiencing SELinux access denials

## Investigation Steps
1. **Initial Analysis**
     - Checked for AVC denials: \`ausearch -m AVC -ts recent\`
     - Analyzed denials: \`sealert -a /var/log/audit/audit.log\`
     - Reviewed current file contexts

2. **Root Cause**
     - Custom application files had incorrect SELinux contexts
     - Files in /opt/customapp/ had default contexts instead of web server contexts
     - Custom port 9090 not in HTTP port context

## Resolution Applied
1. **File Context Policies**:
     \`\`\`bash
     semanage fcontext -a -t httpd_exec_t "/opt/customapp/bin(/.*)?"
     semanage fcontext -a -t httpd_config_t "/opt/customapp/config(/.*)?"
     semanage fcontext -a -t httpd_log_t "/opt/customapp/data(/.*)?"
     restorecon -Rv /opt/customapp/
     \`\`\`

2. **Port Context**:
     \`\`\`bash
     semanage port -a -t http_port_t -p tcp 9090
     \`\`\`
```

```
3. **Boolean Configuration**:
     \`\`\`bash
     setsebool -P httpd_can_network_connect on
     setsebool -P httpd_builtin_scripting on
     \`\`\`

## Verification
- No new AVC denials after fixes
- Application runs without SELinux interference
- All file contexts properly configured

## Lessons Learned
- Custom applications need explicit SELinux context policies
- Always check ausearch and sealert for troubleshooting guidance
- File contexts should match the application's intended use
- Port contexts required for non-standard network services
EOF

echo "=== TROUBLESHOOTING REPORT CREATED ==="
cat /tmp/selinux-troubleshooting-report.md
```

Verification:

```
# Final comprehensive verification
echo "=== FINAL SELINUX CONFIGURATION STATUS ==="
getenforce
sudo semanage fcontext -l | grep "/opt/customapp"
sudo semanage port -l | grep 9090
sudo getsebool httpd_can_network_connect
sudo getsebool httpd_builtin_scripting
ls -lZ /opt/customapp/bin/ /opt/customapp/data/ /opt/customapp/config/
echo "Recent AVC denials: $(sudo ausearch -m AVC -ts recent 2>/dev/null | wc -l)"
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Service Won't Start After SELinux Enforcement

Symptoms:

- Service fails to start when SELinux is enforcing
- Works fine in permissive mode
- No clear error messages in service logs

Diagnosis:

```
# Check SELinux denials
ausearch -m AVC -ts recent
sealert -a /var/log/audit/audit.log

# Check service contexts
systemctl status servicename
ps -eZ | grep servicename

# Verify file contexts
ls -lZ /path/to/service/files
```

Resolution:

```
# Common fixes based on denial type:

# Fix file contexts
semanage fcontext -a -t appropriate_type_t "/path/to/service/files(/.*)?"
restorecon -Rv /path/to/service/files

# Enable required booleans
setsebool -P boolean_name on

# Add custom port contexts
semanage port -a -t service_port_t -p tcp port_number

# Generate custom policy (last resort)
ausearch -m AVC -ts recent | audit2allow -M servicepolicy
semodule -i servicepolicy.pp
```

Prevention: Plan SELinux configuration during service deployment

Issue 2: Web Application Cannot Access Files

Symptoms:

- HTTP 403 forbidden errors
- Web server logs show permission denied
- Files have correct traditional permissions

Diagnosis:

```
# Check file contexts for web content
ls -lZ /var/www/html/
ls -lZ /path/to/web/content/

# Check web server process context
ps -eZ | grep httpd

# Look for AVC denials related to httpd
ausearch -m AVC -c httpd -ts recent
```

Resolution:

```
# Fix web content contexts
semanage fcontext -a -t httpd_config_t "/path/to/web/content(/.*)?"
restorecon -Rv /path/to/web/content/

# Enable web server booleans as needed
setsebool -P httpd_enable_homedirs on      # For user directories
setsebool -P httpd_can_network_connect on  # For proxy/external connections
setsebool -P httpd_builtin_scripting on    # For CGI/scripts
```

Issue 3: Custom Port Access Denied

Symptoms:

- Service cannot bind to custom port
- "Permission denied" when connecting to port
- Standard ports work fine

Diagnosis:

```
# Check current port contexts
semanage port -l | grep port_number
semanage port -l | grep service_name

# Look for port-related denials
ausearch -m AVC -ts recent | grep port
```

Resolution:

```
# Add port to appropriate context
semanage port -a -t http_port_t -p tcp 8080
semanage port -a -t ssh_port_t -p tcp 2222

# Verify port was added
semanage port -l | grep port_number
```

Diagnostic Command Sequence

```
# SELinux troubleshooting workflow
getenforce                          # Check SELinux mode
ausearch -m AVC -ts recent          # Check for recent denials
sealert -a /var/log/audit/audit.log # Analyze denials
ps -eZ | grep process-name          # Check process contexts
ls -lZ /path/to/files               # Check file contexts
semanage fcontext -l | grep path    # Check context policies
getsebool -a | grep relevant        # Check boolean settings
```

Log File Analysis

- `/var/log/audit/audit.log`: Primary SELinux denial log
- `ausearch -m AVC`: Extract AVC (Access Vector Cache) denials
- `sealert`: Human-readable denial analysis with suggestions
- `journalctl -u auditd`: Audit daemon logs

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Status and mode
getenforce                                  # Check current mode
setenforce 0|1                              # Change mode temporarily
sestatus                                    # Detailed status

# File contexts
ls -lZ file                                 # Show file context
restorecon -Rv /path/                       # Restore file contexts
semanage fcontext -a -t type "/path(/.*)?"  # Add context policy

# Booleans
getsebool boolean_name                      # Check boolean status
setsebool -P boolean_name on                # Set boolean permanently

# Troubleshooting
ausearch -m AVC -ts recent                  # Recent denials
sealert -a /var/log/audit/audit.log         # Analyze denials
```

Common File Types

- **httpd_config_t**: Web server configuration files
- **httpd_exec_t**: Web server executable files
- **httpd_log_t**: Web server log files
- **admin_home_t**: Administrator home directories
- **user_home_t**: User home directories
- **etc_t**: System configuration files

Critical Booleans

- **httpd_can_network_connect**: HTTP proxy connections
- **httpd_enable_homedirs**: Access user home directories
- **samba_enable_home_dirs**: Samba home directory access
- **ftpd_anon_write**: Anonymous FTP uploads

Port Context Commands

```
semanage port -l                            # List port contexts
semanage port -a -t http_port_t -p tcp 8080  # Add HTTP port
semanage port -d -t http_port_t -p tcp 8080  # Remove port context
```

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between discretionary access control and mandatory access control in SELinux?

   Answer

   Discretionary access control (traditional permissions) allows users to control access to their own files. Mandatory access control (SELinux) enforces system-wide security policies that users cannot override, providing an additional security layer based on security contexts and policies.

2. **Question**: Why is it better to use semanage fcontext instead of chcon for permanent changes?

   Answer

   `chcon` only changes the context temporarily - it's lost if the file is moved, copied, or if `restorecon` is run. `semanage fcontext` creates a permanent policy rule, so the context is automatically applied to matching files and persists through system operations.

3. **Question**: When would you create a custom SELinux policy module instead of using existing tools?

   Answer

   Custom policy modules are a last resort when legitimate application behavior triggers denials that can't be resolved with existing booleans, file contexts, or port contexts. They're typically needed for applications with unusual security requirements or behaviors not covered by standard policies.

Practical Scenarios

1. **Scenario**: Web server needs to connect to a database on a non-standard port.

   Solution

   Enable the `httpd_can_network_connect` boolean with `setsebool -P httpd_can_network_connect on` and possibly add the database port to appropriate context with `semanage port`.

2. **Scenario**: Custom application installed in /opt needs to be accessed by Apache.

   Solution

   Set appropriate contexts with `semanage fcontext -a -t httpd_config_t "/opt/myapp(/.*)?"` and apply with `restorecon -Rv /opt/myapp/`.

Command Challenges

1. **Challenge**: Find all files in /var/www that have the wrong SELinux context.

Answer

`find /var/www -exec ls -lZ {} \; | grep -v httpd_config_t | grep -v httpd_exec_t`

2. **Challenge**: Create a policy to allow Apache to write to a custom log directory.
   Answer:

```
semanage fcontext -a -t httpd_log_t "/custom/logs(/.*)?"
restorecon -Rv /custom/logs/
setsebool -P httpd_can_network_connect on  # if network logging
```

---

10. Exam Strategy

Topic-Specific Tips

- Always check `getenforce` first when troubleshooting access issues
- Use `ausearch -m AVC -ts recent` to find specific SELinux denials
- Remember that `setsebool -P` makes changes persistent (crucial for exam)
- Practice the systematic approach: check contexts, booleans, ports, then custom policies

Common Exam Scenarios

1. **Scenario**: Configure web server to serve content from custom directory
   Approach: Use `semanage fcontext` to set appropriate httpd contexts, then `restorecon`
2. **Scenario**: Web application needs network access
   Approach: Enable `httpd_can_network_connect` boolean with `-P` flag
3. **Scenario**: Service won't start, works in permissive mode
   Approach: Check `ausearch -m AVC`, analyze with `sealert`, apply appropriate fix

Time Management

- **SELinux boolean changes**: 2-3 minutes including verification
- **File context configuration**: 5-7 minutes for complete setup
- **Troubleshooting denials**: 8-12 minutes depending on complexity
- **Always verify**: Test functionality after applying SELinux changes

Pitfalls to Avoid

- Don't disable SELinux unless explicitly asked (major point deduction)
- Remember `-P` flag with `setsebool` for persistence
- Don't use `chcon` for permanent changes - use `semanage fcontext`

- Always run `restorecon` after setting context policies
- Test applications after SELinux changes to ensure they work

---

Summary

Key Takeaways

- **SELinux provides mandatory access control** - an essential security layer beyond traditional permissions
- **Understand the context format** - user:role:type:level labels control access
- **Use proper tools for permanent changes** - `semanage` for policies, `setsebool -P` for booleans
- **Systematic troubleshooting works** - check denials, analyze with sealert, apply appropriate fixes

Critical Commands to Remember

```
getenforce                                  # Check SELinux status
setenforce 0|1                              # Change mode temporarily
setsebool -P boolean_name on                # Set boolean permanently
semanage fcontext -a -t type "/path(/.*)?"  # Add file context policy
restorecon -Rv /path/                       # Apply context policies
ausearch -m AVC -ts recent                  # Find recent denials
sealert -a /var/log/audit/audit.log         # Analyze denials
```

Next Steps

- Continue to Module 10: Firewall Configuration
- Practice SELinux scenarios in the Vagrant environment
- Review related topics: File Permissions, Process Management

---

Navigation: ← Network Configuration | Index | Next → Firewall Configuration

2.12 10 - Firewall Configuration

Navigation: ← SELinux Management | Index | Next → Boot Process & GRUB

1. Executive Summary

Topic Scope: Firewall configuration using firewalld, zones, services, ports, and rich rules in RHEL 10

RHCSA Relevance: Essential security skill - firewall management is fundamental for server security

Exam Weight: Medium-High - Firewall tasks appear regularly in security-focused exam scenarios

Prerequisites: Understanding of network services, ports, and basic TCP/IP concepts

Related Topics: Network Configuration, SELinux Management, Service Management

2. Conceptual Foundation

Core Theory

RHEL 10 uses firewalld as the default firewall management service, which provides:

- **Zone-based management**: Different security levels for different network contexts
- **Dynamic configuration**: Changes without service restart or connection drops
- **Service definitions**: Predefined collections of ports and protocols
- **Rich rules**: Advanced firewall rules with complex conditions
- **Runtime vs permanent**: Immediate changes vs persistent configuration

Real-World Applications

- **Web server protection**: Allowing HTTP/HTTPS while blocking unauthorized access
- **SSH hardening**: Restricting remote access to specific networks or ports
- **Database security**: Limiting database access to application servers only
- **Service isolation**: Separating different services into appropriate security zones
- **Compliance requirements**: Meeting security standards for regulated environments

Common Misconceptions

- **iptables vs firewalld**: RHEL 10 uses firewalld by default, not direct iptables
- **Zone complexity**: Zones are logical groupings, not physical network segments
- **Runtime changes**: Runtime changes are temporary unless made permanent
- **Service vs port rules**: Services are collections of ports with meaningful names
- **Default deny**: firewalld uses default deny - only explicitly allowed traffic passes

Key Terminology

- **Zone**: Security context with specific rules applied to network interfaces
- **Service**: Predefined firewall rule for common applications (http, ssh, etc.)
- **Port**: Specific TCP/UDP port number that can be opened
- **Rich rule**: Complex firewall rule with advanced conditions and actions
- **Runtime configuration**: Currently active rules (lost on restart)
- **Permanent configuration**: Persistent rules saved to configuration files
- **Source**: IP address or network range for rule targeting
- **Target**: Default action for traffic not matching any rules in a zone

---

3. Command Mastery

Basic Firewall Status and Control

```
# Service management
systemctl status firewalld            # Check firewalld service status
systemctl start firewalld             # Start firewall service
systemctl stop firewalld              # Stop firewall service
systemctl enable firewalld            # Enable firewall at boot
systemctl disable firewalld           # Disable firewall at boot

# Firewall status
firewall-cmd --state                  # Check if firewall is running
firewall-cmd --get-active-zones       # Show active zones with interfaces
firewall-cmd --list-all               # Show all rules for default zone
firewall-cmd --list-all-zones         # Show rules for all zones
firewall-cmd --get-default-zone       # Show default zone
```

Zone Management

```
# Zone operations
firewall-cmd --get-zones              # List all available zones
firewall-cmd --set-default-zone=public # Set default zone
firewall-cmd --zone=public --list-all  # Show rules for specific zone
firewall-cmd --get-zone-of-interface=ens3  # Show zone of interface

# Interface assignment
firewall-cmd --zone=public --add-interface=ens3     # Add interface to zone
firewall-cmd --zone=dmz --change-interface=ens3     # Move interface to zone
firewall-cmd --zone=public --remove-interface=ens3  # Remove interface from zone

# Source-based zones
firewall-cmd --zone=trusted --add-source=192.168.1.0/24  # Add source to zone
firewall-cmd --zone=public --remove-source=192.168.1.100 # Remove source
firewall-cmd --zone=work --list-sources                   # List sources in zone
```

Service Management

```
# Service operations
firewall-cmd --get-services            # List all available services
firewall-cmd --list-services           # List enabled services in default zone
firewall-cmd --zone=public --list-services  # List services in specific zone

# Add/remove services
firewall-cmd --add-service=http         # Add HTTP service (runtime)
firewall-cmd --add-service=https        # Add HTTPS service (runtime)
firewall-cmd --remove-service=ssh       # Remove SSH service (runtime)
firewall-cmd --zone=dmz --add-service=mysql  # Add service to specific zone

# Permanent service changes
firewall-cmd --permanent --add-service=http     # Add service permanently
firewall-cmd --permanent --remove-service=ftp   # Remove service permanently
firewall-cmd --reload                           # Apply permanent changes

# Query services
firewall-cmd --query-service=http       # Check if service is enabled
firewall-cmd --zone=public --query-service=ssh  # Check service in specific zone
```

Port Management

```
# Port operations
firewall-cmd --list-ports              # List open ports in default zone
firewall-cmd --zone=public --list-ports # List ports in specific zone

# Add/remove ports
firewall-cmd --add-port=8080/tcp       # Add TCP port 8080
firewall-cmd --add-port=53/udp         # Add UDP port 53
firewall-cmd --add-port=8000-8010/tcp  # Add port range
firewall-cmd --remove-port=8080/tcp    # Remove port

# Permanent port changes
firewall-cmd --permanent --add-port=9090/tcp    # Add port permanently
firewall-cmd --permanent --remove-port=23/tcp   # Remove port permanently

# Query ports
firewall-cmd --query-port=8080/tcp     # Check if port is open
```

Rich Rules

```
# Rich rule syntax examples
# Accept SSH from specific subnet
firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.1.0/24" service
name="ssh" accept'

# Reject HTTP from specific IP
firewall-cmd --add-rich-rule='rule family="ipv4" source address="10.0.0.50" service name="http"
reject'

# Allow port 8080 from specific network
firewall-cmd --add-rich-rule='rule family="ipv4" source address="172.16.0.0/16" port
protocol="tcp" port="8080" accept'

# Log and accept HTTPS traffic
firewall-cmd --add-rich-rule='rule family="ipv4" service name="https" log prefix="HTTPS-ACCESS"
level="info" accept'

# Rate limit SSH connections
firewall-cmd --add-rich-rule='rule family="ipv4" service name="ssh" accept limit value="3/m"'

# Rich rule management
firewall-cmd --list-rich-rules         # List all rich rules
firewall-cmd --remove-rich-rule='rule...'  # Remove specific rich rule
```

Configuration Persistence

```
# Runtime vs permanent configurations
firewall-cmd --list-all                # Show runtime configuration
firewall-cmd --permanent --list-all    # Show permanent configuration

# Make changes permanent
firewall-cmd --runtime-to-permanent    # Save current runtime to permanent
firewall-cmd --permanent --add-service=http  # Add to permanent directly
firewall-cmd --reload                  # Load permanent config to runtime

# Configuration management
firewall-cmd --complete-reload         # Full reload (breaks connections)
firewall-cmd --check-config           # Validate configuration
```

Advanced Features

```
# Custom services
firewall-cmd --permanent --new-service=myapp       # Create custom service
firewall-cmd --permanent --service=myapp --set-description="My Application"
firewall-cmd --permanent --service=myapp --add-port=9000/tcp
firewall-cmd --reload

# Port forwarding
firewall-cmd --add-forward-port=port=80:proto=tcp:toport=8080:toaddr=192.168.1.100
firewall-cmd --add-masquerade          # Enable NAT/masquerading

# Direct rules (advanced)
firewall-cmd --direct --add-rule ipv4 filter INPUT 0 -p tcp --dport 22 -j ACCEPT
```

Command Reference Table

Command	Purpose	Key Options	Example
`firewall-cmd --state`	Check firewall status		`firewall-cmd --state`
`firewall-cmd --list-all`	Show zone configuration	`--zone=name`	`firewall-cmd --zone=public --list-all`
`firewall-cmd --add-service`	Add service rule	`--permanent`, `--zone=name`	`firewall-cmd --permanent --add-service=http`
`firewall-cmd --add-port`	Add port rule	`--permanent`, `--zone=name`	`firewall-cmd --add-port=8080/tcp`
`firewall-cmd --add-rich-rule`	Add complex rule	`--permanent`	`firewall-cmd --add-rich-rule='rule...'`
`firewall-cmd --reload`	Apply permanent config		`firewall-cmd --reload`

4. Procedural Workflows

Standard Procedure: Configure Web Server Firewall

1. **Check current firewall status**

```
systemctl status firewalld
firewall-cmd --state
firewall-cmd --get-default-zone
firewall-cmd --list-all
```

2. **Add web services**

```
# Add HTTP and HTTPS services
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https

# Apply changes
firewall-cmd --reload
```

3. **Verify configuration**

```
firewall-cmd --list-services
firewall-cmd --query-service=http
firewall-cmd --query-service=https
```

4. **Test connectivity**

```
# Test from external system
telnet server-ip 80
telnet server-ip 443
```

Standard Procedure: Restrict SSH Access

1. **Check current SSH access**

```
firewall-cmd --query-service=ssh
firewall-cmd --list-all | grep ssh
```

2. **Remove SSH from default zone and add restricted access**

```
# Remove SSH from public zone
firewall-cmd --permanent --remove-service=ssh

# Add SSH access only from management network
firewall-cmd --permanent --add-rich-
rule='rule family="ipv4" source address="192.168.1.0/24" service name="ssh" accept'

# Apply changes
firewall-cmd --reload
```

3. **Verify restricted access**

```
firewall-cmd --list-rich-rules
firewall-cmd --list-services | grep ssh || echo "SSH not in services"
```

Decision Tree: Firewall Rule Strategy

```
Firewall Configuration Need
├── Standard service (http, ssh, ftp)?
│   ├── Use predefined service → --add-service=name
│   └── Custom port needed? → --add-port=port/protocol
├── Source-based restrictions?
│   ├── Simple allow/deny? → Use zones with sources
│   └── Complex conditions? → Use rich rules
├── Multiple conditions?
│   ├── Log traffic? → Rich rules with logging
│   ├── Rate limiting? → Rich rules with limits
│   └── Time-based? → Rich rules with time conditions
└── Advanced networking?
    ├── Port forwarding? → --add-forward-port
    ├── NAT/Masquerading? → --add-masquerade
    └── Direct iptables? → --direct rules
```

Standard Procedure: Database Server Security

1. **Create dedicated zone for database servers**

```
# Create database zone
firewall-cmd --permanent --new-zone=database
firewall-cmd --permanent --zone=database --set-description="Database servers zone"
firewall-cmd --permanent --zone=database --set-target=DROP
```

2. **Configure database zone rules**

```
# Allow SSH from management network
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'

# Allow MySQL from application servers
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.10.0/24" port protocol="tcp" port="3306" accept'

# Apply changes
firewall-cmd --reload
```

3. **Assign interface to database zone**

```
# Move interface to database zone
firewall-cmd --permanent --zone=database --change-interface=ens3
firewall-cmd --reload

# Verify assignment
firewall-cmd --get-zone-of-interface=ens3
firewall-cmd --zone=database --list-all
```

---

5. Configuration Deep Dive

Firewall Zones Overview

Default Zones and Their Purposes

```
# Drop zone - deny all incoming, allow outgoing
# Target: DROP - most restrictive

# Block zone - reject all incoming with icmp-host-prohibited
# Target: %%REJECT%% - provides feedback to sender

# Public zone - default for public networks
# Target: default - deny all except explicitly allowed
# Default services: ssh, dhcpv6-client

# External zone - for external networks with masquerading
# Target: default - includes NAT functionality

# DMZ zone - for demilitarized zone servers
# Target: default - limited services for public access

# Work zone - for work networks
# Target: default - allows some additional services

# Home zone - for home networks
# Target: default - more permissive than public

# Internal zone - for internal networks
# Target: default - trusts internal traffic more

# Trusted zone - all traffic allowed
# Target: ACCEPT - least restrictive
```

Zone Configuration Files

```
# Zone configuration location
/etc/firewalld/zones/         # Custom and modified zones
/usr/lib/firewalld/zones/     # Default system zones

# Example zone file: /etc/firewalld/zones/public.xml
<?xml version="1.0" encoding="utf-8"?>
<zone>
  <short>Public</short>
  <description>For use in public areas.</description>
  <service name="ssh"/>
  <service name="dhcpv6-client"/>
  <port protocol="tcp" port="80"/>
  <rule family="ipv4">
    <source address="192.168.1.0/24"/>
    <service name="http"/>
    <accept/>
  </rule>
</zone>
```

Service Definitions

Service Configuration Files

```
# Service definitions location
/etc/firewalld/services/      # Custom services
/usr/lib/firewalld/services/  # Default system services

# Example service file: /usr/lib/firewalld/services/http.xml
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>WWW (HTTP)</short>
  <description>HTTP is the protocol used to serve Web pages.</description>
  <port protocol="tcp" port="80"/>
</service>
```

Creating Custom Services

```
# Create custom web application service
firewall-cmd --permanent --new-service=webapp
firewall-cmd --permanent --service=webapp --set-description="Custom Web Application"
firewall-cmd --permanent --service=webapp --add-port=8080/tcp
firewall-cmd --permanent --service=webapp --add-port=8443/tcp
firewall-cmd --reload

# Use custom service
firewall-cmd --permanent --add-service=webapp
```

Rich Rules Syntax

Rich Rule Components

```
# Basic syntax:
rule [family="ipv4|ipv6"]
     [source [address="address[/mask]"] [mac="mac-address"] [ipset="ipset-name"]]
     [destination [address="address[/mask]"]]
     [element]
     [log [prefix="prefix-text"] [level="log-level"] [limit value="rate/duration"]]
     [audit]
     [action]

# Elements can be:
# service name="service-name"
# port protocol="tcp|udp" port="port-number|port-range"
# protocol value="protocol"
# icmp-block name="icmp-type"
# masquerade
# forward-port port="port-number" protocol="tcp|udp" to-port="port-number" to-addr="address"

# Actions can be:
# accept, reject [type="reject-type"], drop
```

Rich Rule Examples

```
# Accept SSH from management network with logging
firewall-cmd --add-rich-
rule='rule family="ipv4" source address="10.0.1.0/24" service name="ssh" log prefix="SSH-MGMT"
level="info" accept'

# Reject FTP from specific IP
firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.1.100" service
name="ftp" reject type="icmp-admin-prohibited"'

# Allow HTTP with rate limiting
firewall-cmd --add-rich-rule='rule family="ipv4" service name="http" accept limit value="100/s"'

# Port forwarding with rich rule
firewall-cmd --add-rich-rule='rule family="ipv4" forward-port port="2222" protocol="tcp" to-
port="22"'
```

---

6. Hands-On Labs

Lab 6.1: Basic Firewall Configuration (Asghar Ghori Style)

Objective: Configure basic firewall rules for common services

Steps:

1. **Explore current firewall configuration**

```
# Check firewall status and default configuration
systemctl status firewalld
firewall-cmd --state
firewall-cmd --get-default-zone
firewall-cmd --get-active-zones
firewall-cmd --list-all

# Check available zones and services
firewall-cmd --get-zones
firewall-cmd --get-services | head -20
```

2. **Configure web server firewall rules**

```
# Add HTTP and HTTPS services permanently
firewall-cmd --permanent --add-service=http
firewall-cmd --permanent --add-service=https

# Add custom port for web application
firewall-cmd --permanent --add-port=8080/tcp

# Apply changes
firewall-cmd --reload

# Verify configuration
firewall-cmd --list-services
firewall-cmd --list-ports
firewall-cmd --list-all
```

3. **Test runtime vs permanent changes**

```
# Add FTP service to runtime only
firewall-cmd --add-service=ftp

# Check runtime vs permanent
firewall-cmd --list-services
firewall-cmd --permanent --list-services

# Reload and see FTP disappear
firewall-cmd --reload
firewall-cmd --list-services

# Add FTP permanently
firewall-cmd --permanent --add-service=ftp
firewall-cmd --reload
firewall-cmd --list-services
```

4. **Practice zone management**

```
# Check current zone assignment
firewall-cmd --get-zone-of-interface=ens3

# Temporarily change zone
firewall-cmd --zone=work --change-interface=ens3
firewall-cmd --get-active-zones

# Check different zone rules
firewall-cmd --list-all
firewall-cmd --zone=work --list-all

# Change back to public
firewall-cmd --zone=public --change-interface=ens3
```

Verification:

```
# Complete verification of configuration
firewall-cmd --get-default-zone
firewall-cmd --get-active-zones
firewall-cmd --list-all
firewall-cmd --permanent --list-all
systemctl is-enabled firewalld
```

Lab 6.2: Advanced Firewall Rules (Sander van Vugt Style)

Objective: Implement complex firewall rules using rich rules and custom zones

Steps:

1. **Create custom zone for DMZ servers**

```
# Create DMZ zone
firewall-cmd --permanent --new-zone=dmz-servers
firewall-cmd --permanent --zone=dmz-servers --set-description="DMZ servers with restricted
access"
firewall-cmd --permanent --zone=dmz-servers --set-target=DROP

# Apply changes
firewall-cmd --reload

# Verify zone creation
firewall-cmd --get-zones | grep dmz-servers
firewall-cmd --zone=dmz-servers --list-all
```

2. **Configure rich rules for specific access control**

```
# Allow SSH only from management network
firewall-cmd --permanent --zone=dmz-servers --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'

# Allow HTTP from any source but log connections
firewall-cmd --permanent --zone=dmz-servers --add-rich-rule='rule family="ipv4" service
name="http" log prefix="DMZ-HTTP" level="info" accept'

# Allow HTTPS with rate limiting
firewall-cmd --permanent --zone=dmz-servers --add-rich-rule='rule family="ipv4" service
name="https" accept limit value="50/s"'

# Reject FTP from specific problematic network
firewall-cmd --permanent --zone=dmz-servers --add-rich-rule='rule family="ipv4" source
address="10.0.0.0/8" service name="ftp" reject'

# Apply changes
firewall-cmd --reload
```

3. **Create custom service definition**

```
# Create custom application service
firewall-cmd --permanent --new-service=custom-app
firewall-cmd --permanent --service=custom-app --set-description="Custom Application Service"
firewall-cmd --permanent --service=custom-app --add-port=9090/tcp
firewall-cmd --permanent --service=custom-app --add-port=9091/udp

# Add custom service to DMZ zone
firewall-cmd --permanent --zone=dmz-servers --add-service=custom-app

# Apply changes
firewall-cmd --reload

# Verify custom service
firewall-cmd --get-services | grep custom-app
firewall-cmd --zone=dmz-servers --list-services
```

4. **Test zone assignment and rules**

```
# Assign interface to DMZ zone
firewall-cmd --permanent --zone=dmz-servers --change-interface=ens3
firewall-cmd --reload

# Verify active configuration
firewall-cmd --get-active-zones
firewall-cmd --zone=dmz-servers --list-all

# Test rule queries
firewall-cmd --zone=dmz-servers --query-service=http
firewall-cmd --zone=dmz-servers --query-service=custom-app
firewall-cmd --zone=dmz-servers --list-rich-rules
```

Verification:

```
# Complete verification of advanced configuration
firewall-cmd --get-zones | grep dmz-servers
firewall-cmd --zone=dmz-servers --list-all
firewall-cmd --get-services | grep custom-app
firewall-cmd --zone=dmz-servers --list-rich-rules
```

Lab 6.3: Firewall Troubleshooting Scenario (Synthesis Challenge)

Objective: Diagnose and resolve firewall connectivity issues

Scenario: A multi-tier application is experiencing connectivity issues that appear to be firewall-related

Requirements:

- Web tier needs HTTP/HTTPS access from internet
- Application tier needs custom port access from web tier only
- Database tier needs MySQL access from application tier only
- SSH access should be restricted to management network
- All configuration must be persistent

Solution Steps:

1. **Set up the problematic scenario**

```
# Reset firewall to default state
firewall-cmd --complete-reload

# Create restrictive configuration that will cause problems
firewall-cmd --set-default-zone=drop
firewall-cmd --permanent --zone=drop --remove-service=ssh
firewall-cmd --reload

# This should now block most traffic - simulating the problem
echo "=== PROBLEMATIC CONFIGURATION APPLIED ==="
firewall-cmd --list-all
```

2. **Diagnose connectivity issues**

```
# Step 1: Check firewall status and configuration
echo "=== FIREWALL DIAGNOSIS ==="
systemctl status firewalld
firewall-cmd --state
firewall-cmd --get-default-zone
firewall-cmd --get-active-zones
firewall-cmd --list-all

# Step 2: Check what services should be running
echo "=== EXPECTED SERVICES ==="
echo "Web tier should allow: HTTP (80), HTTPS (443)"
echo "App tier should allow: Custom app (9090) from web tier"
echo "DB tier should allow: MySQL (3306) from app tier"
echo "Management: SSH (22) from 192.168.100.0/24"

# Step 3: Identify the problems
echo "=== PROBLEMS IDENTIFIED ==="
echo "1. Default zone is 'drop' - blocks everything"
echo "2. No services are allowed through firewall"
echo "3. SSH access is completely blocked"
```

3. Implement systematic fix

```
# Fix 1: Change to appropriate default zone
echo "=== FIXING DEFAULT ZONE ==="
firewall-cmd --set-default-zone=public
firewall-cmd --get-default-zone

# Fix 2: Configure web tier access (public zone)
echo "=== CONFIGURING WEB TIER ==="
firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --permanent --zone=public --add-service=https

# Fix 3: Create application zone with restricted access
echo "=== CREATING APPLICATION ZONE ==="
firewall-cmd --permanent --new-zone=application
firewall-cmd --permanent --zone=application --set-description="Application tier with
restricted access"
firewall-cmd --permanent --zone=application --set-target=DROP

# Allow SSH from management network
firewall-cmd --permanent --zone=application --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'

# Allow application port 9090 from web servers (assuming web servers are in 192.168.1.0/24)
firewall-cmd --permanent --zone=application --add-rich-rule='rule family="ipv4" source
address="192.168.1.0/24" port protocol="tcp" port="9090" accept'

# Fix 4: Create database zone with even more restricted access
echo "=== CREATING DATABASE ZONE ==="
firewall-cmd --permanent --new-zone=database
firewall-cmd --permanent --zone=database --set-description="Database tier with MySQL access
from app tier only"
firewall-cmd --permanent --zone=database --set-target=DROP

# Allow SSH from management network
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'

# Allow MySQL from application servers (assuming app servers are in 192.168.2.0/24)
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.2.0/24" port protocol="tcp" port="3306" accept'

# Apply all changes
firewall-cmd --reload
```

4. **Verify and test the fix**

```
# Verify zone configurations
echo "=== VERIFYING CONFIGURATIONS ==="

echo "Public zone (web tier):"
firewall-cmd --zone=public --list-all

echo "Application zone (app tier):"
firewall-cmd --zone=application --list-all

echo "Database zone (db tier):"
firewall-cmd --zone=database --list-all

# Test basic connectivity (simulation)
echo "=== CONNECTIVITY TESTS ==="

# Test HTTP service
firewall-cmd --zone=public --query-service=http && echo "✓ HTTP allowed in public zone" || e
cho "✗ HTTP blocked"

# Test HTTPS service
firewall-cmd --zone=public --query-service=https && echo "✓ HTTPS allowed in public zone"
|| echo "✗ HTTPS blocked"

# Test SSH in management networks
firewall-cmd --zone=application --list-rich-rules | grep ssh && echo "✓ SSH restricted in
application zone" || echo "✗ SSH not configured"
firewall-cmd --zone=database --list-rich-rules | grep ssh && echo "✓ SSH restricted in
database zone" || echo "✗ SSH not configured"

# Verify rich rules for application and database access
firewall-cmd --zone=application --list-rich-rules | grep 9090 && echo "✓ App port 9090
configured" || echo "✗ App port not configured"
firewall-cmd --zone=database --list-rich-rules | grep 3306 && echo "✓ MySQL port 3306
configured" || echo "✗ MySQL port not configured"
```

5. **Document the solution**

```
# Create comprehensive troubleshooting report
cat > /tmp/firewall-troubleshooting-report.md << 'EOF'
# Firewall Troubleshooting Report
Date: $(date)

## Problem Description
Multi-tier application experiencing connectivity issues due to overly restrictive firewall
configuration.

## Issues Found
1. **Default Zone**: Set to 'drop' zone blocking all traffic
2. **Missing Services**: No HTTP/HTTPS services allowed for web tier
3. **SSH Access**: Completely blocked, including from management network
4. **Tier Isolation**: No proper network segmentation between application tiers

## Resolution Strategy
### 1. Zone-Based Security Architecture
- **Public Zone**: Web tier with HTTP/HTTPS access
- **Application Zone**: Restricted access, custom application ports
- **Database Zone**: Highly restricted, MySQL access only from app tier

### 2. Implemented Rules
#### Public Zone (Web Tier)
```bash
firewall-cmd --permanent --zone=public --add-service=http
firewall-cmd --permanent --zone=public --add-service=https
```

Application Zone
```bash
firewall-cmd --permanent --zone=application --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'
firewall-cmd --permanent --zone=application --add-rich-rule='rule family="ipv4" source
address="192.168.1.0/24" port protocol="tcp" port="9090" accept'
```

Database Zone
```bash
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.100.0/24" service name="ssh" accept'
firewall-cmd --permanent --zone=database --add-rich-rule='rule family="ipv4" source
address="192.168.2.0/24" port protocol="tcp" port="3306" accept'
```

Security Benefits
1. **Defense in Depth**: Multiple zones provide layered security
2. **Least Privilege**: Each tier only allows necessary access
3. **Source Restriction**: SSH limited to management network
4. **Service Isolation**: Database only accessible from application tier

Testing Results
- ✓ Web services (HTTP/HTTPS) accessible from internet
- ✓ SSH access restricted to management network (192.168.100.0/24)
- ✓ Application port (9090) accessible from web tier only
```

```
- ✓ MySQL port (3306) accessible from application tier only
- ✓ All configurations are permanent and survive reboots

## Maintenance Commands
```bash
# Check zone assignments
firewall-cmd --get-active-zones

# Review specific zone rules
firewall-cmd --zone=zonename --list-all

# Test service access
firewall-cmd --zone=zonename --query-service=servicename

# Monitor firewall logs (if logging enabled)
journalctl -u firewalld -f
```
EOF

Display the report
echo "=== TROUBLESHOOTING REPORT ==="
cat /tmp/firewall-troubleshooting-report.md
```

Verification:

```
# Final comprehensive verification
echo "=== FINAL FIREWALL CONFIGURATION ==="
firewall-cmd --get-default-zone
firewall-cmd --get-zones | grep -E "(application|database)"
firewall-cmd --zone=public --list-all
firewall-cmd --zone=application --list-all
firewall-cmd --zone=database --list-all
echo "Configuration is persistent: $(firewall-cmd --permanent --list-all > /dev/null 2>&1 && ech
o "Yes" || echo "No")"
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Service Can't Be Accessed After Firewall Configuration

Symptoms:

- Connection timeouts to service ports
- Service logs show no connection attempts
- Works when firewall is disabled

Diagnosis:

```
# Check firewall status and rules
firewall-cmd --state
firewall-cmd --list-all
firewall-cmd --list-services
firewall-cmd --list-ports

# Check if service is in correct zone
firewall-cmd --get-active-zones
firewall-cmd --get-zone-of-interface=interface-name

# Test from different zones
firewall-cmd --zone=trusted --list-all
```

Resolution:

```
# Add missing service or port
firewall-cmd --permanent --add-service=service-name
firewall-cmd --permanent --add-port=port/protocol

# Or check if interface is in wrong zone
firewall-cmd --zone=appropriate-zone --change-interface=interface-name

# Apply changes
firewall-cmd --reload

# Verify fix
firewall-cmd --list-all
```

Prevention: Always test connectivity after firewall changes

Issue 2: Rich Rules Not Working as Expected

Symptoms:

- Traffic not matching rich rule conditions
- Rules appear correct but don't apply
- Complex rules causing conflicts

Diagnosis:

```
# Check rich rule syntax
firewall-cmd --list-rich-rules

# Verify rule order (first match wins)
firewall-cmd --zone=zone-name --list-all

# Check for conflicting rules
firewall-cmd --list-all | grep -A 5 -B 5 problematic-rule
```

Resolution:

```
# Remove problematic rule
firewall-cmd --remove-rich-rule='rule family="ipv4"...'

# Add corrected rule
firewall-cmd --permanent --add-rich-rule='corrected-rule'

# Reload configuration
firewall-cmd --reload
```

Issue 3: Changes Not Persisting After Reboot

Symptoms:

- Firewall rules work but disappear after reboot
- Runtime configuration differs from permanent
- Services fail to start after system restart

Diagnosis:

```
# Compare runtime vs permanent
firewall-cmd --list-all
firewall-cmd --permanent --list-all

# Check configuration files
ls -la /etc/firewalld/zones/
cat /etc/firewalld/zones/zone-name.xml
```

Resolution:

```
# Make runtime changes permanent
firewall-cmd --runtime-to-permanent

# Or add rules with --permanent flag
firewall-cmd --permanent --add-service=service-name
firewall-cmd --reload

# Verify permanent configuration
firewall-cmd --permanent --list-all
```

Diagnostic Command Sequence

```
# Firewall troubleshooting workflow
systemctl status firewalld       # Check service status
firewall-cmd --state             # Verify firewall is running
firewall-cmd --get-active-zones  # Check zone assignments
firewall-cmd --list-all          # Show current rules
firewall-cmd --permanent --list-all  # Show permanent rules
journalctl -u firewalld          # Check firewall logs
```

Log File Analysis

- `journalctl -u firewalld`: Firewalld service logs
- `/var/log/messages`: System messages including firewall events
- **Rich rule logging**: Custom logs based on rich rule log statements
- `dmesg`: Kernel messages about netfilter/iptables

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Basic status
firewall-cmd --state            # Check if firewall running
firewall-cmd --list-all         # Show all rules for default zone
firewall-cmd --get-active-zones  # Show active zones

# Services and ports
firewall-cmd --add-service=http  # Add service (runtime)
firewall-cmd --permanent --add-service=https  # Add service (permanent)
firewall-cmd --add-port=8080/tcp # Add port
firewall-cmd --reload           # Apply permanent changes

# Zones
firewall-cmd --set-default-zone=public  # Set default zone
firewall-cmd --zone=dmz --add-interface=ens3  # Assign interface to zone
```

Common Services

- **http**: TCP port 80 (web server)
- **https**: TCP port 443 (secure web)
- **ssh**: TCP port 22 (secure shell)
- **ftp**: TCP port 21 (file transfer)
- **mysql**: TCP port 3306 (MySQL database)
- **dns**: TCP/UDP port 53 (domain name service)
- **smtp**: TCP port 25 (email)

Zone Security Levels (Most to Least Restrictive)

1. **drop**: Drop all incoming, allow outgoing
2. **block**: Reject all incoming with ICMP error
3. **dmz**: Limited services for DMZ servers
4. **public**: Default zone, limited services
5. **external**: For external networks with NAT
6. **work**: Work networks, more services
7. **home**: Home networks, even more services
8. **internal**: Internal networks, most services
9. **trusted**: Allow all traffic

Rich Rule Template

```
firewall-cmd --add-rich-rule='
rule family="ipv4"
     source address="192.168.1.0/24"
     service name="http"
     log prefix="HTTP-ACCESS" level="info"
     accept'
```

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What's the difference between runtime and permanent firewall configuration? **Answer**: Runtime configuration is active immediately but lost on service restart or reboot. Permanent configuration is saved to files and survives restarts but requires `--reload` to become active. Use `--permanent` flag to modify saved configuration.
2. **Question**: Why would you use rich rules instead of simple service or port rules? **Answer**: Rich rules allow complex conditions like source/destination restrictions, logging, rate limiting, and time-based rules. Use them when simple service/port rules aren't sufficient for your security requirements.
3. **Question**: How do firewall zones relate to network interfaces? **Answer**: Zones are security contexts applied to network interfaces. Each interface can be assigned to one zone, and all traffic through that interface follows the zone's rules. This allows different security policies for different network connections.

Practical Scenarios

1. **Scenario**: Web server needs HTTP access from internet but SSH only from management network. **Solution**:

   ```
   firewall-cmd --permanent --add-service=http
   firewall-cmd --permanent --remove-service=ssh
   firewall-cmd --permanent --add-rich-rule='rule family="ipv4" source
   address="192.168.100.0/24" service name="ssh" accept'
   ```

2. **Scenario**: Need to temporarily allow FTP access for maintenance without making it permanent. **Solution**: Use runtime-only configuration: `firewall-cmd --add-service=ftp` (without --permanent flag)

Command Challenges

1. **Challenge**: Create a zone that only allows HTTPS and SSH from specific network, logging all connections. **Answer**:

```
firewall-cmd --permanent --new-zone=secure-web
firewall-cmd --permanent --zone=secure-web --set-target=DROP
firewall-cmd --permanent --zone=secure-web --add-rich-rule='rule family="ipv4" source
address="trusted.network/24" service name="https" log prefix="HTTPS" accept'
firewall-cmd --permanent --zone=secure-web --add-rich-rule='rule family="ipv4" source
address="trusted.network/24" service name="ssh" log prefix="SSH" accept'
```

---

10. Exam Strategy

Topic-Specific Tips

- Always check both runtime and permanent configurations
- Use `--permanent` flag for persistent changes, then `--reload`
- Remember that first matching rule wins in rich rules
- Test connectivity after every firewall change

Common Exam Scenarios

1. **Scenario**: Configure web server with HTTP/HTTPS access **Approach**: Use standard service definitions with `--add-service=http` and `--add-service=https`
2. **Scenario**: Restrict SSH access to management network only **Approach**: Remove SSH service, add rich rule with source restriction
3. **Scenario**: Allow custom application port from specific sources **Approach**: Use rich rules with source and port specifications

Time Management

- **Basic service configuration**: 3-4 minutes including verification
- **Zone configuration**: 5-7 minutes for custom zones with rules
- **Rich rule implementation**: 6-8 minutes for complex rules
- **Always verify**: Test rules with query commands and connectivity tests

Pitfalls to Avoid

- Don't forget `--permanent` flag for persistent changes

- Remember to `--reload` after permanent changes
- Don't block SSH access without alternative access method
- Test connectivity before and after firewall changes
- Watch out for typos in rich rule syntax (they fail silently)

Summary

Key Takeaways

- **Firewalld is zone-based** - understand how zones work and use them effectively
- **Permanent vs runtime** - always use --permanent for lasting changes
- **Rich rules provide flexibility** - use them for complex access control requirements
- **Test everything** - firewall mistakes can lock you out of systems

Critical Commands to Remember

```
firewall-cmd --permanent --add-service=http     # Add service permanently
firewall-cmd --permanent --add-port=8080/tcp    # Add port permanently
firewall-cmd --reload                           # Apply permanent changes
firewall-cmd --list-all                         # Show current zone config
firewall-cmd --add-rich-rule='rule...'          # Add complex rule
firewall-cmd --zone=zone-name --change-interface=ens3  # Assign interface to zone
```

Next Steps

- Continue to Module 11: Boot Process & GRUB
- Practice firewall configuration in the Vagrant environment
- Review related topics: Network Configuration, SELinux Management

---

Navigation: ← SELinux Management | Index | Next → Boot Process & GRUB

2.13 Module 11: Boot Process & GRUB Configuration

1. Learning Objectives

- Understand the RHEL 10 boot process from UEFI/BIOS to systemd
- Configure and customize GRUB2 bootloader settings
- Manage kernel parameters and boot options
- Recover from boot failures using rescue modes
- Work with systemd targets and boot troubleshooting
- Implement emergency access and password recovery procedures

2. Key Concepts

Boot Process Overview

The RHEL 10 boot sequence follows these stages:

1. **UEFI/BIOS**: Hardware initialization and bootloader location
2. **GRUB2**: Boot menu, kernel selection, and parameter passing
3. **Kernel**: Hardware detection, driver loading, initramfs mounting
4. **systemd**: Service initialization and target reaching

GRUB2 Configuration Structure

- **Main config**: `/boot/grub2/grub.cfg` (auto-generated)
- **Default settings**: `/etc/default/grub`
- **Custom entries**: `/etc/grub.d/` directory
- **EFI systems**: `/boot/efi/EFI/redhat/grub.cfg`

Systemd Targets

- **graphical.target**: Full multi-user with GUI
- **multi-user.target**: Multi-user without GUI
- **rescue.target**: Single-user maintenance mode
- **emergency.target**: Minimal environment with read-only root

3. Essential Commands

GRUB Management

```
# Regenerate GRUB configuration
grub2-mkconfig -o /boot/grub2/grub.cfg                # BIOS systems
grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg       # UEFI systems

# Install GRUB to disk
grub2-install /dev/sda                                # BIOS systems
grub2-install --target=x86_64-efi --efi-directory=/boot/efi  # UEFI

# Set default boot entry
grub2-set-default 0                                   # Set first entry as default
grub2-editenv list                                    # Show current default
```

Kernel Parameter Management

```
# Temporary kernel parameters (current boot only)
# Edit in GRUB menu: press 'e', modify linux line, press Ctrl+x

# Permanent kernel parameters
grubby --update-kernel=ALL --args="parameter=value"   # Add parameter
grubby --update-kernel=ALL --remove-args="parameter"  # Remove parameter
grubby --info=ALL                                     # List all kernels and parameters

# View current kernel command line
cat /proc/cmdline
```

Boot Target Management

```
# Get current target
systemctl get-default

# Set default target
systemctl set-default multi-user.target
systemctl set-default graphical.target

# Switch to target (temporary)
systemctl isolate rescue.target
systemctl isolate emergency.target

# Boot to specific target (from GRUB)
# Add: systemd.unit=rescue.target
```

Recovery Procedures

```
# Reset root password (from rescue mode)
mount -o remount,rw /sysroot
chroot /sysroot
passwd root
touch /.autorelabel                                    # For SELinux systems
exit
reboot

# Boot with init=/bin/bash
# Add to kernel line: init=/bin/bash
mount -o remount,rw /
passwd root
mount -o remount,ro /
reboot
```

4. Asghar Ghori's Approach

Boot Process Analysis

Ghori emphasizes understanding each boot stage through observation:

```
# Analyze boot messages
dmesg | less
journalctl -b                                          # Current boot messages
journalctl --list-boots                                # Available boot logs
journalctl -b -1                                       # Previous boot messages
```

GRUB Customization Method

```
# Modify /etc/default/grub
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="crashkernel=auto rd.lvm.lv=rhel/root rd.lvm.lv=rhel/swap rhgb quiet"

# Apply changes
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Rescue Mode Procedure

Ghori's systematic approach to rescue scenarios:

1. Boot from installation media
2. Select "Troubleshooting" → "Rescue a Red Hat Enterprise Linux system"
3. Choose shell option for full system access
4. Mount filesystems and chroot into system
5. Perform repairs and regenerate GRUB if needed

5. Sander van Vugt's Approach

Bootloader Troubleshooting Methodology

Van Vugt focuses on systematic GRUB repair procedures:

```
# Complete GRUB reinstallation procedure
# Boot from live/rescue media
mkdir /mnt/sysimage
mount /dev/sda2 /mnt/sysimage                          # Mount root partition
mount /dev/sda1 /mnt/sysimage/boot                     # Mount boot partition
mount --bind /dev /mnt/sysimage/dev
mount --bind /proc /mnt/sysimage/proc
mount --bind /sys /mnt/sysimage/sys
chroot /mnt/sysimage
grub2-install /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Advanced Kernel Parameter Management

```
# Comprehensive grubby usage
grubby --default-kernel                                # Show default kernel
grubby --set-default-index=1                           # Set specific kernel index
grubby --add-kernel=/boot/vmlinuz-new --title="New Kernel" --initrd=/boot/initramfs-new.img
grubby --remove-kernel=/boot/vmlinuz-old
```

Systemd Boot Analysis

Van Vugt's approach to boot performance analysis:

```
# Boot time analysis
systemd-analyze                                    # Overall boot time
systemd-analyze blame                              # Service startup times
systemd-analyze critical-chain                     # Critical path analysis
systemd-analyze plot > bootchart.svg               # Visual boot chart
```

6. Command Examples and Scenarios

Scenario 1: Kernel Parameter Configuration

```
# Add kernel parameter for debugging
grubby --update-kernel=ALL --args="debug"
grubby --info=ALL | grep -A5 -B5 debug

# Remove quiet and rhgb for verbose boot
grubby --update-kernel=ALL --remove-args="quiet rhgb"

# Add custom memory settings
grubby --update-kernel=ALL --args="mem=2G"
```

Scenario 2: GRUB Menu Customization

```
# Extend GRUB timeout
sed -i 's/GRUB_TIMEOUT=5/GRUB_TIMEOUT=15/' /etc/default/grub
grub2-mkconfig -o /boot/grub2/grub.cfg

# Disable GRUB submenu
echo 'GRUB_DISABLE_SUBMENU=true' >> /etc/default/grub
grub2-mkconfig -o /boot/grub2/grub.cfg

# Add custom menu entry
cat > /etc/grub.d/40_custom << 'EOF'
#!/bin/sh
exec tail -n +3 $0
menuentry 'Memory Test' {
    linux16 /memtest86+
}
EOF
chmod +x /etc/grub.d/40_custom
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Scenario 3: Boot Target Management

```
# Switch to text mode permanently
systemctl set-default multi-user.target
systemctl get-default                                  # Verify change

# Temporary switch to rescue mode
systemctl isolate rescue.target

# Emergency boot troubleshooting
# At GRUB menu, press 'e' and add:
systemd.unit=emergency.target

# Boot with specific runlevel (legacy)
systemd.unit=runlevel3.target                         # Equivalent to multi-user
```

7. Lab Exercises

Lab 11A: GRUB Configuration and Kernel Parameters (Ghori-focused)

Time Limit: 20 minutes **Objective**: Configure GRUB bootloader and manage kernel parameters

Prerequisites:

- RHEL 10 system with multiple kernel versions
- Root access for bootloader modifications

Tasks:

1. Modify GRUB timeout to 15 seconds and disable submenu
2. Add kernel parameter `console=ttyS0,115200` to all kernels
3. Create custom GRUB menu entry for memory test
4. Remove `quiet` parameter from current kernel
5. Set the second kernel as default boot option

Verification Commands:

```
grep GRUB_TIMEOUT /etc/default/grub                   # Check timeout setting
grubby --info=ALL | grep console                      # Verify console parameter
grub2-editenv list                                    # Check default kernel
cat /proc/cmdline                                     # Verify current parameters
```

Lab 11B: Boot Troubleshooting and Recovery (van Vugt-focused)

Time Limit: 25 minutes **Objective**: Practice boot failure recovery procedures

Prerequisites:

- RHEL 10 system with intentionally broken boot configuration
- Installation media or rescue disk available

Tasks:

1. Simulate GRUB corruption by removing `/boot/grub2/grub.cfg`
2. Boot into rescue mode and reinstall GRUB
3. Change root password using emergency boot mode
4. Configure system to boot to multi-user target by default
5. Analyze boot performance and identify slowest service

Verification Commands:

```
ls -la /boot/grub2/grub.cfg                          # Verify GRUB config exists
systemctl get-default                                 # Check default target
systemd-analyze blame | head -5                      # Show slowest services
journalctl -b | grep -i error                        # Check for boot errors
```

Lab 11C: Synthesis Challenge - Complete Boot Environment Setup

Time Limit: 30 minutes **Objective**: Integrate both methodologies for comprehensive boot management

Prerequisites:

- Fresh RHEL 10 installation
- Multiple kernel versions installed
- Access to rescue media

Tasks:

1. Configure GRUB with custom splash image and 20-second timeout
2. Add persistent kernel parameters for debugging and console redirection
3. Create custom rescue menu entry that boots directly to single-user mode
4. Set up automatic boot to graphical target with fallback to multi-user
5. Implement password protection for GRUB menu editing
6. Document complete recovery procedure for boot failure scenarios

Advanced Requirements:

- Use both grubby and manual GRUB configuration methods
- Combine Ghori's systematic approach with van Vugt's advanced troubleshooting
- Create comprehensive boot analysis report using systemd tools

Verification Commands:

```
grub2-editenv list                                    # Verify default settings
grubby --info=ALL                                     # Check all kernel parameters
systemd-analyze critical-chain                       # Analyze boot dependencies
journalctl -b --no-pager | grep -E "(Started|Failed)" # Boot service status
```

8. Troubleshooting Common Issues

GRUB Not Loading

```
# Symptoms: System boots directly to BIOS/UEFI
# Solution: Reinstall GRUB to MBR/ESP

# For BIOS systems:
grub2-install /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg

# For UEFI systems:
grub2-install --target=x86_64-efi --efi-directory=/boot/efi
grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
```

Kernel Panic on Boot

```
# Symptoms: Kernel panic, unable to mount root filesystem
# Solution: Boot with different kernel or rescue mode

# From GRUB menu:
# 1. Select older kernel version
# 2. Add kernel parameter: init=/bin/bash
# 3. Boot to rescue mode from installation media
```

Wrong systemd Target

```
# Symptoms: System boots to wrong runlevel/target
# Solution: Check and correct default target

systemctl get-default
systemctl set-default graphical.target
systemctl list-units --type=target --state=active    # Show active targets
```

GRUB Configuration Corruption

```
# Symptoms: Syntax errors, missing menu entries
# Solution: Regenerate configuration

# Backup existing configuration
cp /boot/grub2/grub.cfg /boot/grub2/grub.cfg.backup

# Check /etc/default/grub for syntax errors
cat /etc/default/grub

# Regenerate clean configuration
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Forgotten Root Password Recovery

```
# Method 1: rd.break method
# Add to kernel line: rd.break
mount -o remount,rw /sysroot
chroot /sysroot
passwd root
touch /.autorelabel
exit
reboot

# Method 2: init=/bin/bash method
# Add to kernel line: init=/bin/bash
mount -o remount,rw /
passwd root
exec /sbin/init
```

9. Best Practices

GRUB Configuration Management

- Always backup `/boot/grub2/grub.cfg` before changes
- Use `/etc/default/grub` for global settings
- Place custom entries in `/etc/grub.d/40_custom`
- Test boot changes immediately after implementation
- Keep rescue media available for emergency recovery

Kernel Parameter Management

- Use `grubby` for persistent kernel parameter changes
- Document all custom parameters and their purposes
- Test parameter changes before making them permanent
- Monitor system behavior after parameter modifications
- Maintain list of working parameter combinations

Boot Security

- Implement GRUB password protection for menu editing
- Secure physical access to prevent boot parameter tampering
- Use encrypted boot partitions for sensitive environments
- Regularly update bootloader for security patches
- Monitor boot logs for unauthorized access attempts

Recovery Preparedness

- Create and test rescue media regularly
- Document complete recovery procedures
- Practice password recovery methods
- Maintain emergency contact information
- Keep system documentation current and accessible

10. Integration with Other RHCSA Topics

Storage Integration

- Boot partition requirements for LVM root filesystems
- GRUB configuration for encrypted root partitions
- Rescue procedures for storage failures
- Boot from different storage devices

Security Integration

- SELinux autorelabel during password recovery
- Boot security with GRUB passwords
- Secure boot configuration in UEFI environments
- Audit trail for boot-time security events

Network Integration

- Network boot with PXE and GRUB
- Console redirection for remote management
- Boot parameter configuration for network interfaces
- Remote boot troubleshooting procedures

Service Integration

- systemd target dependencies and boot order
- Service startup optimization for faster boot
- Boot-time service failure troubleshooting
- Integration with monitoring systems for boot alerts

Module 11 Summary: Boot process management and GRUB configuration are critical skills for system recovery and optimization. This module combines systematic troubleshooting approaches with practical recovery procedures, ensuring administrators can handle boot failures and customize the boot environment effectively. Understanding both the theory of the boot process and hands-on recovery techniques is essential for RHCSA certification and real-world system administration.

2.14 Module 12: Logging & Monitoring

1. Learning Objectives

- Master systemd journald and traditional syslog logging systems
- Configure rsyslog for centralized log management
- Monitor system performance using built-in RHEL tools
- Set up log rotation and retention policies
- Implement log filtering and analysis techniques
- Configure system monitoring alerts and notifications
- Troubleshoot system issues using log analysis

2. Key Concepts

Logging Architecture in RHEL 10

- **systemd-journald**: Primary logging daemon for systemd services
- **rsyslog**: Traditional syslog daemon for compatibility and advanced features
- **Log Storage**: Binary journal files and text-based syslog files
- **Log Forwarding**: Integration between journald and rsyslog

Journal vs Syslog

- **Journal**: Binary format, structured metadata, automatic rotation
- **Syslog**: Text format, traditional facilities/priorities, manual rotation
- **Integration**: journald forwards to rsyslog for persistent storage

Log Facilities and Priorities

```
Facilities: kern, user, mail, daemon, auth, syslog, lpr, news, uucp, cron, authpriv, ftp,
local0-7
Priorities: emerg, alert, crit, err, warning, notice, info, debug
```

System Monitoring Tools

- **top/htop**: Real-time process monitoring
- **vmstat**: Virtual memory statistics
- **iostat**: I/O statistics
- **sar**: System activity reporter
- **netstat/ss**: Network statistics

3. Essential Commands

Journal Management

```
# View journal logs
journalctl                                          # All logs
journalctl -b                                       # Current boot
journalctl -b -1                                    # Previous boot
journalctl -f                                       # Follow (tail -f equivalent)
journalctl -u sshd                                  # Specific unit
journalctl -p err                                   # By priority level

# Time-based filtering
journalctl --since "2023-01-01 12:00:00"
journalctl --since yesterday
journalctl --since "1 hour ago"
journalctl --until "2023-12-31"

# Advanced filtering
journalctl -u httpd -p warning --since today
journalctl _PID=1234                                # By process ID
journalctl _UID=1000                                # By user ID
journalctl -k                                       # Kernel messages only
```

Journal Configuration

```
# Journal persistence configuration
mkdir -p /var/log/journal
systemctl restart systemd-journald

# Journal storage limits
# Edit /etc/systemd/journald.conf:
SystemMaxUse=1G                                      # Maximum disk usage
SystemKeepFree=500M                                  # Minimum free space
MaxRetentionSec=1month                              # Maximum retention time

# Verify journal statistics
journalctl --disk-usage                             # Current disk usage
journalctl --vacuum-time=1week                      # Clean old entries
journalctl --vacuum-size=100M                       # Limit size
```

Rsyslog Configuration

```
# Main configuration file: /etc/rsyslog.conf
# Additional configs: /etc/rsyslog.d/*.conf

# Basic rsyslog rules syntax:
# facility.priority    action
*.info;mail.none;authpriv.none;cron.none    /var/log/messages
authpriv.*                                  /var/log/secure
mail.*                                      /var/log/maillog
cron.*                                      /var/log/cron

# Restart rsyslog after configuration changes
systemctl restart rsyslog
```

Log Rotation Management

```
# Logrotate configuration
/etc/logrotate.conf                                 # Main config
/etc/logrotate.d/                                   # Service-specific configs

# Manual logrotate execution
logrotate -f /etc/logrotate.conf                    # Force rotation
logrotate -d /etc/logrotate.conf                    # Debug/dry-run
logrotate -v /etc/logrotate.conf                    # Verbose output

# Example logrotate configuration
cat > /etc/logrotate.d/custom << 'EOF'
/var/log/custom/*.log {
    daily
    rotate 30
    compress
    delaycompress
    missingok
    notifempty
    create 0644 root root
}
EOF
```

System Monitoring Commands

```
# Process and memory monitoring
top                                                 # Real-time process viewer
htop                                                # Enhanced process viewer
ps aux                                              # Process snapshot
free -h                                             # Memory usage
uptime                                              # System uptime and load

# I/O and disk monitoring
iostat -x 1                                         # I/O statistics
iotop                                               # I/O by process
df -h                                               # Disk usage
du -sh /path/*                                      # Directory sizes

# Network monitoring
ss -tulnp                                           # Socket statistics
netstat -i                                          # Network interface statistics
iftop                                               # Network traffic by connection
```

4. Asghar Ghori's Approach

Systematic Log Analysis Method

Ghori emphasizes structured log examination:

```
# Step-by-step log analysis workflow
# 1. Identify the time frame
journalctl --since "10 minutes ago" --until now

# 2. Filter by service or component
journalctl -u httpd --since "1 hour ago"

# 3. Focus on error levels
journalctl -p err --since today

# 4. Extract specific patterns
journalctl | grep -i "error\|fail\|denied"

# 5. Correlate with system events
journalctl -k --since "30 minutes ago"              # Kernel messages
```

Rsyslog Centralization Setup

Ghori's approach to centralized logging:

```
# Server configuration (/etc/rsyslog.conf)
$ModLoad imudp
$UDPServerRun 514
$AllowedSender UDP, 192.168.1.0/24

# Client configuration
*.* @192.168.1.100:514                              # UDP forwarding
*.* @@192.168.1.100:514                             # TCP forwarding

# Restart services
systemctl restart rsyslog
firewall-cmd --add-port=514/udp --permanent
firewall-cmd --reload
```

Performance Monitoring Workflow

```
# Ghori's systematic performance analysis
# 1. Overall system health
uptime && free -h && df -h

# 2. Process analysis
top -b -n1 | head -20
ps aux --sort=-%cpu | head -10
ps aux --sort=-%mem | head -10

# 3. I/O analysis
iostat -x 1 3
sar -d 1 5

# 4. Network analysis
ss -s                                                # Socket summary
netstat -i                                          # Interface statistics
```

5. Sander van Vugt's Approach

Advanced Journal Queries

Van Vugt focuses on sophisticated filtering techniques:

```
# Complex journal queries using field matching
journalctl _COMM=sshd _PID=1234                      # Multiple field filters
journalctl _SYSTEMD_UNIT=httpd.service + _SYSTEMD_UNIT=nginx.service  # OR logic
journalctl PRIORITY=3 _TRANSPORT=kernel              # Critical kernel messages

# JSON output for scripting
journalctl -o json --since today | jq '._PID'        # Extract PIDs with jq
journalctl -o json-pretty -n 5                       # Pretty JSON format

# Field discovery
journalctl -N                                         # List all field names
journalctl -F _SYSTEMD_UNIT                          # List all units
journalctl -F PRIORITY                               # List priority values
```

Rsyslog Advanced Configuration

Van Vugt's sophisticated rsyslog setup:

```
# Template-based logging
# Add to /etc/rsyslog.conf:
$template CustomFormat,"%timegenerated% %HOSTNAME% %syslogtag% %msg%\n"
*.info;mail.none;authpriv.none    /var/log/messages;CustomFormat

# Property-based filtering
:programname, isequal, "sshd"    /var/log/ssh.log
:msg, contains, "error"          /var/log/errors.log
:msg, regex, "Failed.*from"      /var/log/failed_logins.log

# Rate limiting
$SystemLogRateLimitInterval 0                        # Disable rate limiting
$ModLoad imjournal                                   # Load journal module
$IMJournalStateFile imjournal.state                 # State file location
```

SAR-based Long-term Monitoring

```
# Configure SAR data collection
# Edit /etc/sysconfig/sysstat
HISTORY=60                                          # Keep 60 days of data

# Manual SAR commands
sar -u 1 10                                         # CPU usage, 10 samples
sar -r 1 5                                          # Memory usage
sar -d 1 3                                          # Disk I/O
sar -n DEV 1 5                                      # Network statistics

# Historical analysis
sar -u -f /var/log/sa/sa15                         # CPU data from 15th
sar -A -s 09:00:00 -e 17:00:00                     # All stats, time range
```

6. Command Examples and Scenarios

Scenario 1: Troubleshooting Service Failures

```
# Service failed to start - comprehensive analysis
systemctl status httpd                              # Service status
journalctl -u httpd --since "1 hour ago" -p err     # Recent errors
journalctl -xeu httpd                               # Detailed failure info

# Check configuration and permissions
httpd -t                                            # Test configuration
ls -la /etc/httpd/conf/                             # Check file permissions
semanage port -l | grep http                       # Check SELinux ports
```

Scenario 2: Performance Investigation

```
# System running slowly - systematic analysis
# 1. Quick overview
uptime                                              # Load averages
free -h                                             # Memory usage
df -h                                               # Disk space

# 2. Process analysis
top -b -n1 -o %CPU | head -15                      # CPU-intensive processes
ps aux --sort=-%mem | head -10                     # Memory-intensive processes

# 3. I/O analysis
iostat -x 1 5                                      # I/O wait times
iotop -ao                                           # I/O by process

# 4. Log correlation
journalctl --since "30 minutes ago" -p warning      # Recent warnings
dmesg | tail -20                                    # Recent kernel messages
```

Scenario 3: Security Event Analysis

```
# Investigating failed login attempts
journalctl -u sshd | grep "Failed password"         # SSH failures
grep "Failed password" /var/log/secure | tail -20   # Recent failures
lastb | head -10                                     # Bad logins

# Authentication analysis
journalctl _COMM=sudo --since today                 # Sudo usage
grep "authentication failure" /var/log/secure       # Auth failures
aureport --auth --summary                           # SELinux auth summary
```

7. Lab Exercises

Lab 12A: Journal and Rsyslog Configuration (Ghori-focused)

Time Limit: 25 minutes **Objective**: Configure comprehensive logging system with journal persistence and rsyslog customization

Prerequisites:

- RHEL 10 system with systemd and rsyslog installed
- Root access for configuration modifications

Tasks:

1. Configure journal persistence with 2GB maximum usage
2. Set up rsyslog to separate SSH logs to `/var/log/ssh.log`
3. Configure log rotation for SSH logs (daily, keep 30 days)
4. Create custom rsyslog template with hostname and timestamp
5. Forward all critical messages to remote server (simulated)

Verification Commands:

```
ls -la /var/log/journal/                             # Check journal persistence
grep -i ssh /etc/rsyslog.conf                        # Verify SSH logging config
tail -10 /var/log/ssh.log                            # Check SSH log file
cat /etc/logrotate.d/ssh                             # Check rotation config
```

Lab 12B: System Monitoring and Analysis (van Vugt-focused)

Time Limit: 30 minutes **Objective**: Implement comprehensive system monitoring using built-in tools

Prerequisites:

- RHEL 10 system with full monitoring tools installed
- Network connectivity for remote logging tests

Tasks:

1. Configure SAR to collect data every 2 minutes
2. Analyze system performance during high load simulation
3. Set up advanced journal queries to identify security events
4. Create monitoring script that alerts on high CPU usage
5. Generate performance report covering 24-hour period

Verification Commands:

```
crontab -l | grep sa                                    # Check SAR cron job
sar -u 1 3                                             # Test SAR functionality
journalctl -F _COMM | wc -l                            # Count unique commands
ls -la /var/log/sa/                                     # Check SAR data files
```

Lab 12C: Synthesis Challenge - Complete Logging Infrastructure

Time Limit: 35 minutes **Objective**: Build enterprise-grade logging and monitoring system

Prerequisites:

- Multiple RHEL 10 systems (or containers) for centralized logging
- Administrative access to all systems

Tasks:

1. Set up centralized rsyslog server with client forwarding
2. Configure journal with structured logging for application troubleshooting
3. Implement automated log analysis with alerting mechanisms
4. Create comprehensive monitoring dashboard using system tools
5. Design log retention policy balancing storage and compliance needs
6. Document incident response procedures using log analysis

Advanced Requirements:

- Combine both Ghori's systematic approach and van Vugt's advanced techniques
- Implement security-focused logging with audit integration
- Create automated scripts for common troubleshooting scenarios

Verification Commands:

```
ss -tulnp | grep :514                                # Check rsyslog server
journalctl --disk-usage                              # Check journal usage
grep "@@.*:514" /etc/rsyslog.conf                    # Verify log forwarding
systemctl status rsyslog systemd-journald            # Check service status
```

8. Troubleshooting Common Issues

Journal Not Persisting

```
# Symptoms: Logs lost after reboot
# Solution: Enable persistent journal storage

# Create journal directory
mkdir -p /var/log/journal
chown root:systemd-journal /var/log/journal
chmod 2755 /var/log/journal

# Configure persistence in /etc/systemd/journald.conf
Storage=persistent
SystemMaxUse=1G

# Restart journald
systemctl restart systemd-journald
```

High Log Volume Consuming Disk Space

```
# Symptoms: Logs filling up filesystem
# Solutions: Implement proper rotation and retention

# Emergency cleanup
journalctl --vacuum-time=1week
journalctl --vacuum-size=500M

# Configure journal limits in /etc/systemd/journald.conf
SystemMaxUse=2G
SystemKeepFree=1G
RuntimeMaxUse=200M

# Check logrotate configuration
logrotate -d /etc/logrotate.conf | grep -A5 -B5 error
```

Rsyslog Not Receiving Remote Logs

```
# Symptoms: Central log server not receiving client logs
# Solution: Check network and configuration

# On server - check listening ports
ss -ulnp | grep :514
netstat -ulnp | grep :514

# Check firewall
firewall-cmd --list-services --permanent
firewall-cmd --add-service=syslog --permanent
firewall-cmd --reload

# Test connectivity from client
telnet logserver 514
logger -n logserver "Test message from client"
```

Missing Log Entries

```
# Symptoms: Expected log entries not appearing
# Solution: Check service status and configuration

# Verify logging services
systemctl status systemd-journald rsyslog

# Check journal integration
grep -i "imjournal" /etc/rsyslog.conf
grep -i "ForwardToSyslog" /etc/systemd/journald.conf

# Test logging
logger "Test message"
journalctl -f &                                    # Monitor in background
logger "Another test message"
```

Performance Impact from Logging

```
# Symptoms: System slowdown due to excessive logging
# Solution: Optimize logging configuration

# Check I/O impact
iostat -x 1 5                                      # Monitor disk I/O
iotop | grep -E "(rsyslog|journal)"                # Check logging processes

# Optimize journal sync settings in /etc/systemd/journald.conf
SyncIntervalSec=60                                 # Reduce sync frequency
Storage=volatile                                   # Use memory storage temporarily

# Implement log filtering
# In /etc/rsyslog.conf:
:msg, regex, ".*verbose debug.*" stop             # Filter verbose messages
```

9. Best Practices

Log Management Strategy

- Implement centralized logging for multi-server environments
- Configure appropriate retention policies based on compliance requirements
- Use structured logging formats for easier analysis
- Separate application logs from system logs
- Monitor log growth and implement automated cleanup

Performance Optimization

- Balance between log detail and system performance

- Use asynchronous logging for high-volume applications
- Configure appropriate buffer sizes for network log forwarding
- Monitor I/O impact of logging operations
- Implement log compression for long-term storage

Security Considerations

- Protect log files with appropriate permissions (640 or 644)
- Implement log integrity checking for critical systems
- Use encrypted connections for remote log forwarding
- Separate security logs from application logs
- Regular security log analysis and alerting

Monitoring Best Practices

- Establish baseline performance metrics
- Set up automated alerting for critical thresholds
- Document normal system behavior patterns
- Create runbooks for common performance issues
- Regular review and analysis of monitoring data

10. Integration with Other RHCSA Topics

Security Integration

- Correlate SELinux denials with application errors
- Monitor authentication and authorization events
- Track file permission changes and access attempts
- Integrate with audit subsystem for compliance logging

Network Integration

- Monitor network service performance and errors
- Track connection attempts and failures
- Correlate network issues with system performance
- Monitor firewall rule effectiveness through logs

Storage Integration

- Monitor filesystem usage and I/O performance

- Track LVM operations and storage events
- Correlate storage errors with application failures
- Monitor backup and restore operations

Service Integration

- Monitor systemd service dependencies and failures
- Track service startup and shutdown times
- Correlate service errors with system events
- Monitor resource usage by services

Module 12 Summary: Effective logging and monitoring are essential for maintaining system health and security. This module combines traditional syslog management with modern systemd journal capabilities, providing comprehensive coverage of RHEL 10 logging infrastructure. Understanding both reactive troubleshooting through log analysis and proactive monitoring for performance optimization is crucial for RHCSA certification and production system management.

2.15 Module 13: Scheduled Tasks & Automation

1. Learning Objectives

- Master cron and anacron scheduling systems
- Configure systemd timers for service automation
- Implement at and batch commands for one-time tasks
- Manage user access to scheduling systems
- Automate system maintenance tasks
- Monitor and troubleshoot scheduled tasks
- Design efficient automation strategies for system administration

2. Key Concepts

Task Scheduling Systems in RHEL 10

- **cron**: Traditional time-based job scheduler
- **anacron**: Enhanced scheduler for systems not always running
- **systemd timers**: Modern systemd-based scheduling
- **at**: One-time task execution
- **batch**: Queue-based task execution

Cron Architecture

- **crond**: Main cron daemon
- **User crontabs**: Individual user scheduling
- **System crontab**: `/etc/crontab` for system-wide tasks
- **Cron directories**: `/etc/cron.{hourly,daily,weekly,monthly}/`

Systemd Timer Types

- **Realtime timers**: Calendar-based scheduling (like cron)
- **Monotonic timers**: Relative to system events (boot, service start)
- **Transient timers**: Temporary timers created on-the-fly

Access Control

- **Allow files**: `/etc/cron.allow`, `/etc/at.allow`
- **Deny files**: `/etc/cron.deny`, `/etc/at.deny`
- **Default behavior**: If no allow file exists, all users except those in deny file can schedule tasks

3. Essential Commands

Cron Management

```
# User crontab management
crontab -e                                          # Edit current user's crontab
crontab -l                                          # List current user's crontab
crontab -r                                          # Remove current user's crontab
crontab -u username -e                              # Edit another user's crontab (root only)

# System crontab files
/etc/crontab                                        # System-wide crontab
/etc/cron.d/                                        # Additional system cron files
/var/spool/cron/                                    # User crontab storage

# Cron service management
systemctl status crond
systemctl enable --now crond
systemctl restart crond
```

Crontab Syntax

```
# Format: minute hour day_of_month month day_of_week command
# Fields:    0-59  0-23    1-31        1-12      0-7 (0 and 7 are Sunday)

# Special characters:
# *     - any value
# ,     - list separator (1,3,5)
# -     - range (1-5)
# /     - step values (*/5 = every 5)
# @     - special strings (@reboot, @daily, @hourly, @weekly, @monthly, @annually)

# Examples:
0 2 * * *     /usr/local/bin/backup.sh             # Daily at 2:00 AM
30 14 * * 1   /usr/bin/update-system               # Mondays at 2:30 PM
0 */6 * * *   /usr/bin/system-check                # Every 6 hours
@daily        /usr/local/bin/cleanup.sh            # Once per day
@reboot       /usr/local/bin/startup.sh            # At system boot
```

Systemd Timer Management

```
# List all timers
systemctl list-timers                                  # Active timers
systemctl list-timers --all                           # All timers (active and inactive)

# Timer control
systemctl enable timer-name.timer
systemctl start timer-name.timer
systemctl status timer-name.timer
systemctl stop timer-name.timer

# View timer logs
journalctl -u timer-name.timer
journalctl -u timer-name.service
```

At and Batch Commands

```
# Schedule one-time tasks with at
at 15:30                                               # Run at 3:30 PM today
at now + 2 hours                                       # Run in 2 hours
at 2:30 PM tomorrow                                    # Run at 2:30 PM tomorrow
at -f script.sh now + 5 minutes                        # Run script in 5 minutes

# At command interface
at> command to run
at> <Ctrl+D>                                           # End input

# Manage at jobs
atq                                                    # List queued at jobs
atrm job_number                                        # Remove at job
at -c job_number                                       # Show job details

# Batch command (runs when system load permits)
batch
batch> command to run
batch> <Ctrl+D>
```

Access Control Management

```
# Cron access control
echo "username" >> /etc/cron.allow                  # Allow user
echo "username" >> /etc/cron.deny                   # Deny user
ls -la /etc/cron.{allow,deny}                       # Check existing files

# At access control
echo "username" >> /etc/at.allow                    # Allow user
echo "username" >> /etc/at.deny                     # Deny user

# Default permissions (if no allow file exists):
# - All users except root can use cron/at
# - Users listed in deny file cannot use cron/at
```

4. Asghar Ghori's Approach

Systematic Cron Implementation

Ghori emphasizes step-by-step cron configuration:

```
# 1. Plan the task schedule
# Identify task frequency and timing requirements
# Document task dependencies and prerequisites

# 2. Create and test the script
cat > /usr/local/bin/system-backup.sh << 'EOF'
#!/bin/bash
# System backup script
BACKUP_DIR="/backup/$(date +%Y%m%d)"
mkdir -p "$BACKUP_DIR"
tar -czf "$BACKUP_DIR/system-backup.tar.gz" /etc /home /var/log
echo "Backup completed at $(date)" >> /var/log/backup.log
EOF

chmod +x /usr/local/bin/system-backup.sh

# 3. Test script manually
/usr/local/bin/system-backup.sh

# 4. Add to crontab with logging
crontab -e
# Add: 0 3 * * * /usr/local/bin/system-backup.sh >> /var/log/cron-backup.log 2>&1
```

Anacron Configuration for Laptops

Ghori's approach for systems not always running:

```
# Configure anacron in /etc/anacrontab
# period_in_days delay_in_minutes job-identifier command

# Example anacron entries:
1    10    backup.daily      /usr/local/bin/daily-backup.sh
7    20    backup.weekly     /usr/local/bin/weekly-backup.sh
30   30    backup.monthly    /usr/local/bin/monthly-backup.sh

# Anacron execution
anacron -f                                          # Force run all jobs
anacron -T                                          # Test configuration
```

Cron Security Best Practices

```
# Ghori's security recommendations:
# 1. Use full paths in cron scripts
# 2. Set PATH variable in crontab
# 3. Redirect output for debugging
# 4. Use dedicated service accounts

# Example secure crontab entry:
PATH=/usr/local/bin:/usr/bin:/bin
MAILTO=admin@company.com
0 2 * * * /usr/local/bin/secure-backup.sh >> /var/log/backup.log 2>&1
```

5. Sander van Vugt's Approach

Systemd Timer Implementation

Van Vugt emphasizes modern systemd timers over traditional cron:

```
# Create service unit file
cat > /etc/systemd/system/system-cleanup.service << 'EOF'
[Unit]
Description=Daily system cleanup
Wants=system-cleanup.timer

[Service]
Type=oneshot
ExecStart=/usr/local/bin/cleanup.sh
User=cleanup
Group=cleanup

[Install]
WantedBy=multi-user.target
EOF

# Create timer unit file
cat > /etc/systemd/system/system-cleanup.timer << 'EOF'
[Unit]
Description=Daily system cleanup timer
Requires=system-cleanup.service

[Timer]
OnCalendar=daily
Persistent=true
RandomizedDelaySec=1800

[Install]
WantedBy=timers.target
EOF

# Enable and start timer
systemctl daemon-reload
systemctl enable --now system-cleanup.timer
```

Advanced Timer Scheduling

Van Vugt's sophisticated timer configurations:

```
# Complex calendar specifications
OnCalendar=Mon,Tue,Wed,Thu,Fri *-*-* 02:00:00      # Weekdays at 2 AM
OnCalendar=*-*-* 00/3:00:00                        # Every 3 hours
OnCalendar=monthly                                 # First day of each month
OnCalendar=weekly                                  # Every Monday
OnCalendar=*-01,04,07,10-01 12:00:00              # Quarterly

# Monotonic timers
OnBootSec=15min                                    # 15 minutes after boot
OnStartupSec=1h                                    # 1 hour after systemd start
OnUnitActiveSec=2w                                 # 2 weeks after service activation

# Combined timer example
[Timer]
OnBootSec=10min
OnUnitActiveSec=1h
Persistent=true
```

Timer Debugging and Analysis

```
# Van Vugt's timer troubleshooting approach
systemctl list-timers --all                        # Show all timer status
systemctl cat timer-name.timer                     # Show timer configuration
systemd-analyze calendar "Mon *-*-* 02:00:00"      # Validate calendar syntax

# Detailed timer analysis
systemctl show timer-name.timer                    # All timer properties
journalctl -u timer-name.timer -f                  # Follow timer logs
systemctl status timer-name.timer timer-name.service  # Combined status
```

6. Command Examples and Scenarios

Scenario 1: System Maintenance Automation

```
# Comprehensive system maintenance crontab
# Edit: crontab -e

# Daily tasks
0 1 * * * /usr/sbin/updatedb                    # Update locate database
30 1 * * * /usr/bin/find /tmp -type f -atime +7 -delete  # Clean old temp files
0 2 * * * /usr/local/bin/backup-logs.sh         # Backup log files

# Weekly tasks
0 3 * * 0 /usr/bin/yum clean all                # Clean package cache (Sunday)
30 3 * * 0 /usr/bin/package-security-update.sh  # Security updates

# Monthly tasks
0 4 1 * * /usr/local/bin/rotate-archives.sh     # Archive rotation
0 5 1 * * /usr/bin/find /var/log -name "*.gz" -mtime +365 -delete  # Old log cleanup
```

Scenario 2: User-specific Task Scheduling

```
# User crontab for development environment
# Run as regular user: crontab -e

# Environment variables
PATH=/usr/local/bin:/usr/bin:/bin
SHELL=/bin/bash
MAILTO=developer@company.com

# Development tasks
0 9 * * 1-5 /home/user/bin/project-sync.sh      # Workday morning sync
*/15 * * * * /home/user/bin/check-services.sh   # Every 15 minutes
0 18 * * * /home/user/bin/daily-commit.sh       # End of day commit
0 0 * * 6 /home/user/bin/weekly-backup.sh       # Saturday midnight backup
```

Scenario 3: One-time and Conditional Tasks

```
# Schedule immediate one-time tasks
echo "systemctl restart httpd" | at now + 5 minutes
echo "/usr/local/bin/maintenance.sh" | at 02:00 tomorrow

# Batch processing for resource-intensive tasks
echo "/usr/local/bin/generate-reports.sh" | batch
echo "/usr/local/bin/video-processing.sh" | batch

# Conditional execution in cron
*/5 * * * * [ $(uptime | cut -d',' -f4 | cut -d':' -f2 | cut -d' ' -f2) -lt 2.0 ] && /usr/local/
bin/backup.sh
```

7. Lab Exercises

Lab 13A: Cron and Anacron Configuration (Ghori-focused)

Time Limit: 25 minutes **Objective**: Implement comprehensive cron-based task scheduling with proper security and logging

Prerequisites:

- RHEL 10 system with crond and anacron installed
- Multiple user accounts for testing access control

Tasks:

1. Create system-wide backup script that runs daily at 2:30 AM
2. Configure user crontab for log rotation every 6 hours
3. Set up anacron for weekly system updates (for laptop usage)
4. Implement cron access control allowing only specific users
5. Create monitoring script that checks cron job execution

Verification Commands:

```
crontab -l                                          # Check user crontab
cat /etc/crontab                                    # Check system crontab
ls -la /etc/cron.allow /etc/cron.deny              # Check access control
grep CRON /var/log/cron                            # Check cron execution logs
```

Lab 13B: Systemd Timer Implementation (van Vugt-focused)

Time Limit: 30 minutes **Objective**: Build modern systemd-based scheduling system with advanced timer features

Prerequisites:

- RHEL 10 system with systemd
- Understanding of systemd unit files

Tasks:

1. Create systemd service and timer for automated system cleanup
2. Configure calendar-based timer for business hours only (9 AM - 5 PM, weekdays)
3. Implement persistent timer that catches up missed executions
4. Set up monotonic timer for post-boot system configuration
5. Create timer with randomized delay for distributed execution

Verification Commands:

```
systemctl list-timers --all                       # Check all timers
systemctl status cleanup.timer cleanup.service    # Check timer status
journalctl -u cleanup.timer                       # Check timer logs
systemd-analyze calendar "Mon..Fri *-*-* 09..17:00:00"  # Validate calendar
```

Lab 13C: Synthesis Challenge - Enterprise Task Scheduling

Time Limit: 35 minutes **Objective**: Design comprehensive enterprise scheduling system combining all methodologies

Prerequisites:

- Multiple RHEL 10 systems for distributed scheduling
- Network connectivity for centralized monitoring

Tasks:

1. Design multi-tier scheduling system using both cron and systemd timers
2. Implement centralized task monitoring and alerting
3. Create backup scheduling with dependency management
4. Set up user-specific development environment automation
5. Design disaster recovery procedures for scheduling systems
6. Implement security hardening for all scheduled tasks

Advanced Requirements:

- Combine Ghori's systematic approach with van Vugt's modern timer techniques
- Implement cross-system task coordination
- Create automated failover mechanisms for critical tasks

Verification Commands:

```
systemctl list-timers && crontab -l               # Check all scheduling
grep -r "CRON\|Timer" /var/log/                   # Check execution logs
ss -tulnp | grep -E "(cron|systemd)"              # Check related services
find /etc -name "*cron*" -o -name "*.timer" | head -10  # Find config files
```

8. Troubleshooting Common Issues

Cron Jobs Not Executing

```
# Symptoms: Scheduled tasks not running
# Check cron service status
systemctl status crond

# Check cron logs
journalctl -u crond
tail -f /var/log/cron

# Common issues:
# 1. Incorrect crontab syntax
crontab -l | head -5                              # Check syntax

# 2. Missing executable permissions
ls -la /path/to/script.sh
chmod +x /path/to/script.sh

# 3. PATH issues in script
# Add to script: PATH=/usr/local/bin:/usr/bin:/bin

# 4. User access denied
ls -la /etc/cron.{allow,deny}
```

Environment Variables in Cron

```
# Symptoms: Script works manually but fails in cron
# Solution: Set environment variables in crontab

# Method 1: In crontab
SHELL=/bin/bash
PATH=/usr/local/bin:/usr/bin:/bin
HOME=/home/username
MAILTO=admin@company.com

# Method 2: Source environment in script
#!/bin/bash
source /etc/environment
source ~/.bashrc
# Rest of script...
```

Systemd Timer Not Triggering

```
# Symptoms: Timer exists but service doesn't run
# Check timer status
systemctl status timer-name.timer
systemctl list-timers timer-name.timer

# Check service dependencies
systemctl cat timer-name.timer
systemctl show timer-name.timer | grep Requires

# Verify calendar syntax
systemd-analyze calendar "Mon *-*-* 02:00:00"

# Check logs
journalctl -u timer-name.timer -f
journalctl -u timer-name.service
```

At Jobs Not Running

```
# Symptoms: at command accepts job but doesn't execute
# Check atd service
systemctl status atd

# Check job queue
atq
at -c job_number                                    # Show job details

# Check permissions
ls -la /etc/at.{allow,deny}
ls -la /var/spool/at/

# Check logs
journalctl -u atd
tail -f /var/log/cron                               # at uses cron logging
```

Permission Denied Errors

```
# Symptoms: Jobs fail with permission errors
# Check script ownership and permissions
ls -la /path/to/script.sh
chmod 755 /path/to/script.sh
chown username:group /path/to/script.sh

# Check SELinux contexts
ls -Z /path/to/script.sh
restorecon -v /path/to/script.sh

# Check sudo requirements for system tasks
# Add to /etc/sudoers if needed:
username ALL=(ALL) NOPASSWD: /usr/local/bin/script.sh
```

9. Best Practices

Security Considerations

- Use full paths for all commands and scripts
- Set restrictive permissions on cron scripts (755 or 750)
- Implement proper logging and monitoring
- Use service accounts for system tasks
- Regularly review and audit scheduled tasks
- Implement access control using allow/deny files

Performance Optimization

- Avoid scheduling multiple resource-intensive tasks simultaneously
- Use batch command for CPU-intensive tasks
- Implement task dependencies to prevent conflicts
- Monitor system resources during scheduled task execution
- Use randomized delays for distributed environments

Error Handling and Monitoring

- Redirect output to log files for debugging
- Implement notification mechanisms for task failures
- Use MAILTO variable for cron error notifications
- Create monitoring scripts to verify task completion
- Maintain audit logs of all scheduled task changes

Modern Scheduling Strategy

- Prefer systemd timers for new implementations
- Use persistent timers for critical tasks
- Implement proper service dependencies
- Leverage systemd's logging and status capabilities
- Design for easy monitoring and troubleshooting

10. Integration with Other RHCSA Topics

Service Management Integration

- Schedule service restarts and updates
- Coordinate scheduled tasks with service dependencies
- Monitor service health through scheduled checks
- Implement service failover through automation

Storage Integration

- Schedule filesystem cleanup and maintenance
- Automate backup and archive operations
- Monitor disk usage and implement alerts
- Coordinate with LVM operations for snapshots

Security Integration

- Schedule security updates and patches
- Automate log analysis and security monitoring
- Coordinate with SELinux policy updates
- Implement automated security scanning

Network Integration

- Schedule network monitoring and diagnostics
- Automate network configuration backups
- Coordinate with network service maintenance
- Implement distributed task coordination

Module 13 Summary: Task scheduling and automation are essential for maintaining efficient and reliable systems. This module provides comprehensive coverage of both traditional cron-based scheduling and modern systemd timer approaches. Understanding how to design, implement, and troubleshoot automated tasks is crucial for RHCSA certification and effective system administration. The synthesis of different scheduling methodologies ensures flexibility and reliability in diverse environments.

2.16 14 - Flatpak Software Management

Navigation: ← Scheduled Tasks | Index | Next → Troubleshooting

---

1. Executive Summary

Topic Scope: Flatpak application packaging, repository management, and software distribution on RHEL 10

RHCSA Relevance: Flatpak replaces container management (Podman) as an RHCSA exam objective starting with RHEL 10. Candidates must be able to configure Flatpak repositories and manage Flatpak-based software.

Exam Weight: High

Prerequisites: Basic package management (Module 06), familiarity with DNF/RPM

Related Topics: Package Management

---

2. Conceptual Foundation

What is Flatpak?

Flatpak is a framework for distributing desktop and command-line applications on Linux. It provides a sandboxed environment where applications run isolated from the host system, with their own bundled dependencies.

Key characteristics:

- **Sandboxed execution**: Applications run in isolated environments with controlled access to host resources
- **Bundled dependencies**: Each application ships with its own runtime and libraries, avoiding dependency conflicts
- **Distribution-agnostic**: The same Flatpak runs on any Linux distribution
- **Automatic updates**: Applications can be updated independently of the host OS
- **OSTree-based**: Uses content-addressable storage for efficient deduplication and delta updates

Flatpak Architecture

- **Runtimes**: Shared sets of base libraries (e.g., `org.freedesktop.Platform`, `org.gnome.Platform`) that provide common dependencies. Multiple applications can share the same runtime.
- **Applications**: The actual software packages, built against a specific runtime
- **Remotes**: Repositories from which runtimes and applications are fetched (similar to DNF repositories)
- **Refs**: References to specific versions of applications or runtimes (e.g., `app/org.gimp.GIMP/x86_64/stable`)
- **Sandbox**: The isolation boundary using namespaces, seccomp, and portals

System vs User Installs

Flatpak supports two installation scopes:

- **System installs** (`--system`, default): Available to all users, stored in `/var/lib/flatpak/`. Requires root or polkit authorization.
- **User installs** (`--user`): Available only to the current user, stored in `~/.local/share/flatpak/`. No root required.

Remotes and Repositories

- **Flathub** (`https://flathub.org/repo/flathub.flatpakrepo`): The largest Flatpak repository with thousands of community and vendor applications
- **RHEL Flatpak remote**: Red Hat's curated Flatpak repository for enterprise applications
- **Fedora Flatpaks**: Fedora's official Flatpak repository

Sandboxing and Permissions

Flatpak uses a portal-based permission system:

- **Filesystem access**: Controlled via `--filesystem=` overrides
- **Network access**: Enabled/disabled per application
- **Device access**: Camera, GPU, etc. via portals
- **D-Bus access**: Controlled bus access for desktop integration
- Applications request permissions at install time; users can override them

Common Misconceptions

- **Flatpak is not a container runtime** — Unlike Podman/Docker, Flatpak is designed for desktop/CLI applications, not server workloads
- **Flatpaks are not always large** — Runtimes are shared across applications, so the second Flatpak using the same runtime adds minimal disk usage

- **Flatpak does not replace RPM/DNF** — System packages (kernel, systemd, libraries) are still managed by DNF. Flatpak is for application-layer software

---

3. Command Mastery

Essential Commands

```
# Repository (remote) management
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo  # Add
Flathub
flatpak remote-add --user myremote https://example.com/repo/example.flatpakrepo          # Add
user remote
flatpak remote-delete flathub                          # Remove a remote
flatpak remote-ls flathub                              # List available apps from remote
flatpak remotes                                        # List configured remotes

# Search and information
flatpak search gimp                                    # Search for applications
flatpak info org.gimp.GIMP                             # Show detailed app information

# Install and uninstall
flatpak install flathub org.gimp.GIMP                  # Install from specific remote
flatpak install org.gimp.GIMP                          # Install (auto-selects remote)
flatpak install --user org.gimp.GIMP                   # Install for current user only
flatpak uninstall org.gimp.GIMP                        # Uninstall application
flatpak uninstall --unused                             # Remove unused runtimes

# List installed applications
flatpak list                                           # List all installed Flatpaks
flatpak list --app                                     # List only applications (not runtimes)
flatpak list --runtime                                 # List only runtimes

# Run and update
flatpak run org.gimp.GIMP                              # Run application
flatpak update                                         # Update all Flatpaks
flatpak update org.gimp.GIMP                           # Update specific application
```

Command Reference Table

Command	Purpose	Key Options	Example
`flatpak remote-add`	Add repository	`--if-not-exists`, `--user`	`flatpak remote-add flathub URL`
`flatpak remote-delete`	Remove repository	`--force`	`flatpak remote-delete flathub`
`flatpak remote-ls`	List remote apps	`--app`, `--runtime`	`flatpak remote-ls flathub`
`flatpak remotes`	Show configured remotes	`--show-details`	`flatpak remotes`
`flatpak search`	Search for apps	—	`flatpak search firefox`
`flatpak install`	Install application	`--user`, `--system`, `-y`	`flatpak install flathub org.gimp.GIMP`
`flatpak uninstall`	Remove application	`--unused`	`flatpak uninstall org.gimp.GIMP`
`flatpak list`	List installed	`--app`, `--runtime`	`flatpak list --app`
`flatpak run`	Run application	`--command=`	`flatpak run org.gimp.GIMP`
`flatpak update`	Update apps	`-y`	`flatpak update`
`flatpak info`	Show app details	—	`flatpak info org.gimp.GIMP`

---

4. Procedural Workflows

Standard Procedure: Adding a Remote and Installing Software

1. **Add the Flathub remote** (if not already configured):

```
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
```

2. **Verify the remote is configured**:

```
flatpak remotes
```

3. **Search for the desired application**:

```
flatpak search gimp
```

4. **Install the application**:

```
flatpak install flathub org.gimp.GIMP -y
```

5. **Verify installation**:

```
flatpak list --app | grep -i gimp
flatpak info org.gimp.GIMP
```

6. **Run the application**:

```
flatpak run org.gimp.GIMP
```

Standard Procedure: User-Level Installation

1. **Add remote for current user only**:

```
flatpak remote-add --user --if-not-exists flathub https://flathub.org/repo/
flathub.flatpakrepo
```

2. **Install application for current user**:

```
flatpak install --user flathub org.mozilla.firefox -y
```

3. **Verify user-level install**:

```
flatpak list --user --app
ls ~/.local/share/flatpak/app/
```

Standard Procedure: Updating and Cleaning Up

1. **Check for available updates**:

```
flatpak update --appstream    # Update metadata
flatpak remote-ls --updates   # List available updates
```

2. **Update all installed Flatpaks**:

```
flatpak update -y
```

3. **Remove unused runtimes** (after uninstalling applications):

```
flatpak uninstall --unused -y
```

---

5. Configuration Deep Dive

Primary Configuration Locations

- **System remotes**: `/var/lib/flatpak/repo/`
- **System installations**: `/var/lib/flatpak/app/`, `/var/lib/flatpak/runtime/`
- **User remotes and installations**: `~/.local/share/flatpak/`
- **Remote configuration**: `/etc/flatpak/remotes.d/` (system-wide remote definitions)

Adding Remotes via Configuration File

Flatpak remotes can be pre-configured by placing `.flatpakrepo` files in `/etc/flatpak/remotes.d/`:

```
# /etc/flatpak/remotes.d/flathub.flatpakrepo
[Flatpak Repo]
Title=Flathub
Url=https://dl.flathub.org/repo/
Homepage=https://flathub.org/
Comment=Central repository of Flatpak applications
Icon=https://dl.flathub.org/repo/logo.svg
GPGKey=mQINBFlD2sABEADsiUZUO...
```

Permission Overrides

Override application sandbox permissions:

```
# Grant filesystem access
flatpak override --user --filesystem=home org.gimp.GIMP

# Remove network access
flatpak override --user --no-network org.example.App

# View current overrides
flatpak override --user --show org.gimp.GIMP

# Reset overrides to defaults
flatpak override --user --reset org.gimp.GIMP
```

Override files are stored in:

- System: `/var/lib/flatpak/overrides/`
- User: `~/.local/share/flatpak/overrides/`

---

6. Hands-On Labs

Lab 14.1: Configure Flatpak and Install Applications (Ghori Ch 12)

Objective: Set up Flatpak repositories and install applications at system and user levels

Steps:

1. **Verify Flatpak is installed** (it should be on RHEL 10 by default):

```
rpm -q flatpak
flatpak --version
```

2. **List currently configured remotes**:

```
flatpak remotes
```

3. **Add the Flathub repository** (system-wide, requires root):

```
sudo flatpak remote-add --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
```

4. **Verify the remote was added**:

```
flatpak remotes --show-details
```

5. **Search for and install an application**:

```
flatpak search calculator
sudo flatpak install flathub org.gnome.Calculator -y
```

6. **Verify the installation**:

```
flatpak list --app
flatpak info org.gnome.Calculator
```

7. **Run the installed application**:

```
flatpak run org.gnome.Calculator
```

8. **Install an application at user level** (no root needed):

```
flatpak remote-add --user --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
flatpak install --user flathub org.gnome.TextEditor -y
flatpak list --user --app
```

Verification:

```
flatpak remotes                    # Should show flathub
flatpak list --app                 # Should show installed apps
flatpak info org.gnome.Calculator  # Should show app details
```

Expected Result: Flathub is configured, applications are installed at both system and user levels, and can be launched successfully.

Lab 14.2: Managing Flatpak Applications

Objective: Practice updating, removing, and managing Flatpak applications

Steps:

1. **Update all installed Flatpaks**:

```
flatpak update -y
```

2. **List installed runtimes**:

```
flatpak list --runtime
```

3. **Uninstall an application**:

```
sudo flatpak uninstall org.gnome.Calculator -y
```

4. **Clean up unused runtimes**:

```
sudo flatpak uninstall --unused -y
```

5. **Verify removal**:

```
flatpak list --app
```

6. **Check disk usage**:

```
du -sh /var/lib/flatpak/
du -sh ~/.local/share/flatpak/
```

Verification:

```
flatpak list --app                # Removed apps should be gone
flatpak list --runtime            # Unused runtimes should be cleaned
```

Lab 14.3: Synthesis Challenge - Enterprise Flatpak Deployment

Objective: Configure a complete Flatpak environment suitable for enterprise use

Scenario: As a system administrator, configure Flatpak on a RHEL 10 system so that:

- Flathub is available as a system-wide remote
- A standard set of applications is installed for all users
- A regular user can install additional applications at the user level

Requirements:

1. Add Flathub as a system-wide remote
2. Install two system-wide applications
3. As a regular user, add a user-level remote and install one application
4. Update all installed Flatpaks
5. Verify system vs user install locations

Solution Steps:

1. **System-wide setup** (as root):

```
sudo flatpak remote-add --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
sudo flatpak install flathub org.gnome.Calculator org.gnome.TextEditor -y
```

2. **User-level setup** (as regular user):

```
flatpak remote-add --user --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
flatpak install --user flathub org.gnome.Logs -y
```

3. **Update everything**:

```
sudo flatpak update -y
flatpak update --user -y
```

4. **Verify**:

```
flatpak list --app                          # All apps
flatpak list --app --system                 # System apps
flatpak list --app --user                   # User apps
ls /var/lib/flatpak/app/                    # System install path
ls ~/.local/share/flatpak/app/              # User install path
```

---

7. Troubleshooting Playbook

Common Issues

Issue 1: Remote Add Fails with GPG Error

Symptoms:

- Error about GPG verification when adding a remote
- "GPG signatures found, but none are in trusted keyring"

Diagnosis:

```
flatpak remotes --show-details     # Check existing remote config
```

Resolution:

```
# Re-add the remote (the .flatpakrepo file includes the GPG key)
flatpak remote-delete flathub
flatpak remote-add flathub https://flathub.org/repo/flathub.flatpakrepo
```

Prevention: Always use the official `.flatpakrepo` URL which bundles the GPG key.

Issue 2: Application Won't Install — Missing Runtime

Symptoms:

- Installation fails with "runtime not found" error

Diagnosis:

```
flatpak info --show-runtime org.example.App    # Check required runtime
flatpak list --runtime                          # List installed runtimes
```

Resolution:

```
# Install the required runtime manually
flatpak install flathub org.freedesktop.Platform//24.08 -y
# Then retry the application install
flatpak install flathub org.example.App -y
```

Issue 3: Application Crashes or Cannot Access Files

Symptoms:

- Application starts but cannot read/write files
- Permission denied errors in application

Diagnosis:

```
flatpak info --show-permissions org.example.App
flatpak override --user --show org.example.App
```

Resolution:

```
# Grant filesystem access
flatpak override --user --filesystem=home org.example.App
# Or grant access to a specific path
flatpak override --user --filesystem=/path/to/data org.example.App
```

Diagnostic Command Sequence

```
flatpak --version                 # Verify Flatpak installation
flatpak remotes --show-details    # Check remote configuration
flatpak list --app                # List installed applications
flatpak list --runtime            # List installed runtimes
flatpak info org.example.App      # Check specific app details
journalctl --user -b | grep flatpak  # Check logs for errors
```

---

8. Quick Reference Card

Essential Commands At-a-Glance

```
# Remotes
flatpak remotes                                    # List remotes
flatpak remote-add --if-not-exists NAME URL        # Add remote
flatpak remote-delete NAME                         # Remove remote
flatpak remote-ls NAME                             # List apps in remote

# Applications
flatpak search KEYWORD                             # Search for apps
flatpak install REMOTE APP_ID                      # Install app
flatpak uninstall APP_ID                           # Remove app
flatpak list --app                                 # List installed apps
flatpak run APP_ID                                 # Run app
flatpak update                                     # Update all
flatpak info APP_ID                                # App details

# Cleanup
flatpak uninstall --unused                         # Remove unused runtimes
```

Key File Locations

- **System installations**: `/var/lib/flatpak/`
- **User installations**: `~/.local/share/flatpak/`
- **System remote configs**: `/etc/flatpak/remotes.d/`
- **Permission overrides (system)**: `/var/lib/flatpak/overrides/`
- **Permission overrides (user)**: `~/.local/share/flatpak/overrides/`

Verification Commands

```
flatpak remotes                    # Confirm remotes are configured
flatpak list --app                 # Confirm apps are installed
flatpak info APP_ID                # Confirm app details
flatpak run APP_ID                 # Confirm app runs
```

---

9. Knowledge Check

Conceptual Questions

1. **Question**: What is the difference between a Flatpak runtime and a Flatpak application? **Answer**: A runtime is a shared set of base libraries (like `org.freedesktop.Platform`) that provides common dependencies. An application is the actual software built against a specific runtime. Multiple applications can share the same runtime, reducing disk usage.
2. **Question**: What is the difference between system-level and user-level Flatpak installs? **Answer**: System installs (default) are stored in `/var/lib/flatpak/` and available to all users but require root privileges. User installs (`--user`) are stored in `~/.local/share/flatpak/` and available only to the installing user but require no elevated privileges.
3. **Question**: How does Flatpak differ from RPM/DNF package management? **Answer**: DNF manages system-level packages (kernel, libraries, system services) from RPM repositories. Flatpak manages sandboxed applications with bundled dependencies, providing isolation from the host system. They serve complementary roles — DNF for the base OS, Flatpak for application-layer software.

Practical Scenarios

1. **Scenario**: A user needs to install GIMP from Flathub but Flathub is not configured on the system. **Solution**:

```
sudo flatpak remote-add --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
sudo flatpak install flathub org.gimp.GIMP -y
```

2. **Scenario**: A regular user wants to install applications without root access. **Solution**:

```
flatpak remote-add --user --if-not-exists flathub \
  https://flathub.org/repo/flathub.flatpakrepo
flatpak install --user flathub org.example.App -y
```

Command Challenges

1. **Challenge**: List all Flatpak applications (not runtimes) installed on the system. **Answer**: `flatpak list --app` **Explanation**: The `--app` flag filters output to show only applications, excluding shared runtimes.
2. **Challenge**: Remove all unused runtimes left over from uninstalled applications. **Answer**: `flatpak uninstall --unused` **Explanation**: After uninstalling applications, their runtimes may remain. `--unused` identifies and removes runtimes no longer needed by any installed application.

---

10. Exam Strategy

Topic-Specific Tips

- Know the difference between `--system` (default) and `--user` installation scopes
- Remember that `flatpak remote-add` requires a URL to a `.flatpakrepo` file, not just a hostname
- Use `--if-not-exists` when adding remotes to make commands idempotent
- The Flatpak application ID format is reverse-DNS: `org.gimp.GIMP`, `org.mozilla.firefox`

Common Exam Scenarios

1. **Scenario**: Configure Flathub repository and install a specified application **Approach**:

```
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak install flathub org.example.App -y
flatpak list --app    # Verify
```

2. **Scenario**: Install a Flatpak application for a specific user without root **Approach**:

```
flatpak remote-add --user --if-not-exists flathub https://flathub.org/repo/
flathub.flatpakrepo
flatpak install --user flathub org.example.App -y
```

Time Management

- **Estimated Time**: 5-8 minutes for Flatpak tasks (remote setup + install + verify)
- **Quick Verification**: `flatpak list --app` confirms installation immediately

Pitfalls to Avoid

- Forgetting to add the remote before trying to install
- Not using `--if-not-exists` (causes errors if remote already configured)
- Confusing system vs user installs (check which the question asks for)
- Not verifying with `flatpak list` after installation

Summary

Key Takeaways

- Flatpak is the RHEL 10 RHCSA method for application distribution (replacing containers/Podman from RHEL 9)
- Remotes (repositories) must be configured before applications can be installed
- System installs require root; user installs do not
- Applications run sandboxed with controlled permissions
- Unused runtimes should be cleaned up with `flatpak uninstall --unused`

Critical Commands to Remember

```
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak install flathub org.example.App
flatpak list --app
flatpak uninstall org.example.App
flatpak uninstall --unused
flatpak update
flatpak search keyword
flatpak run org.example.App
```

Next Steps

- Continue to Troubleshooting
- Review related topics: Package Management
- Practice installing and managing Flatpak applications on your RHEL 10 lab VM

Navigation: ← Scheduled Tasks | Index | Next → Troubleshooting

2.17 Module 15: System Troubleshooting & Recovery

1. Learning Objectives

- Master systematic troubleshooting methodologies
- Diagnose and resolve boot, network, and service failures
- Analyze system performance issues and resource constraints
- Recover from filesystem corruption and storage failures
- Troubleshoot SELinux, firewall, and security-related issues
- Implement preventive maintenance and monitoring strategies
- Document troubleshooting procedures and solutions

2. Key Concepts

Troubleshooting Methodology

- **Problem identification**: Define symptoms and scope
- **Information gathering**: Collect system state and logs
- **Analysis**: Correlate data and identify root causes
- **Solution implementation**: Apply fixes systematically
- **Verification**: Confirm problem resolution
- **Documentation**: Record solutions for future reference

System State Analysis

- **Boot process**: GRUB, kernel, systemd initialization
- **Service status**: systemd unit states and dependencies
- **Resource utilization**: CPU, memory, disk, network usage
- **Log analysis**: System and application log examination
- **Configuration validation**: Syntax and logical correctness

Recovery Techniques

- **Boot recovery**: Rescue mode, emergency mode, single-user mode
- **Filesystem repair**: fsck, xfs_repair, data recovery
- **Service restoration**: Dependency resolution, configuration fixes
- **Network recovery**: Interface configuration, routing, DNS
- **Security recovery**: SELinux troubleshooting, permission fixes

Diagnostic Tools

- **System information**: lscpu, lsmem, lsblk, lspci, lsusb
- **Performance monitoring**: top, htop, iotop, vmstat, iostat
- **Network diagnostics**: ping, traceroute, netstat, ss, tcpdump
- **Storage analysis**: df, du, lsof, fuser, blkid

3. Essential Commands

System Information Gathering

```
# Hardware information
lscpu                                    # CPU information
lsmem                                    # Memory information
lsblk                                    # Block device information
lspci                                    # PCI device information
lsusb                                    # USB device information
dmidecode                                # DMI/SMBIOS information

# System state
uptime                                   # System uptime and load
uname -a                                 # Kernel and system info
hostnamectl                              # System hostname info
timedatectl                              # Time and timezone info
systemctl status                         # Overall system status
```

Process and Resource Analysis

```
# Process monitoring
ps aux                                   # Process snapshot
ps -ef --forest                          # Process tree
top -b -n1                               # One-time top output
htop                                     # Interactive process viewer
pstree                                   # Process tree visualization

# Resource utilization
free -h                                  # Memory usage
df -h                                    # Disk usage
du -sh /path/*                           # Directory sizes
lsof                                     # Open files
fuser -v /path/file                      # Processes using file
```

Network Diagnostics

```
# Network connectivity
ping -c 4 target                                    # Test connectivity
traceroute target                                   # Route tracing
mtr target                                          # Combined ping/traceroute

# Network configuration
ip addr show                                        # Interface addresses
ip route show                                       # Routing table
ss -tulnp                                          # Socket statistics
netstat -rn                                        # Routing table (legacy)

# DNS resolution
nslookup hostname                                   # DNS lookup
dig hostname                                        # Detailed DNS query
host hostname                                       # Simple DNS lookup
```

Service Troubleshooting

```
# Service analysis
systemctl status service_name                       # Service status
systemctl is-active service_name                   # Check if active
systemctl is-enabled service_name                  # Check if enabled
systemctl list-dependencies service_name           # Service dependencies

# Configuration validation
nginx -t                                            # Nginx config test
httpd -t                                            # Apache config test
sshd -T                                             # SSH config test
postfix check                                       # Postfix config test
```

Log Analysis

```
# System logs
journalctl -b                                       # Current boot logs
journalctl --since "1 hour ago"                    # Recent logs
journalctl -u service_name                         # Service-specific logs
journalctl -p err                                  # Error-level messages
journalctl -f                                      # Follow logs (tail -f)

# Traditional logs
tail -f /var/log/messages                          # System messages
grep -i error /var/log/messages                    # Error messages
awk '/ERROR/ {print $1, $2, $3, $NF}' /var/log/secure  # Extract error info
```

4. Asghar Ghori's Approach

Systematic Problem Analysis

Ghori emphasizes structured troubleshooting workflow:

```
# Step 1: Problem definition and scope
echo "Problem: Service X not responding"
echo "Scope: Single service on one server"
echo "Impact: Users cannot access application"
echo "Timeline: Started 2 hours ago"

# Step 2: Initial system health check
uptime && free -h && df -h
systemctl --failed                                  # Failed services
journalctl -p err --since "3 hours ago"            # Recent errors

# Step 3: Service-specific analysis
systemctl status httpd
journalctl -u httpd --since "3 hours ago"
httpd -t                                            # Config validation
```

Boot Troubleshooting Methodology

Ghori's systematic boot problem resolution:

```
# Boot analysis workflow
# 1. Identify boot stage failure
dmesg | grep -i error                               # Kernel messages
journalctl -b | grep -i fail                       # Boot failures

# 2. GRUB issues
grub2-editenv list                                  # Check default kernel
grub2-mkconfig -o /boot/grub2/grub.cfg             # Regenerate GRUB config

# 3. Filesystem issues
mount | grep " / "                                  # Check root filesystem
fsck /dev/sda1                                      # Filesystem check (unmounted)

# 4. Service startup issues
systemctl list-units --failed                      # Failed services
systemctl list-jobs                                # Pending jobs
```

Network Troubleshooting Steps

```
# Ghori's network diagnosis process
# 1. Physical/Link layer
ip link show                                    # Interface status
ethtool eth0                                    # Interface details

# 2. Network layer
ip addr show                                    # IP configuration
ip route show                                   # Routing table
ping -c 4 gateway_ip                           # Gateway connectivity

# 3. Application layer
ss -tulnp | grep :80                           # Service listening
curl -I http://localhost                       # Local service test
nmap -p 80 target_server                       # Remote service test
```

5. Sander van Vugt's Approach

Advanced Diagnostic Techniques

Van Vugt focuses on deep system analysis:

```
# Comprehensive system performance analysis
# 1. CPU analysis
vmstat 1 10                                     # CPU and memory stats
mpstat 1 5                                      # Per-CPU statistics
sar -u 1 10                                     # CPU utilization over time

# 2. Memory analysis
vmstat -s                                       # Memory statistics
slabtop                                         # Kernel slab allocation
/proc/meminfo                                   # Detailed memory info

# 3. I/O analysis
iostat -x 1 10                                  # Extended I/O statistics
iotop                                           # I/O by process
lsof +D /path                                   # Files open in directory
```

Root Cause Analysis Framework

Van Vugt's systematic root cause identification:

```
# Multi-layer analysis approach
# 1. Hardware layer
dmesg | grep -i "hardware error"                    # Hardware issues
mcelog --client                                     # Machine check errors
smartctl -a /dev/sda                                # Disk health

# 2. Kernel layer
dmesg | grep -i "kernel"                            # Kernel messages
cat /proc/sys/kernel/tainted                        # Kernel taint status
modinfo module_name                                 # Module information

# 3. Application layer
strace -p PID                                       # System call tracing
ltrace -p PID                                       # Library call tracing
gdb --pid PID                                       # Debug running process
```

Advanced Log Correlation

```
# Van Vugt's log correlation methodology
# 1. Timeline reconstruction
journalctl --since "2023-01-01 14:00" --until "2023-01-01 15:00" | head -50
aureport --start 01/01/2023 14:00:00 --end 01/01/2023 15:00:00

# 2. Multi-source correlation
# Combine system logs, application logs, and audit logs
tail -f /var/log/messages /var/log/secure /var/log/audit/audit.log

# 3. Pattern analysis
awk '/pattern/ {count++} END {print count}' /var/log/messages
grep -E "ERROR|CRITICAL|FATAL" /var/log/application.log | sort | uniq -c
```

6. Command Examples and Scenarios

Scenario 1: Service Startup Failure

```
# Problem: Web server won't start after system reboot
# Systematic diagnosis:

# 1. Check service status
systemctl status httpd
systemctl is-enabled httpd

# 2. Check configuration
httpd -t
ls -la /etc/httpd/conf/httpd.conf

# 3. Check dependencies
systemctl list-dependencies httpd
systemctl status network.target

# 4. Check logs
journalctl -u httpd --since boot
grep httpd /var/log/messages

# 5. Check ports and firewall
ss -tulnp | grep :80
firewall-cmd --list-services
```

Scenario 2: System Performance Degradation

```
# Problem: System running slowly, high load average
# Performance analysis:

# 1. Overall system health
uptime                                          # Load averages
free -h                                         # Memory usage
df -h                                           # Disk space

# 2. Process analysis
top -b -n1 -o %CPU | head -15                  # CPU-intensive processes
ps aux --sort=-%mem | head -10                 # Memory-intensive processes

# 3. I/O analysis
iostat -x 1 5                                  # I/O wait times
iotop -ao                                       # I/O by process

# 4. Network analysis
ss -s                                           # Socket summary
netstat -i                                      # Interface statistics
```

Scenario 3: Boot Failure Recovery

```
# Problem: System won't boot, dropped to emergency shell
# Recovery procedure:

# 1. Check filesystem integrity
mount -o remount,rw /                           # Remount root writable
fsck /dev/sda2                                  # Check root filesystem

# 2. Check and fix fstab
cat /etc/fstab                                  # Review mount points
blkid                                           # Verify UUIDs

# 3. Regenerate initramfs if needed
dracut -f                                       # Force regenerate initramfs

# 4. Fix GRUB if necessary
grub2-install /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg
```

7. Lab Exercises

Lab 15A: Service and Configuration Troubleshooting (Ghori-focused)

Time Limit: 30 minutes **Objective**: Diagnose and resolve common service configuration issues

Prerequisites:

- RHEL 10 system with intentionally misconfigured services
- Apache httpd and SSH services installed

Setup (Instructor creates these issues):

1. Apache httpd service fails to start due to configuration syntax error
2. SSH service running but refusing connections due to permission issue
3. Network service configured with conflicting IP addresses
4. Cron service not executing jobs due to permission problems

Tasks:

1. Identify and fix Apache configuration syntax error
2. Resolve SSH connection issues and verify remote access
3. Correct network configuration conflicts
4. Troubleshoot and fix cron job execution problems
5. Document all findings and solutions

Verification Commands:

```
systemctl status httpd sshd                          # Service status
curl http://localhost                                 # Test web service
ssh localhost id                                     # Test SSH access
ip addr show                                          # Network configuration
crontab -l && grep CRON /var/log/cron               # Cron verification
```

Lab 15B: Performance and Resource Troubleshooting (van Vugt-focused)

Time Limit: 35 minutes **Objective**: Analyze and resolve system performance issues using advanced diagnostic techniques

Prerequisites:

- RHEL 10 system with performance monitoring tools installed
- Simulated high load conditions

Setup (Instructor creates these conditions):

1. Memory leak causing system slowdown
2. High I/O wait times due to disk issues
3. Network connectivity problems affecting services
4. CPU-intensive process consuming resources

Tasks:

1. Identify memory leak source and implement solution
2. Diagnose and resolve I/O performance bottleneck
3. Troubleshoot network connectivity issues
4. Find and manage resource-intensive processes
5. Create monitoring strategy to prevent recurrence

Verification Commands:

```
free -h && vmstat 1 3                                  # Memory status
iostat -x 1 3                                          # I/O performance
ss -tulnp && ping -c 4 8.8.8.8                        # Network status
top -b -n1 | head -15                                  # Process overview
```

Lab 15C: Synthesis Challenge - Complete System Recovery

Time Limit: 45 minutes **Objective**: Perform comprehensive system recovery using integrated troubleshooting methodologies

Prerequisites:

- RHEL 10 system with multiple simulated failures
- Access to rescue media and documentation

Setup (Multiple interconnected issues):

1. Boot failure due to corrupted filesystem
2. Network services not starting due to SELinux denials
3. Storage issues affecting application data
4. Security configuration preventing user access
5. Logging system failures hiding other issues

Tasks:

1. Recover system from boot failure using rescue mode
2. Resolve SELinux issues preventing service startup
3. Repair storage problems and recover application data
4. Fix security configuration issues
5. Restore logging functionality and analyze root causes
6. Implement preventive measures and monitoring
7. Create comprehensive incident report

Advanced Requirements:

- Combine both Ghori's systematic approach and van Vugt's deep analysis
- Use multiple diagnostic tools and correlation techniques
- Document complete recovery timeline and lessons learned

Verification Commands:

```
systemctl status && systemctl --failed             # Overall system health
mount && df -h                                     # Storage status
getenforce && ausearch -m AVC -ts recent          # SELinux status
journalctl --disk-usage && journalctl -p err      # Logging status
ss -tulnp | grep -E ":22|:80|:443"                # Critical services
```

8. Troubleshooting Common Issues

Boot Failure Scenarios

```
# GRUB not loading
# Symptoms: System boots directly to BIOS/UEFI
# Solution: Reinstall GRUB bootloader
grub2-install /dev/sda
grub2-mkconfig -o /boot/grub2/grub.cfg

# Kernel panic
# Symptoms: Kernel panic messages, system halt
# Solution: Boot with older kernel or recovery mode
# From GRUB menu: select older kernel version

# Root filesystem corruption
# Symptoms: Cannot mount root filesystem
# Solution: Boot to rescue mode and run fsck
mount -o remount,ro /
fsck /dev/sda2                                     # Replace with correct device
mount -o remount,rw /
```

Network Connectivity Issues

```
# No network connectivity
# Symptoms: Cannot reach external hosts
# Diagnosis and resolution:

# 1. Check interface status
ip link show                                    # Interface up/down status
nmcli device status                             # NetworkManager status

# 2. Check IP configuration
ip addr show                                    # IP addresses assigned
ip route show                                   # Routing table

# 3. Test connectivity layers
ping -c 4 127.0.0.1                            # Loopback test
ping -c 4 gateway_ip                           # Gateway test
ping -c 4 8.8.8.8                              # External IP test
ping -c 4 google.com                           # DNS resolution test

# 4. Fix common issues
systemctl restart NetworkManager                # Restart network service
nmcli connection up connection_name             # Bring up connection
```

High Load and Performance Issues

```
# System running slowly
# Symptoms: High load average, slow response
# Analysis and solutions:

# 1. Identify resource constraints
uptime                                          # Load averages
free -h                                         # Memory availability
df -h                                           # Disk space

# 2. Find resource consumers
ps aux --sort=-%cpu | head -10                  # CPU usage
ps aux --sort=-%mem | head -10                  # Memory usage
iotop -ao                                       # I/O activity

# 3. Address specific issues
# Kill runaway processes
kill -TERM PID
kill -KILL PID                                  # If TERM doesn't work

# Clean up disk space
du -sh /var/log/* | sort -h                    # Find large log files
journalctl --vacuum-time=1week                 # Clean journal logs
```

Service Dependencies and Failures

```
# Service won't start due to dependencies
# Symptoms: Service fails with dependency errors
# Resolution approach:

# 1. Check service dependencies
systemctl list-dependencies service_name
systemctl status dependency_service

# 2. Start dependencies manually
systemctl start dependency_service
systemctl enable dependency_service

# 3. Check for circular dependencies
systemctl show service_name | grep -E "Requires|After|Before"

# 4. Resolve configuration issues
# Check service configuration files
# Validate syntax where applicable
service_name -t                                    # If applicable
```

9. Best Practices

Troubleshooting Methodology

- Document all symptoms before making changes
- Follow systematic approach from general to specific
- Make one change at a time and test results
- Keep detailed log of all actions taken
- Back up configuration files before modifications
- Have rollback plan for all changes

Information Gathering

- Collect system information immediately when issue occurs
- Preserve log files and system state for analysis
- Use multiple information sources for correlation
- Take screenshots or save command output
- Interview users about what they were doing when issue occurred

Solution Implementation

- Test solutions in non-production environment first
- Implement least disruptive solution first

- Monitor system closely after implementing fixes
- Document all changes made for future reference
- Verify that solution doesn't create new problems

Preventive Measures

- Implement comprehensive monitoring and alerting
- Perform regular system health checks
- Keep system and applications updated
- Maintain current documentation and runbooks
- Regular backup and disaster recovery testing
- Train team on common troubleshooting procedures

10. Integration with Other RHCSA Topics

Service Management Integration

- Understand systemd service dependencies and failures
- Troubleshoot service startup and runtime issues
- Analyze service logs and performance metrics
- Implement service monitoring and alerting

Storage Integration

- Diagnose filesystem corruption and recovery procedures
- Troubleshoot LVM and storage performance issues
- Implement storage monitoring and capacity planning
- Recover from storage hardware failures

Security Integration

- Troubleshoot SELinux denials and policy issues
- Diagnose firewall rule conflicts and connectivity problems
- Investigate security incidents and unauthorized access
- Implement security monitoring and incident response

Network Integration

- Diagnose network connectivity and performance issues
- Troubleshoot DNS resolution and service discovery

- Analyze network traffic and security events
- Implement network monitoring and capacity planning

Module 15 Summary: System troubleshooting is the culmination of all RHCSA skills, requiring deep understanding of Linux system components and their interactions. This module provides comprehensive coverage of systematic troubleshooting methodologies, from basic problem identification to complex system recovery scenarios. Mastering both structured diagnostic approaches and advanced analysis techniques is essential for RHCSA certification and effective system administration in production environments. The synthesis of different troubleshooting philosophies ensures comprehensive problem-solving capabilities across all system components.

3. Quick References

3.1 RHCSA Exam Quick Reference

Essential Acronyms & Terms

Certification & System

- **RHCSA** - Red Hat Certified System Administrator (EX200)
- **RHEL** - Red Hat Enterprise Linux (version 9)
- **OS** - Operating System
- **CLI** - Command Line Interface
- **GUI** - Graphical User Interface
- **API** - Application Programming Interface
- **FQDN** - Fully Qualified Domain Name
- **TTY** - Teletypewriter (terminal)
- **EOF** - End of File
- **STDIN/STDOUT/STDERR** - Standard input/output/error

Hardware & Boot

- **CPU** - Central Processing Unit
- **RAM** - Random Access Memory
- **BIOS** - Basic Input/Output System
- **UEFI** - Unified Extensible Firmware Interface
- **GRUB** - Grand Unified Bootloader (version 2)
- **initramfs** - Initial RAM filesystem
- **kernel** - Core operating system
- **KVM** - Kernel-based Virtual Machine

Pre-Exam Checklist

- [ ] Verify VM access and connectivity
- [ ] Test sudo access: `sudo -l`
- [ ] Check available storage devices: `lsblk`
- [ ] Verify SELinux status: `getenforce`
- [ ] Note network interface names: `ip link show`
- [ ] Check default target: `systemctl get-default`
- [ ] Verify firewall status: `firewall-cmd --state`

File Management & Text Processing

Key Terms & Acronyms

- **inode** - Index node (file metadata structure)
- **hard link** - Direct link to inode (same filesystem)
- **soft link** - Symbolic link (can cross filesystems)
- **glob** - Pattern matching with wildcards
- **regex** - Regular expressions
- **pipe** - Data transfer between commands (|)
- **redirection** - Output/input redirection (>, >>, <)
- **archive** - Collection of files (tar)
- **compression** - Data reduction (gzip, bzip2, xz)
- **MIME type** - File type identification
- **buffer** - Temporary data storage

Key File Paths

```
/tmp/                       # Temporary files directory
/var/tmp/                   # Persistent temporary files
/proc/                      # Process and system information
/dev/null                   # Null device (discard output)
/dev/zero                   # Zero device (null bytes)
```

Essential Commands

```
# File operations
ls -la                              # List files with details
ls -ltr                             # List by time, newest last
cp file1 file2                      # Copy file
cp -r dir1 dir2                     # Copy directory recursively
mv file1 file2                      # Move/rename file
rm file1                            # Remove file
rm -rf directory                    # Remove directory and contents
mkdir -p /path/to/directory         # Create directory path
rmdir directory                     # Remove empty directory
touch filename                      # Create empty file or update timestamp

# File content viewing
cat filename                        # Display file content
less filename                       # View file with paging
head -n 10 filename                 # First 10 lines
tail -n 10 filename                 # Last 10 lines
tail -f filename                    # Follow file changes
wc -l filename                      # Count lines
wc -w filename                      # Count words
file filename                       # Determine file type

# Text processing
grep pattern filename               # Search for pattern
grep -i pattern filename            # Case-insensitive search
grep -r pattern /path               # Recursive search
grep -v pattern filename            # Invert match (exclude pattern)
grep -n pattern filename            # Show line numbers
sed 's/old/new/g' filename          # Replace text
awk '{print $1}' filename           # Print first column
cut -d: -f1 /etc/passwd             # Extract first field
sort filename                       # Sort lines
sort -u filename                    # Sort and remove duplicates
uniq filename                       # Remove consecutive duplicates
tr 'a-z' 'A-Z' < file               # Translate characters

# File searching
find /path -name "*.txt"            # Find files by name
find /path -type f -size +100M      # Find large files
find /path -user username           # Find files by owner
find /path -perm 755                # Find files by permissions
find /path -mtime -7                # Modified in last 7 days
locate filename                     # Fast file search (requires updatedb)
which command                       # Find command location
whereis command                     # Find command and manual locations

# Links
ln file1 file2                      # Create hard link
ln -s /path/to/file link_name       # Create symbolic link
ls -l link_name                     # Check link target
readlink link_name                  # Show link target
```

File Archiving & Compression

```
# Tar operations
tar -cf archive.tar files          # Create tar archive
tar -czf archive.tar.gz files      # Create compressed tar (gzip)
tar -cjf archive.tar.bz2 files     # Create compressed tar (bzip2)
tar -cJf archive.tar.xz files      # Create compressed tar (xz)
tar -tf archive.tar               # List archive contents
tar -xf archive.tar               # Extract archive
tar -xzf archive.tar.gz           # Extract gzip compressed tar
tar -xzf archive.tar.gz -C /path  # Extract to specific directory

# Compression utilities
gzip filename                      # Compress file
gunzip filename.gz                # Decompress file
bzip2 filename                    # Better compression
bunzip2 filename.bz2              # Decompress bzip2
xz filename                       # Best compression
unxz filename.xz                  # Decompress xz
zip archive.zip files             # Create zip archive
unzip archive.zip                 # Extract zip archive
```

Input/Output Redirection

```
# Redirection operators
command > file                     # Redirect stdout to file (overwrite)
command >> file                    # Redirect stdout to file (append)
command 2> file                    # Redirect stderr to file
command 2>&1                      # Redirect stderr to stdout
command < file                     # Use file as stdin
command | command2                 # Pipe output to next command
command | tee file                 # Output to both stdout and file
command &> file                    # Redirect both stdout and stderr
```

Common Tasks

```
# Create directory structure and files
mkdir -p /project/{docs,src,tests}
touch /project/docs/README.md
echo "Project files" > /project/docs/README.md

# Search and process log files
grep ERROR /var/log/messages | tail -10
grep -i "failed" /var/log/*.log | cut -d: -f1 | sort -u

# Archive and compress directories
tar -czf backup-$(date +%Y%m%d).tar.gz /home/user
find /tmp -name "*.log" -mtime +7 -exec rm {} \;

# Text processing workflow
cat file.txt | grep "pattern" | sort | uniq -c | sort -nr > results.txt

# Find and replace across multiple files
find /path -name "*.conf" -exec sed -i 's/old/new/g' {} \;
```

Troubleshooting

```
# File system issues
df -h                               # Check disk space
du -sh directory                    # Check directory size
lsof filename                       # See what's using file
fuser filename                      # Alternative to lsof

# Permission issues
ls -la filename                     # Check file permissions
file filename                       # Verify file type
stat filename                       # Detailed file information

# Text processing problems
file filename                       # Check file encoding
wc -l filename                      # Verify line count
od -c filename | head               # Check for special characters
```

Common Pitfalls

- **WRONG**: Using `rm -rf /` accidentally → **RIGHT**: Always double-check paths
- **WRONG**: Not quoting file names with spaces → **RIGHT**: Use quotes or escape spaces
- **WRONG**: Forgetting `-r` for directory operations → **RIGHT**: Use `-r` for recursive operations
- **WRONG**: Using `>` instead of `>>` → **RIGHT**: Use `>>` to append, `>` overwrites

---

User and Group Management

Key Terms & Acronyms

- **UID** - User Identifier (numeric user ID, root=0)
- **GID** - Group Identifier (numeric group ID)
- **sudo** - Superuser do (privilege escalation)
- **PAM** - Pluggable Authentication Modules
- **shadow** - Password hashing system
- **skel** - Skeleton directory template for new users
- **wheel** - Administrative group with sudo privileges

Key File Paths

```
/etc/passwd                  # User account information
/etc/shadow                  # Password hashes
/etc/group                   # Group information
/etc/sudoers                 # Sudo access (edit with visudo only)
/etc/default/useradd         # Default user settings
/home/username               # User home directories
/etc/skel/                   # Template for new user homes
```

Essential Commands

```
# User creation and management
useradd alice                          # Basic user with defaults
useradd -u 1001 -G wheel bob          # Specific UID + sudo access
useradd -r -s /sbin/nologin svcuser    # Service account
useradd -e 2024-12-31 tempuser        # Account with expiration
useradd -c "Full Name" -m username     # With comment and home dir

# Password management
passwd alice                           # Set password
chage -M 30 alice                     # Password expires in 30 days
chage -W 7 alice                      # 7-day warning before expiration
chage -l alice                        # List password aging info
chage -d 0 alice                      # Force password change on next login

# Account modifications
usermod -aG developers alice          # Add to supplementary group
usermod -L alice                      # Lock account
usermod -U alice                      # Unlock account
usermod -s /bin/bash alice            # Change shell

# Group management
groupadd -g 1500 developers           # Create group with specific GID
gpasswd -a alice developers           # Add user to group
gpasswd -d alice developers           # Remove user from group

# Information and verification
id alice                              # Show user ID and groups
groups alice                          # Show user groups
who                                   # Currently logged in users
last alice                            # Login history for user
```

Common Tasks

```
# Create user with sudo access
useradd -G wheel username && passwd username

# Create service account
useradd -r -s /sbin/nologin -d /var/lib/service serviceuser

# Set up password aging policy
chage -M 90 -m 7 -W 7 username

# Verify user setup
id username && groups username
su - username  # Test login
```

Sudo Configuration

```
# Always use visudo to edit
visudo

# Common sudoers entries:
alice ALL=(ALL) ALL                    # Full sudo access
%wheel ALL=(ALL) NOPASSWD: ALL          # Wheel group no password
alice ALL=(ALL) NOPASSWD: /bin/systemctl  # Specific command only
```

Troubleshooting

```
# User can't login
passwd -S username                     # Check password status
chage -l username                      # Check account expiration
ls -la /home/username                  # Check home directory permissions

# Sudo issues
visudo -c                              # Check sudoers syntax
groups username                        # Verify group membership
```

Common Pitfalls

- **WRONG**: Direct editing `/etc/sudoers` → **RIGHT**: Use `visudo`
- **WRONG**: Creating user without password → **RIGHT**: Always set password after `useradd`
- **WRONG**: Forgetting home directory → **RIGHT**: Use `-m` or check `/home`

File Permissions & Access Control

Key Terms & Acronyms

- **ACL** - Access Control List (extended permissions)
- **setuid** - Set User ID (execute as owner)
- **setgid** - Set Group ID (execute as group)
- **sticky bit** - Restrict deletion in shared directories
- **umask** - User file creation mask
- **chmod** - Change mode (permissions)
- **chown** - Change owner
- **chgrp** - Change group
- **facl** - File Access Control List utilities
- **effective permissions** - Final permissions after ACL evaluation
- **mask** - Maximum ACL permissions allowed

Key File Paths

```
/etc/passwd                   # User account information
/etc/group                    # Group information
/etc/login.defs              # Default umask and settings
```

Essential Commands

```
# Basic permission management
chmod 755 filename                    # Set permissions (octal)
chmod u+x filename                    # Add execute for user (symbolic)
chmod go-w filename                   # Remove write for group/others
chmod a+r filename                    # Add read for all
chown user:group filename             # Change owner and group
chown user filename                   # Change owner only
chgrp group filename                  # Change group only
chown -R user:group directory         # Recursive ownership change

# View permissions
ls -l filename                        # Long listing with permissions
stat filename                         # Detailed file information
getfacl filename                      # Show ACL permissions
ls -ld directory                      # Directory permissions

# Special permissions
chmod 4755 filename                   # Set setuid bit
chmod 2755 filename                   # Set setgid bit
chmod 1755 directory                  # Set sticky bit
chmod u+s filename                    # Set setuid (symbolic)
chmod g+s filename                    # Set setgid (symbolic)
chmod +t directory                    # Set sticky bit (symbolic)
find /path -perm -4000                # Find setuid files
find /path -perm -2000                # Find setgid files
find /path -perm -1000                # Find sticky bit files

# ACL management
setfacl -m u:alice:rw filename        # Grant user ACL permissions
setfacl -m g:developers:rwx directory # Grant group ACL permissions
setfacl -m o::r filename              # Set other permissions
setfacl -x u:alice filename           # Remove user ACL entry
setfacl -b filename                   # Remove all ACL entries
setfacl -d -m g:developers:rwx directory # Set default ACL
setfacl -R -m u:alice:rx directory    # Recursive ACL application

# umask operations
umask                                 # Show current umask
umask 022                            # Set umask (files 644, dirs 755)
umask 002                            # Set umask (files 664, dirs 775)
```

Permission Calculation

```
# Octal permissions breakdown
# Read (r) = 4, Write (w) = 2, Execute (x) = 1
# 7 = 4+2+1 = rwx (read, write, execute)
# 6 = 4+2   = rw- (read, write)
# 5 = 4+1   = r-x (read, execute)
# 4 = 4     = r-- (read only)

# Common permission combinations
644 = rw-r--r--  # Files: owner read/write, group/others read
755 = rwxr-xr-x  # Executables: owner full, group/others read/execute
600 = rw-------  # Private files: owner read/write only
700 = rwx------  # Private directories: owner full access only
666 = rw-rw-rw-  # World writable files (rare)
777 = rwxrwxrwx  # World writable directories (dangerous)

# Special permission values (first digit)
4000 = setuid    # Execute as file owner
2000 = setgid    # Execute as file group
1000 = sticky    # Restrict deletion
```

Common Tasks

```
# Set up shared directory with group collaboration
mkdir /shared
chgrp developers /shared
chmod 2775 /shared                    # setgid + group write
setfacl -d -m g::rwx /shared         # Default group permissions
setfacl -d -m o::r-x /shared         # Default other permissions

# Secure personal directory
chmod 700 /home/alice                 # Owner only access
chmod 600 /home/alice/.ssh/id_rsa     # Private key protection

# Make script executable for everyone
chmod +x script.sh
# Or more specifically:
chmod 755 script.sh

# Grant specific user access to file
setfacl -m u:bob:rw /data/important.txt
getfacl /data/important.txt           # Verify ACL

# Remove all permissions for group and others
chmod go= filename
# Or using octal:
chmod 600 filename

# Find files with wrong permissions
find /home -perm 777 -type f          # World-writable files (security risk)
find /usr/bin -not -user root         # Non-root owned executables
```

Troubleshooting Permission Issues

```
# Permission denied troubleshooting
ls -la /path/to/file                  # Check file permissions
ls -lad /path/to/                     # Check directory permissions
getfacl /path/to/file                 # Check ACLs
groups username                       # Check user's groups
id username                           # Check user ID and groups

# Fix common permission problems
chmod +x script                       # Make script executable
chown $(whoami) file                  # Take ownership of file
chmod u+w file                        # Add write permission for user
setfacl -b file                       # Remove all ACLs if causing issues

# Restore default permissions
chmod --reference=reference_file target_file  # Copy permissions
find /path -type f -exec chmod 644 {} \;     # Files to 644
find /path -type d -exec chmod 755 {} \;     # Directories to 755
```

ACL vs Traditional Permissions (Supplementary — not on RHEL 10 exam)

```
# Traditional permissions (3 entities: user, group, other)
chmod 750 file        # rwxr-x--- (user: rwx, group: r-x, other: ---)

# ACL allows multiple users and groups
setfacl -m u:alice:rw,u:bob:r,g:admins:rwx file
getfacl file          # Shows detailed ACL permissions

# ACL inheritance (default ACLs)
setfacl -d -m u:alice:rwx /directory  # New files inherit ACL
```

Common Pitfalls

- **WRONG**: Using `777` permissions everywhere → **RIGHT**: Use least privilege principle
- **WRONG**: Forgetting recursive flag for directories → **RIGHT**: Use `-R` for recursive operations
- **WRONG**: Not checking parent directory permissions → **RIGHT**: Verify full path permissions
- **WRONG**: Mixing ACLs and traditional permissions → **RIGHT**: Understand ACL mask interactions
- **WRONG**: Setting setuid on scripts → **RIGHT**: setuid only works on binary executables

---

Package Management

Key Terms & Acronyms

- **DNF** - Dandified YUM (RHEL 10 package manager)
- **YUM** - Yellowdog Updater Modified (legacy package manager)
- **RPM** - Red Hat Package Manager (low-level package format)
- **repository** - Package source location
- **GPG** - GNU Privacy Guard (package signing)
- **metadata** - Repository information cache
- **group** - Collection of related packages
- **module** - Application streams with multiple versions
- **EPEL** - Extra Packages for Enterprise Linux
- **BaseOS** - Core RHEL repository
- **AppStream** - Application and runtime repository

Key File Paths

```
/etc/dnf/dnf.conf              # DNF main configuration
/etc/yum.repos.d/              # Repository configuration files
/var/cache/dnf/                # DNF cache directory
/var/log/dnf.log              # DNF transaction log
/etc/rpm/                     # RPM configuration
```

Essential Commands

```
# Package information and search
dnf list                            # List all packages
dnf list installed                  # List installed packages
dnf list available                  # List available packages
dnf search httpd                    # Search package names/descriptions
dnf info httpd                      # Detailed package information
dnf provides /usr/sbin/httpd        # Find package providing file

# Package installation and removal
dnf install httpd                   # Install package
dnf install -y httpd                # Install without confirmation
dnf remove httpd                    # Remove package
dnf reinstall httpd                 # Reinstall package
dnf downgrade httpd                 # Downgrade to previous version

# System updates
dnf check-update                    # Check for updates
dnf update                          # Update all packages
dnf update httpd                    # Update specific package
dnf upgrade                         # Same as update (preferred)

# Group operations
dnf group list                      # List package groups
dnf group info "Development Tools"  # Group information
dnf group install "Development Tools" # Install group
dnf group remove "Development Tools"  # Remove group

# Repository management
dnf repolist                        # List enabled repositories
dnf repolist all                    # List all repositories
dnf config-manager --enable epel    # Enable repository
dnf config-manager --disable epel   # Disable repository
dnf makecache                       # Rebuild metadata cache

# Local package installation
dnf localinstall package.rpm        # Install local RPM
rpm -ivh package.rpm                # Install with RPM directly
rpm -Uvh package.rpm                # Upgrade with RPM
rpm -e package                      # Remove with RPM

# Package queries (RPM)
rpm -qa                             # List all installed packages
rpm -qi httpd                       # Package information
rpm -ql httpd                       # List files in package
rpm -qf /usr/sbin/httpd            # Find package owning file
rpm -qc httpd                       # List config files
rpm -qd httpd                       # List documentation
```

Module Operations (Application Streams)

```
# Module management
dnf module list                      # List available modules
dnf module list nodejs               # List specific module streams
dnf module info nodejs:16            # Module stream information
dnf module install nodejs:16         # Install specific stream
dnf module enable nodejs:16          # Enable stream (don't install)
dnf module disable nodejs            # Disable module
dnf module reset nodejs              # Reset module state
```

Repository Configuration

```
# Add custom repository
cat > /etc/yum.repos.d/custom.repo << EOF
[custom]
name=Custom Repository
baseurl=https://example.com/repo
enabled=1
gpgcheck=1
gpgkey=https://example.com/GPG-KEY
EOF

# Enable EPEL repository
dnf install epel-release
dnf config-manager --enable epel
```

Common Tasks

```
# Install web server stack
dnf install -y httpd php php-mysqlnd mariadb-server
systemctl enable --now httpd mariadb

# Install development tools
dnf group install "Development Tools"
dnf install -y git vim

# Update system security patches only
dnf update --security

# Find and install package providing specific file
dnf provides */netstat
dnf install -y net-tools

# Clean up package cache
dnf clean all                        # Clean all cache
dnf autoremove                       # Remove orphaned packages
```

Troubleshooting

```
# Package issues
dnf check                           # Check for problems
dnf history                         # Show transaction history
dnf history undo 5                  # Undo transaction ID 5
dnf history info 5                  # Details of transaction 5

# Repository problems
dnf repolist -v                     # Verbose repository info
dnf makecache --refresh             # Force cache refresh
dnf config-manager --dump           # Show configuration

# Dependency issues
dnf install --allowerasing package  # Allow erasing conflicts
dnf install --best package          # Install best version available
dnf install --nobest package        # Allow suboptimal dependencies

# GPG key problems
rpm --import /path/to/GPG-KEY       # Import signing key
dnf install --nogpgcheck package    # Skip GPG verification (not recommended)
```

Common Pitfalls

- **WRONG**: Using `yum` commands → **RIGHT**: Use `dnf` in RHEL 10
- **WRONG**: Not updating before installing → **RIGHT**: Run `dnf update` regularly
- **WRONG**: Installing from untrusted sources → **RIGHT**: Verify GPG signatures
- **WRONG**: Mixing RPM and DNF operations → **RIGHT**: Use DNF for dependency management

---

Storage and LVM Management

Key Terms & Acronyms

- **LVM** - Logical Volume Manager (flexible disk management)
- **PV** - Physical Volume (physical disk/partition)
- **VG** - Volume Group (pool of PVs)
- **LV** - Logical Volume (usable storage from VG)
- **UUID** - Universally Unique Identifier (persistent device ID)
- **XFS** - X File System (RHEL 10 default)
- **ext4** - Fourth Extended Filesystem
- **GPT** - GUID Partition Table (modern partitioning)
- **MBR** - Master Boot Record (legacy, 2TB limit)
- **fsck** - File System Check

- **fstab** - File System Table
- **swap** - Virtual memory on disk
- **mount point** - Directory where filesystem is attached
- **PE** - Physical Extent (LVM allocation unit)

Key File Paths

```
/etc/fstab                    # Filesystem mount configuration
/dev/mapper/                  # Device mapper devices (LVM)
/dev/vg_name/lv_name          # Logical volume paths
/proc/mounts                  # Currently mounted filesystems
/proc/cmdline                 # Current kernel command line
/boot/                        # Boot files (kernels, initramfs)
```

Essential Commands

```
# Disk and partition management
lsblk                                 # Check available disks
blkid                                 # Show UUIDs and file systems
fdisk /dev/sdb                        # Create partitions
# fdisk commands: n (new), p (primary), 1 (partition number),
#                 t (change type), 8e (LVM), w (write)
partprobe                             # Re-read partition table

# LVM workflow
pvcreate /dev/sdb1 /dev/sdc1          # Create physical volumes
vgcreate vg_data /dev/sdb1 /dev/sdc1  # Create volume group
lvcreate -L 2G -n lv_database vg_data # Create logical volume
lvcreate -l 100%FREE -n lv_app vg_data # Use all remaining space

# LVM information
pvs; vgs; lvs                         # Show PV, VG, LV summary
pvdisplay; vgdisplay; lvdisplay       # Detailed information

# LVM extension
lvextend -L +1G /dev/vg_data/lv_database  # Extend LV
lvextend -l +100%FREE /dev/vg_data/lv_app # Use all free space

# Filesystem operations
mkfs.xfs /dev/vg_data/lv_database     # Create XFS filesystem
mkfs.ext4 /dev/vg_data/lv_app         # Create ext4 filesystem
mkdir /database /app                  # Create mount points
mount /dev/vg_data/lv_database /database

# Filesystem resize (after LV extension)
xfs_growfs /database                  # Grow XFS filesystem
resize2fs /dev/vg_data/lv_app         # Resize ext4 filesystem

# Verify changes
df -h /database                       # Check filesystem size
lsblk                                 # Verify block device structure
```

Swap Management

```
# Create and enable swap
mkswap /dev/sdd1                      # Create swap on partition
swapon /dev/sdd1                      # Activate swap
swapon -a                             # Activate all swap in fstab
swapon --show                         # Verify active swap
```

fstab Configuration

```
# Add entries to /etc/fstab for persistence
echo "/dev/vg_data/lv_database /database xfs defaults 0 2" >> /etc/fstab
echo "/dev/sdd1 swap swap defaults 0 0" >> /etc/fstab

# Test fstab entries
mount -a                              # Mount all in fstab
umount /database && mount /database   # Test specific mount
```

Common Tasks

```
# Complete LVM setup from scratch
lsblk                                 # Check available disks
fdisk /dev/sdb                        # Create partition (n,p,1,enter,enter,t,8e,w)
partprobe
pvcreate /dev/sdb1
vgcreate vg_data /dev/sdb1
lvcreate -L 2G -n lv_app vg_data
mkfs.xfs /dev/vg_data/lv_app
mkdir /app
mount /dev/vg_data/lv_app /app
echo "/dev/vg_data/lv_app /app xfs defaults 0 2" >> /etc/fstab
mount -a                              # Test fstab

# Extend existing LV and filesystem
lvextend -L +1G /dev/vg_data/lv_app
xfs_growfs /app                       # For XFS
# OR resize2fs /dev/vg_data/lv_app    # For ext4
df -h /app                            # Verify new size
```

Troubleshooting

```
# Storage issues diagnostic
df -h                                 # Check disk space
lsblk                                 # Check block devices
mount | grep target_mount             # Check mount status
mount -a                              # Test fstab syntax
pvs; vgs; lvs                         # Check LVM status

# Filesystem problems
fsck -n /dev/device                   # Check filesystem (ext4)
xfs_repair -n /dev/device             # Check XFS filesystem
lsof +D /mountpoint                   # See what's using mountpoint
fuser -mv /mountpoint                 # Alternative to lsof
```

Common Pitfalls

- **WRONG**: Using `resize2fs` for XFS → **RIGHT**: Use `xfs_growfs` for XFS

- **WRONG**: Forgetting partition type → **RIGHT**: Set type `8e` for LVM in fdisk
- **WRONG**: Not testing fstab → **RIGHT**: Always `mount -a` before reboot
- **WRONG**: Forgetting `partprobe` → **RIGHT**: Run after partition changes

---

Network Configuration

Key Terms & Acronyms

- **NetworkManager** - Primary network service in RHEL
- **nmcli** - NetworkManager CLI
- **nmtui** - NetworkManager TUI
- **DHCP** - Dynamic Host Configuration Protocol
- **DNS** - Domain Name System
- **IP** - Internet Protocol (IPv4/IPv6)
- **TCP** - Transmission Control Protocol
- **UDP** - User Datagram Protocol
- **CIDR** - Classless Inter-Domain Routing (/24 notation)
- **MAC** - Media Access Control address
- **gateway** - Default route for external networks
- **interface** - Network device (eth0, ens33, etc.)
- **connection** - NetworkManager configuration profile

Key File Paths

```
/etc/NetworkManager/          # NetworkManager configuration
/etc/resolv.conf              # DNS configuration
/etc/hosts                    # Local hostname resolution
/etc/services                 # Internet network services list
/etc/hostname                 # System hostname
/etc/sysconfig/network-scripts/  # Legacy network config (RHEL 8)
```

Essential Commands

```
# Network information
ip addr show                          # Show IP addresses
ip link show                          # Show network interfaces
ip route show                         # Show routing table
nmcli device status                   # NetworkManager device status
nmcli connection show                 # Show connections

# Static IP configuration (step-by-step)
nmcli con show                        # List connections
nmcli con modify "System eth0" ipv4.method manual
nmcli con modify "System eth0" ipv4.addresses "192.168.1.100/24"
nmcli con modify "System eth0" ipv4.gateway "192.168.1.1"
nmcli con modify "System eth0" ipv4.dns "8.8.8.8,8.8.4.4"
nmcli con modify "System eth0" autoconnect yes
nmcli con up "System eth0"

# Create new connection from scratch
nmcli con add type ethernet con-name "static-eth0" ifname eth0 \
  ip4 192.168.1.100/24 gw4 192.168.1.1
nmcli con modify "static-eth0" ipv4.dns "8.8.8.8,8.8.4.4"
nmcli con up "static-eth0"

# Reset to DHCP
nmcli con modify "System eth0" ipv4.method auto
nmcli con modify "System eth0" ipv4.addresses ""
nmcli con modify "System eth0" ipv4.gateway ""
nmcli con modify "System eth0" ipv4.dns ""
nmcli con up "System eth0"
```

Network Testing

```
# Connectivity testing
ping -c 3 192.168.1.1               # Test gateway
ping -c 3 8.8.8.8                   # Test external IP
ping -c 3 google.com                # Test DNS resolution
traceroute 8.8.8.8                  # Trace network path

# DNS testing
nslookup google.com                  # Basic DNS lookup
dig google.com                       # Detailed DNS lookup
dig @8.8.8.8 google.com             # Query specific DNS server
```

Common Tasks

```
# Configure static IP from scratch
nmcli con add type ethernet con-name "server" ifname ens33 \
  ip4 192.168.1.50/24 gw4 192.168.1.1
nmcli con modify "server" ipv4.dns "192.168.1.1,8.8.8.8"
nmcli con up "server"
ping -c 3 8.8.8.8                    # Verify connectivity
```

Troubleshooting

```
# Network connectivity issues (layer-by-layer)
ip link show                          # Physical layer
ip addr show                          # Network layer
ip route show                         # Routing layer
ping 127.0.0.1                       # Loopback test
ping gateway_ip                       # Gateway test
ping 8.8.8.8                         # External connectivity
nslookup hostname                     # DNS resolution

# NetworkManager troubleshooting
nmcli device status                   # Check device status
nmcli connection show                 # Check connections
nmcli con up "connection_name"        # Activate connection
systemctl status NetworkManager       # Check NetworkManager service
```

Common Pitfalls

- **WRONG**: Forgetting to activate connection → **RIGHT**: Always `nmcli con up` after changes
- **WRONG**: Not setting method to manual → **RIGHT**: Set `ipv4.method manual` for static IP
- **WRONG**: Configuring without checking interface name → **RIGHT**: Check `ip link show` first

---

SELinux Management

Key Terms & Acronyms

- **SELinux** - Security-Enhanced Linux (MAC system)
- **MAC** - Mandatory Access Control
- **DAC** - Discretionary Access Control
- **context** - SELinux security label (user:role:type:level)
- **type** - Most important part of context for RHCSA
- **boolean** - SELinux on/off policy switches
- **AVC** - Access Vector Cache (denial logs)
- **MLS** - Multi-Level Security

- **MCS** - Multi-Category Security
- **targeted** - Default SELinux policy
- **enforcing** - SELinux blocking violations
- **permissive** - SELinux logging only
- **disabled** - SELinux completely off
- **relabel** - Reapply correct SELinux contexts

Key File Paths

```
/etc/selinux/config           # SELinux mode configuration
/var/log/audit/audit.log      # SELinux audit logs
/var/lib/selinux/             # SELinux policy files
/etc/selinux/targeted/        # Targeted policy files
```

Essential Commands

```
# SELinux status and modes
getenforce                          # Current mode
sestatus                            # Detailed status
setenforce 0                        # Set permissive (temporary)
setenforce 1                        # Set enforcing (temporary)

# Persistent configuration (EXAM TIP: grubby commands are IN the config file!)
vi /etc/selinux/config              # Contains helpful grubby examples in comments
# SELINUX=enforcing|permissive|disabled
# For RHEL 10: disabled only unloads policy, doesn't fully disable SELinux
# To fully disable: grubby --update-kernel ALL --args selinux=0
# To re-enable:     grubby --update-kernel ALL --remove-args selinux

# Permanent mode changes
# Method 1: Edit configuration file
vim /etc/selinux/config             # Set SELINUX=enforcing|permissive|disabled

# Method 2: Using grubby (bootloader kernel parameters)
grubby --update-kernel=ALL --args="selinux=0"     # Disable SELinux at boot
grubby --update-kernel=ALL --remove-args="selinux=0"  # Remove disable parameter
grubby --update-kernel=ALL --args="enforcing=0"   # Boot into permissive mode
grubby --update-kernel=ALL --remove-args="enforcing=0"  # Remove permissive parameter

# Check current kernel parameters
grubby --info=DEFAULT               # Show default kernel info
cat /proc/cmdline                   # Show current boot parameters

# File contexts
ls -Z filename                      # Show file context
ps -eZ                              # Show process contexts
restorecon -Rv /path                # Restore default contexts
semanage fcontext -a -t httpd_exec_t "/web(/.*)?"
semanage fcontext -l                # List file contexts
semanage fcontext -d "/web(/.*)?"     # Delete context rule

# SELinux booleans
getsebool -a                        # List all booleans
getsebool httpd_can_network_connect   # Check specific boolean
setsebool -P httpd_can_network_connect on  # Set boolean permanently
setsebool httpd_can_network_connect on     # Set boolean temporarily

# Port contexts
semanage port -l                    # List port contexts
semanage port -a -t http_port_t -p tcp 8080  # Add port context
semanage port -d -p tcp 8080          # Delete port context
semanage port -l | grep http          # Show HTTP ports
```

Troubleshooting SELinux

```
# Essential ausearch commands (RED HAT OFFICIAL SYNTAX)
ausearch -m AVC -ts recent                                    # Recent AVC denials
ausearch -m AVC -ts today                                     # Today's denials
ausearch -m AVC,USER_AVC,SELINUX_ERR,USER_SELINUX_ERR -ts boot # All denials since boot
ausearch -c 'daemon_name' --raw | audit2allow -M my-daemon    # Generate policy for specific
daemon
ausearch -m AVC -ts recent | audit2allow -R                   # Readable policy suggestions

# Analysis and policy creation
sealert -a /var/log/audit/audit.log   # Analyze denials with recommendations
sealert -l UUID                       # Detailed denial analysis
semodule -X 300 -i my-policy.pp       # Install custom policy module (new syntax)

# If auditd not running, check dmesg
dmesg | grep -i -e type=1300 -e type=1400  # SELinux messages in kernel log
grep "SELinux is preventing" /var/log/messages  # setroubleshoot messages
```

Common Tasks

```
# Configure custom document root for Apache
mkdir -p /web/html
echo "<h1>Test</h1>" > /web/html/index.html
chown -R apache:apache /web/html
# Set SELinux contexts
semanage fcontext -a -t httpd_exec_t "/web(/.*)?"
restorecon -Rv /web
# Configure Apache document root in httpd.conf
systemctl restart httpd
# Test and check for denials
curl http://localhost/
ausearch -m AVC -ts recent

# Temporarily disable SELinux for troubleshooting
setenforce 0                          # Set permissive immediately
# Test your configuration
# If everything works, re-enable
setenforce 1

# Permanently disable SELinux (requires reboot)
grubby --update-kernel=ALL --args="selinux=0"
reboot
# Verify after reboot
getenforce                            # Should show "Disabled"

# Re-enable SELinux after disabling
grubby --update-kernel=ALL --remove-args="selinux=0"
vim /etc/selinux/config               # Set SELINUX=enforcing
reboot
# After reboot, may need to relabel filesystem
touch /.autorelabel                   # Force relabeling on next boot
reboot
```

Troubleshooting Workflow

```
# Step-by-step SELinux troubleshooting
getenforce                            # 1. Check mode
ausearch -m AVC -ts recent            # 2. Look for recent denials
sealert -a /var/log/audit/audit.log   # 3. Analyze denials
# 4. Apply suggested fixes (contexts/booleans/ports)
systemctl restart service_name       # 5. Restart service
ausearch -m AVC -ts recent            # 6. Verify no new denials
```

Common Pitfalls

- **WRONG**: Temporary boolean change `setsebool boolean on` → **RIGHT**: Use `-P` for permanent
- **WRONG**: Forgetting `restorecon` after context changes → **RIGHT**: Always run `restorecon -Rv`
- **WRONG**: Using wrong context type → **RIGHT**: Use `semanage fcontext -l | grep service` to find correct type

- **WRONG**: Disabling SELinux without reboot → **RIGHT**: `selinux=0` kernel parameter requires reboot
- **WRONG**: Re-enabling SELinux without relabeling → **RIGHT**: Use `touch /.autorelabel` before reboot
- **WRONG**: Editing GRUB config directly → **RIGHT**: Use `grubby` to modify kernel parameters

---

Firewall Management

Key Terms & Acronyms

- **firewalld** - Dynamic firewall daemon
- **zone** - Network security trust level
- **service** - Predefined firewall rule set
- **port** - Network communication endpoint
- **rich rule** - Complex firewall rules
- **masquerade** - NAT for outgoing connections
- **forward** - Pass traffic between interfaces
- **ICMP** - Internet Control Message Protocol (ping)
- **runtime** - Temporary configuration
- **permanent** - Persistent configuration

Key File Paths

```
/etc/firewalld/              # Firewall configuration files
/etc/firewalld/zones/        # Zone configuration files
/etc/firewalld/services/     # Service definitions
```

Essential Commands

```
# Firewall status and information
firewall-cmd --state                  # Check if firewalld is running
firewall-cmd --get-active-zones       # Show active zones
firewall-cmd --list-all               # Show all rules in default zone
firewall-cmd --list-services          # Show allowed services
firewall-cmd --list-ports             # Show allowed ports

# Service management
firewall-cmd --add-service=http --permanent      # Allow HTTP
firewall-cmd --add-service=https --permanent     # Allow HTTPS
firewall-cmd --add-service=ssh --permanent       # Allow SSH
firewall-cmd --remove-service=ssh --permanent    # Remove SSH
firewall-cmd --reload                            # Apply permanent changes

# Port management
firewall-cmd --add-port=8080/tcp --permanent     # Allow custom port
firewall-cmd --add-port=1000-1100/tcp --permanent # Port range
firewall-cmd --remove-port=8080/tcp --permanent   # Remove port
firewall-cmd --reload

# Zone management
firewall-cmd --get-default-zone       # Show default zone
firewall-cmd --set-default-zone=public  # Set default zone
firewall-cmd --zone=dmz --list-all     # Show specific zone rules
firewall-cmd --change-interface=eth1 --zone=dmz --permanent

# Rich rules (advanced)
firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.1.0/24" accept' --
permanent
firewall-cmd --add-rich-rule='rule family="ipv4" source address="10.0.0.5" port port="22"
protocol="tcp" accept' --permanent
```

Common Tasks

```
# Allow web server traffic
firewall-cmd --add-service=http --permanent
firewall-cmd --add-service=https --permanent
firewall-cmd --reload
firewall-cmd --list-services          # Verify

# Allow custom application on port 8080
firewall-cmd --add-port=8080/tcp --permanent
firewall-cmd --reload
ss -tuln | grep :8080                 # Verify port is listening

# Quick one-liners for common tasks
firewall-cmd --add-service=http --permanent && firewall-cmd --reload
firewall-cmd --add-port=443/tcp --permanent && firewall-cmd --reload
```

Troubleshooting

```
# Check if firewall is blocking service
firewall-cmd --list-all              # Check current rules
ss -tuln | grep :port                # Check if service is listening
telnet server_ip port                # Test connection from client
firewall-cmd --add-port=port/tcp --permanent  # Add rule if needed
firewall-cmd --reload
```

Common Pitfalls

- **WRONG**: Adding rule without `--permanent` → **RIGHT**: Always use `--permanent` and `--reload`
- **WRONG**: Forgetting to reload → **RIGHT**: Run `--reload` to apply permanent changes
- **WRONG**: Not verifying service is listening → **RIGHT**: Check with `ss -tuln | grep :port`

Services and Systemd Management

Key Terms & Acronyms

- **systemd** - System and service manager (PID 1)
- **unit** - Basic systemd object
- **service** - Daemon unit type (.service)
- **target** - Group of units (replaces runlevels)
- **socket** - IPC/network socket activation
- **timer** - Scheduled task unit (replaces cron)
- **daemon** - Background service process
- **PID** - Process Identifier
- **journal** - Systemd logging system
- **dependency** - Unit requirement/ordering
- **mask** - Prevent unit from starting
- **runlevel** - Legacy term (replaced by targets)

Key File Paths

```
/etc/systemd/system/          # Custom systemd unit files
/usr/lib/systemd/system/      # System-provided unit files
/var/log/journal/             # Systemd journal logs
/etc/systemd/journald.conf    # Journal configuration
```

Essential Commands

```
# Service lifecycle management
systemctl status httpd                # Check service status
systemctl start httpd                 # Start service
systemctl enable httpd                # Enable at boot
systemctl enable --now httpd          # Enable and start together
systemctl reload httpd                # Reload config without restart
systemctl restart httpd               # Full restart
systemctl disable httpd               # Disable at boot
systemctl stop httpd                  # Stop service

# Service verification
systemctl is-active httpd             # Check if running
systemctl is-enabled httpd            # Check if enabled
systemctl list-units --type=service   # List all services
systemctl --failed                    # List failed services
systemctl list-dependencies httpd     # Show dependencies

# Target management (runlevels)
systemctl get-default                 # Show current default target
systemctl set-default multi-user.target  # Set text mode default
systemctl set-default graphical.target   # Set GUI mode default
systemctl isolate rescue.target      # Switch to rescue mode
systemctl isolate multi-user.target  # Switch to multi-user mode

# Journal and logging
journalctl -u httpd                   # Service-specific logs
journalctl -u httpd -f                # Follow logs
journalctl -u httpd --since "1 hour ago"  # Recent logs
journalctl -p err                     # Error priority and above
journalctl -b                         # Current boot logs
journalctl --disk-usage               # Check journal space usage

# Make journal persistent
mkdir -p /var/log/journal
systemctl restart systemd-journald
```

Common Tasks

```
# Install and configure web server
dnf install -y httpd
systemctl enable --now httpd
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
curl http://localhost                # Test

# Set up database server
dnf install -y mariadb-server
systemctl enable --now mariadb
mysql_secure_installation
systemctl status mariadb
```

Troubleshooting Services

```
# Service startup troubleshooting workflow
systemctl status service_name         # 1. Check service status
systemctl is-enabled service_name     # 2. Check if enabled
journalctl -u service_name --no-pager # 3. Check detailed logs
systemctl list-dependencies service_name  # 4. Check dependencies
systemctl list-units --failed         # 5. Check for dependency failures

# Configuration validation
httpd -t                              # Test Apache config
sshd -t                              # Test SSH config
nginx -t                             # Test Nginx config

# Check for blocking issues
ausearch -m AVC -ts recent | grep service_name  # SELinux
firewall-cmd --list-all              # Firewall
ss -tuln | grep :port                # Port availability
```

Common Pitfalls

- **WRONG**: Starting service but not enabling → **RIGHT**: Use `systemctl enable --now`
- **WRONG**: Using restart when reload suffices → **RIGHT**: Use `reload` when possible
- **WRONG**: Not checking logs for errors → **RIGHT**: Always check `journalctl -u service`

Boot Process & GRUB Configuration

Key Terms & Acronyms

- **GRUB** - Grand Unified Bootloader (version 2)
- **UEFI** - Unified Extensible Firmware Interface

- **BIOS** - Basic Input/Output System (legacy)
- **initramfs** - Initial RAM filesystem
- **kernel** - Core operating system
- **dracut** - initramfs creation tool
- **grubby** - GRUB configuration tool
- **systemd targets** - Boot runlevels (multi-user, graphical, rescue)
- **kernel parameters** - Boot-time configuration options
- **MBR** - Master Boot Record (legacy boot)
- **GPT** - GUID Partition Table (modern)
- **ESP** - EFI System Partition

Key File Paths

```
/boot/grub2/grub.cfg          # GRUB configuration (auto-generated)
/etc/default/grub             # GRUB settings file
/etc/grub.d/                  # GRUB configuration scripts
/boot/loader/entries/         # Boot loader entries (systemd-boot)
/boot/initramfs-*.img         # Initial RAM filesystem images
/boot/vmlinuz-*               # Kernel images
/proc/cmdline                 # Current kernel command line
```

Essential Commands

```
# GRUB configuration management
grub2-mkconfig -o /boot/grub2/grub.cfg    # Regenerate GRUB config
grubby --default-kernel                   # Show default kernel
grubby --info=ALL                         # Show all kernel entries
grubby --set-default /boot/vmlinuz-*      # Set default kernel

# Kernel parameter management
grubby --update-kernel=ALL --args="quiet"           # Add parameter to all kernels
grubby --update-kernel=ALL --remove-args="quiet"    # Remove parameter from all kernels
grubby --update-kernel=DEFAULT --args="selinux=0"   # Add to default kernel only
grubby --info=DEFAULT                               # Show default kernel info

# Boot target management
systemctl get-default                      # Show current default target
systemctl set-default multi-user.target  # Set text mode as default
systemctl set-default graphical.target   # Set GUI mode as default
systemctl list-units --type=target       # List available targets
systemctl isolate rescue.target          # Switch to rescue mode immediately

# Emergency boot options (GRUB menu)
# Add these to kernel line in GRUB:
# rd.break                    # Break before root filesystem mount
# systemd.unit=rescue.target  # Boot to rescue mode
# systemd.unit=emergency.target # Boot to emergency mode
# selinux=0                   # Disable SELinux
# enforcing=0                 # Boot SELinux in permissive mode

# initramfs management
dracut --force                            # Rebuild initramfs for current kernel
dracut --force /boot/initramfs-$(uname -r).img $(uname -r)  # Specific kernel
lsinitrd                                  # List contents of initramfs
lsinitrd /boot/initramfs-$(uname -r).img # List specific initramfs
```

GRUB Configuration

```
# Edit GRUB defaults (/etc/default/grub)
GRUB_TIMEOUT=5                          # Boot menu timeout
GRUB_DEFAULT=saved                      # Remember last selection
GRUB_DISABLE_QUIET=true                # Show boot messages
GRUB_CMDLINE_LINUX_DEFAULT="quiet"     # Default kernel parameters
GRUB_CMDLINE_LINUX="crashkernel=auto"  # Always applied parameters

# After editing /etc/default/grub:
grub2-mkconfig -o /boot/grub2/grub.cfg  # Apply changes

# Custom GRUB entries (create in /etc/grub.d/40_custom)
menuentry "My Custom Boot" {
    linux /boot/vmlinuz-custom root=/dev/sda1 quiet
    initrd /boot/initramfs-custom.img
}
```

Boot Targets (Systemd)

```
# Common systemd targets
graphical.target     # Multi-user + GUI (runlevel 5)
multi-user.target    # Multi-user text mode (runlevel 3)
rescue.target        # Single user mode (runlevel 1)
emergency.target     # Minimal system
reboot.target        # Reboot system
poweroff.target      # Shutdown system

# Target management
systemctl isolate multi-user.target    # Switch target temporarily
systemctl set-default graphical.target # Set permanent default
systemctl get-default                  # Check current default
```

Password Recovery Procedure

```
# RHEL 10 Password Reset Steps:
# 1. Boot system and interrupt GRUB menu (press 'e')
# 2. Find linux line, add: rd.break
# 3. Press Ctrl+X to boot
# 4. At emergency prompt:
mount -o remount,rw /sysroot
chroot /sysroot
passwd root
touch /.autorelabel    # If SELinux enabled
exit
exit
# System will reboot and relabel filesystem
```

Common Tasks

```
# Change default boot target to text mode
systemctl set-default multi-user.target
systemctl get-default                   # Verify change

# Add kernel parameter permanently
grubby --update-kernel=ALL --args="intel_iommu=on"
grubby --info=DEFAULT | grep args       # Verify parameter added

# Create custom GRUB menu entry
cat >> /etc/grub.d/40_custom << 'EOF'
menuentry "RHEL 10 Debug Mode" {
    linux /boot/vmlinuz-$(uname -r) root=/dev/sda1 debug
    initrd /boot/initramfs-$(uname -r).img
}
EOF
grub2-mkconfig -o /boot/grub2/grub.cfg

# Rebuild initramfs after hardware changes
dracut --force --add-drivers "driver_name"
# Or for all modules:
dracut --force --regenerate-all
```

Boot Troubleshooting

```
# Boot process diagnosis
journalctl -b                           # Current boot logs
journalctl -b -1                        # Previous boot logs
journalctl -u systemd-logind          # Specific service at boot
dmesg                                   # Kernel ring buffer messages

# GRUB issues
grub2-install /dev/sda                # Reinstall GRUB bootloader
grub2-mkconfig -o /boot/grub2/grub.cfg # Regenerate configuration
efibootmgr -v                         # Show UEFI boot entries (UEFI systems)

# Kernel/initramfs problems
rpm -qa kernel                        # List installed kernels
ls -la /boot/                         # Check boot files exist
dracut --force                        # Rebuild current initramfs
grubby --remove-kernel=/boot/vmlinuz-old # Remove problematic kernel

# Emergency access methods
# Method 1: rd.break (password reset)
# Method 2: systemd.unit=rescue.target (rescue shell)
# Method 3: systemd.unit=emergency.target (minimal system)
# Method 4: Live CD/USB rescue mode
```

Common Pitfalls

- **WRONG**: Editing `/boot/grub2/grub.cfg` directly → **RIGHT**: Use `grub2-mkconfig` or `grubby`
- **WRONG**: Forgetting `/.autorelabel` after password reset → **RIGHT**: Always touch when SELinux enabled
- **WRONG**: Not regenerating GRUB config after changes → **RIGHT**: Run `grub2-mkconfig` after editing defaults
- **WRONG**: Using legacy GRUB commands → **RIGHT**: Use `grub2-*` commands in RHEL 10

Logging & System Monitoring

Key Terms & Acronyms

- **journald** - Systemd journal daemon
- **rsyslog** - System logging service
- **syslog** - System logging protocol
- **logrotate** - Log rotation utility
- **dmesg** - Kernel ring buffer messages
- **audit** - Linux audit framework
- **facility** - Syslog message category
- **priority** - Syslog message severity
- **persistent** - Journal storage on disk
- **volatile** - Journal storage in memory only
- **systemd-journald** - Journal service daemon
- **rsyslogd** - Remote system logging daemon

Key File Paths

```
/var/log/messages             # General system messages (rsyslog)
/var/log/secure              # Authentication and security messages
/var/log/audit/audit.log     # Audit framework logs
/var/log/journal/            # Persistent systemd journal
/run/log/journal/            # Volatile systemd journal
/etc/systemd/journald.conf   # Journal daemon configuration
/etc/rsyslog.conf           # Rsyslog main configuration
/etc/rsyslog.d/             # Rsyslog additional configurations
/etc/logrotate.conf         # Log rotation configuration
/etc/logrotate.d/           # Service-specific logrotate configs
```

Essential Commands

```
# Systemd Journal (journalctl)
journalctl                              # View all journal entries
journalctl -f                           # Follow journal (like tail -f)
journalctl -u httpd                     # Service-specific logs
journalctl -u httpd -f                  # Follow service logs
journalctl -p err                       # Priority level (emerg, alert, crit, err, warning,
notice, info, debug)
journalctl --since "1 hour ago"        # Time-based filtering
journalctl --since "2023-01-01" --until "2023-01-02"
journalctl -b                           # Current boot logs
journalctl -b -1                        # Previous boot logs
journalctl --disk-usage                 # Check journal disk usage
journalctl --vacuum-time=1month         # Clean logs older than 1 month
journalctl --vacuum-size=100M           # Limit journal size to 100MB

# Journal output formatting
journalctl -o json                      # JSON format
journalctl -o verbose                   # Verbose output
journalctl -o cat                       # Message text only
journalctl --no-pager                   # Don't use pager

# System information and monitoring
dmesg                                   # Kernel ring buffer
dmesg -w                               # Follow kernel messages
uptime                                  # System uptime and load
who                                     # Currently logged in users
w                                       # Detailed user activity
last                                    # Login history
lastlog                                 # Last login times

# Process monitoring
top                                     # Real-time process monitor
htop                                    # Enhanced process monitor (if installed)
ps aux                                 # Process snapshot
ps -ef                                 # Full format process list
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%mem  # Custom format, sorted by memory
pstree                                 # Process tree view
pgrep httpd                            # Find processes by name
pkill httpd                            # Kill processes by name

# System resource monitoring
free -h                                # Memory usage (human readable)
df -h                                  # Disk space usage
du -sh /var/log                        # Directory size
lsof                                   # List open files
lsof -u username                       # Files opened by user
lsof +D /path                          # Files in directory
ss -tuln                               # Network socket statistics
netstat -tuln                          # Network connections (legacy)
```

Performance Monitoring

```
# System performance tools
sar                                 # System activity reporter
sar -u 1 5                          # CPU usage, 1 second intervals, 5 times
sar -r 1 5                          # Memory usage
sar -d 1 5                          # Disk I/O
sar -n DEV 1 5                      # Network statistics
iostat                              # I/O statistics
iostat -x 1                         # Extended stats every second
vmstat                              # Virtual memory statistics
vmstat 1 5                          # 1 second intervals, 5 times
```

Log Configuration

```
# Make journal persistent
mkdir -p /var/log/journal
systemctl restart systemd-journald
# Or edit /etc/systemd/journald.conf:
# Storage=persistent

# Configure journal retention
echo "MaxRetentionSec=1month" >> /etc/systemd/journald.conf
echo "SystemMaxUse=100M" >> /etc/systemd/journald.conf
systemctl restart systemd-journald

# Rsyslog configuration examples
# In /etc/rsyslog.conf or /etc/rsyslog.d/custom.conf:
auth.*                  /var/log/auth.log     # Authentication messages
mail.*                  /var/log/mail.log     # Mail system messages
*.emerg                 :omusrmsg:*           # Emergency to all users
*.info;mail.none;authpriv.none  /var/log/messages  # General messages

# Custom log file for application
echo "local0.*    /var/log/myapp.log" > /etc/rsyslog.d/myapp.conf
systemctl restart rsyslog
# In application: logger -p local0.info "Application message"
```

Log Rotation

```
# Logrotate configuration
# Edit /etc/logrotate.d/myapp:
/var/log/myapp.log {
    daily           # Rotate daily
    missingok       # Don't error if log missing
    rotate 7        # Keep 7 old logs
    compress        # Compress old logs
    delaycompress   # Compress on next rotation
    notifempty      # Don't rotate empty logs
    copytruncate    # Copy and truncate original
    postrotate      # Commands after rotation
        systemctl reload myapp
    endscript
}

# Test logrotate configuration
logrotate -d /etc/logrotate.d/myapp     # Debug mode (dry run)
logrotate -f /etc/logrotate.d/myapp     # Force rotation
```

Common Tasks

```
# Monitor system startup issues
journalctl -b -p err                    # Boot errors
journalctl -u systemd-networkd -b       # Network service boot logs
systemctl --failed                      # Failed services

# Track user activity
journalctl _UID=1000                    # Logs for specific user ID
last -n 10                              # Last 10 logins
who -a                                  # All login information

# Monitor file access
lsof /var/log/messages                  # What's accessing log file
fuser -v /var/log/messages              # Alternative to lsof

# System performance baseline
sar -A > /tmp/system-baseline.txt       # Complete system activity report
vmstat 1 60 > /tmp/vmstat-1min.txt      # 1 minute of VM statistics

# Clean up logs
journalctl --vacuum-time=2weeks         # Keep 2 weeks of journal logs
find /var/log -name "*.gz" -mtime +30 -delete  # Remove compressed logs > 30 days
```

Troubleshooting with Logs

```
# Service won't start
systemctl status servicename           # Service status
journalctl -u servicename --no-pager   # Service logs
journalctl -u servicename --since "10 minutes ago"

# System performance issues
top                                    # Current resource usage
sar -u 1 10                           # CPU usage pattern
sar -r 1 10                           # Memory usage pattern
iostat -x 1 10                        # I/O patterns

# Authentication problems
grep "Failed password" /var/log/secure
grep "authentication failure" /var/log/secure
journalctl _SYSTEMD_UNIT=sshd.service -p warning

# Network connectivity issues
ss -tuln | grep :80                   # Check if web server listening
journalctl -u NetworkManager --since "1 hour ago"
dmesg | grep -i network               # Network-related kernel messages

# Boot problems
journalctl -b -p err                  # Current boot errors
journalctl -b -1 -p warning           # Previous boot warnings
dmesg | grep -i error                 # Kernel errors
```

Log Analysis Techniques

```
# Pattern matching and analysis
grep "ERROR" /var/log/messages | tail -20
grep -c "Failed password" /var/log/secure    # Count occurrences
awk '/Failed password/ {print $1, $2, $3, $11}' /var/log/secure  # Extract fields

# Time-based log analysis
journalctl --since "2023-12-01 09:00:00" --until "2023-12-01 17:00:00" -p err
grep "Dec 01 09:" /var/log/messages | grep ERROR

# Real-time monitoring
tail -f /var/log/messages | grep ERROR
journalctl -f | grep -i fail
multitail /var/log/messages /var/log/secure  # Multiple files simultaneously
```

Common Pitfalls

- **WRONG**: Using `tail -f` on journal files → **RIGHT**: Use `journalctl -f`
- **WRONG**: Not making journal persistent → **RIGHT**: Create `/var/log/journal` directory
- **WRONG**: Ignoring log rotation → **RIGHT**: Configure logrotate for custom logs

- **WRONG**: Not filtering logs by time/service → **RIGHT**: Use journalctl filters for efficiency

---

NFS and AutoFS

Key Terms & Acronyms

- **NFS** - Network File System (remote file sharing protocol, NFS 4.2 is the RHEL 10 default)
- **AutoFS** - Automatic File System (on-demand mounting service)
- **export** - Making shares available on NFS server
- **mount** - Accessing shares on NFS client
- **showmount** - Command to list NFS exports
- **exportfs** - Command to export NFS shares
- **automount** - AutoFS daemon process
- **direct map** - AutoFS mount on specific paths
- **indirect map** - AutoFS mount under common parent directory
- **master map** - Main AutoFS configuration file
- **wildcard** - Pattern matching in AutoFS maps (* and &)
- **_netdev** - Mount option indicating network dependency

Key File Paths

```
/etc/exports                # NFS server export configuration
/etc/fstab                  # Persistent mount configuration
/etc/auto.master            # AutoFS master map
/etc/auto.master.d/         # AutoFS user-defined maps directory
/etc/autofs.conf            # AutoFS service configuration
/etc/auto.misc              # Default indirect map for removable media
```

NFS Server Setup

```
# Install NFS server software
dnf install -y nfs-utils

# Create and configure share directory
mkdir -p /nfs-share
chmod 755 /nfs-share
chown nobody:nobody /nfs-share

# Configure exports (/etc/exports)
echo "/nfs-share *(rw,sync,no_root_squash)" >> /etc/exports
# Common export options:
# rw/ro                - read-write/read-only
# sync/async           - synchronous/asynchronous writes
# no_root_squash       - don't map root to nobody
# root_squash          - map root to nobody (default)
# all_squash          - map all users to nobody

# Export the shares
exportfs -avr                # Export all shares with verbose output
exportfs -v                  # Show current exports

# Start and enable NFS services
systemctl enable --now nfs-server
systemctl enable --now rpcbind

# Configure firewall
firewall-cmd --add-service=nfs --permanent
firewall-cmd --add-service=rpc-bind --permanent
firewall-cmd --add-service=mountd --permanent
firewall-cmd --reload
```

NFS Client Operations

```
# Install NFS client software
dnf install -y nfs-utils

# List available exports from server
showmount -e server.example.com
showmount -e 192.168.1.100

# Create mount point and mount manually
mkdir /mnt/nfs-data
mount -t nfs server.example.com:/nfs-share /mnt/nfs-data
mount -t nfs -o nfsvers=4.2 server:/nfs-share /mnt/nfs-data

# Test NFS mount
df -h /mnt/nfs-data
ls -la /mnt/nfs-data
echo "test file" > /mnt/nfs-data/testfile

# Unmount NFS share
umount /mnt/nfs-data
```

Persistent NFS Mounting via fstab

```
# Add entry to /etc/fstab for persistent mounting
echo "server.example.com:/nfs-share /mnt/nfs-data nfs defaults,_netdev 0 0" >> /etc/fstab

# Common NFS mount options:
# _netdev              - wait for network before mounting
# rw/ro               - read-write/read-only
# hard/soft           - retry behavior on server failure
# intr                - allow interruption of NFS calls
# rsize/wsize         - read/write buffer sizes
# timeo=n             - timeout value in deciseconds
# retrans=n           - number of retries

# Test fstab entry
umount /mnt/nfs-data
mount -a                      # Mount all fstab entries
mount /mnt/nfs-data          # Mount specific entry
```

AutoFS Configuration

Direct vs Indirect Maps Overview:

- **Indirect Maps**: Mount point is a directory that contains subdirectories for each share
- Master map: `/mnt/auto /etc/auto.nfs` → Access `/mnt/auto/sharename`
- Map file defines keys (subdirectory names) and their NFS locations
- **Direct Maps**: Each share has its own specific mount point anywhere in filesystem
- Master map: `/- /etc/auto.direct` → Access exact paths like `/shared-data`
- Map file defines full mount paths and their NFS locations

```
# Install AutoFS
dnf install -y autofs

# INDIRECT MAP CONFIGURATION
# Configure master map for indirect mapping
echo "/mnt/auto /etc/auto.nfs --timeout=60" >> /etc/auto.master

# Create indirect map file (/etc/auto.nfs)
# Format: key [options] location
# Key becomes subdirectory under /mnt/auto/
echo "shared -rw server.example.com:/nfs-share" > /etc/auto.nfs
echo "data -ro,intr server.example.com:/data" >> /etc/auto.nfs
# Results: /mnt/auto/shared and /mnt/auto/data will be available

# DIRECT MAP CONFIGURATION
# Configure master map for direct mapping
echo "/- /etc/auto.direct" >> /etc/auto.master

# Create direct map file (/etc/auto.direct)
# Format: full-mount-point [options] location
echo "/shared-data -rw server.example.com:/nfs-share" > /etc/auto.direct
echo "/company-files -ro server.example.com:/company" >> /etc/auto.direct
# Results: /shared-data and /company-files will be available directly

# Start and enable AutoFS service
systemctl enable --now autofs

# Verify AutoFS operation
systemctl status autofs

# Test indirect maps (subdirectories under mount point)
ls /mnt/auto                  # Shows available keys: shared, data
cd /mnt/auto/shared          # Triggers mount of server.example.com:/nfs-share
ls /mnt/auto/data            # Triggers mount of server.example.com:/data

# Test direct maps (specific filesystem paths)
ls /shared-data              # Triggers mount at exact path
ls /company-files            # Triggers mount at exact path

mount | grep autofs          # Show active automounts
```

AutoFS Wildcards for User Home Directories

```
# Wildcard mapping in indirect map
# * matches subdirectory name, & substitutes server/path
echo "* -rw server.example.com:/home/&" > /etc/auto.home

# Master map entry for home directories
echo "/home /etc/auto.home" >> /etc/auto.master

# This automatically mounts server.example.com:/home/username
# when user accesses /home/username
```

Common NFS and AutoFS Tasks

```
# Set up NFS client with AutoFS indirect map
showmount -e nfs-server              # List available shares
mkdir -p /mnt/auto                   # Create mount point
echo "/mnt/auto /etc/auto.nfs" >> /etc/auto.master
echo "share1 -rw nfs-server:/export/share1" > /etc/auto.nfs
systemctl enable --now autofs
ls /mnt/auto/share1                  # Triggers mount

# Set up AutoFS direct map
echo "/- /etc/auto.direct" >> /etc/auto.master
echo "/shared-data -rw nfs-server:/data" > /etc/auto.direct
systemctl reload autofs
ls /shared-data                      # Triggers mount

# Monitor AutoFS activity
tail -f /var/log/messages | grep automount
systemctl status autofs
mount | grep autofs
```

Troubleshooting NFS and AutoFS

```
# NFS server troubleshooting
exportfs -v                          # Show active exports
rpcinfo -p                          # Show RPC services
systemctl status nfs-server rpcbind
showmount -e localhost               # Test local exports
firewall-cmd --list-services         # Verify firewall rules

# NFS client troubleshooting
showmount -e server                  # Test server connectivity
rpcinfo -p server                   # Check server RPC services
mount -v -t nfs server:/share /mnt   # Verbose mount
ping server                         # Basic connectivity
telnet server 2049                  # Test NFS port

# AutoFS troubleshooting
systemctl status autofs             # Check service status
automount -f -v                     # Run in foreground with verbose
tail -f /var/log/messages           # Watch AutoFS log messages
ls -la /etc/auto.*                  # Verify map file permissions
mount | grep autofs                 # Show active automounts

# Restart AutoFS after configuration changes
systemctl reload autofs             # Reload maps without restart
systemctl restart autofs            # Full restart if needed
```

Common Pitfalls

- **WRONG**: Forgetting `_netdev` in fstab → **RIGHT**: Always use `_netdev` for network filesystems

- **WRONG**: Using AutoFS and fstab together → **RIGHT**: Choose either AutoFS or fstab, not both
- **WRONG**: Not starting rpcbind service → **RIGHT**: Ensure rpcbind runs before nfs-server
- **WRONG**: Incorrect firewall rules → **RIGHT**: Allow nfs, rpc-bind, and mountd services
- **WRONG**: Forgetting to export after editing /etc/exports → **RIGHT**: Run `exportfs -ra` after changes
- **WRONG**: AutoFS maps with wrong permissions → **RIGHT**: Ensure map files are readable by autofs

---

Flatpak Software Management

Key Terms & Acronyms

- **Flatpak** - Application distribution framework with sandboxing
- **Flathub** - Largest public Flatpak repository (flathub.org)
- **remote** - Flatpak repository (similar to DNF repo)
- **runtime** - Shared base libraries used by multiple Flatpak apps
- **application** - Sandboxed software package
- **OSTree** - Content-addressable storage system used by Flatpak
- **sandbox** - Isolated execution environment for Flatpak apps
- **portal** - D-Bus interface for controlled host access from sandbox
- **app ID** - Reverse-DNS application identifier (e.g., org.gimp.GIMP)

Key File Paths

```
/var/lib/flatpak/                    # System-wide Flatpak installations
~/.local/share/flatpak/              # User Flatpak installations
/etc/flatpak/remotes.d/              # System-wide remote configuration
/var/lib/flatpak/overrides/          # System permission overrides
~/.local/share/flatpak/overrides/    # User permission overrides
```

Essential Commands

```
# Remote (repository) management
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak remote-add --user flathub https://flathub.org/repo/flathub.flatpakrepo  # User only
flatpak remote-delete flathub                    # Remove remote
flatpak remotes                                  # List configured remotes
flatpak remotes --show-details                   # Detailed remote info
flatpak remote-ls flathub                        # List apps in remote

# Search and information
flatpak search keyword                           # Search for applications
flatpak info org.example.App                     # Show app details

# Install and uninstall
flatpak install flathub org.example.App          # Install from remote
flatpak install --user flathub org.example.App   # Install for user only
flatpak uninstall org.example.App                # Uninstall application
flatpak uninstall --unused                       # Remove unused runtimes

# List, run, and update
flatpak list --app                               # List installed apps
flatpak list --runtime                           # List installed runtimes
flatpak run org.example.App                      # Run application
flatpak update                                   # Update all Flatpaks
```

Common Tasks

```
# Set up Flathub and install application
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak install flathub org.gnome.Calculator -y
flatpak list --app                               # Verify
flatpak run org.gnome.Calculator                 # Test

# User-level install (no root needed)
flatpak remote-add --user --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak install --user flathub org.mozilla.firefox -y
flatpak list --user --app                        # Verify

# Clean up after uninstalls
flatpak uninstall --unused -y                    # Remove orphaned runtimes
```

Permission Overrides

```
# Grant filesystem access to sandboxed app
flatpak override --user --filesystem=home org.example.App
flatpak override --user --show org.example.App    # View overrides
flatpak override --user --reset org.example.App   # Reset to defaults
```

Common Pitfalls

- **WRONG**: Trying to install without adding remote first → **RIGHT**: Add remote with `flatpak remote-add` before installing
- **WRONG**: Forgetting `--if-not-exists` → **RIGHT**: Use it to make commands idempotent
- **WRONG**: Not cleaning up runtimes → **RIGHT**: Run `flatpak uninstall --unused` after removing apps

---

Scheduled Tasks & Automation

Key Terms & Acronyms

- **cron** - Time-based job scheduler
- **crontab** - Cron table (user's scheduled jobs)
- **crond** - Cron daemon
- **anacron** - Cron for systems not always on
- **systemd timers** - Modern alternative to cron
- **at** - One-time scheduled task
- **batch** - Queue jobs when system load permits
- **systemd.timer** - Timer unit file
- **systemd.service** - Service unit file
- **OnCalendar** - Timer schedule specification
- **Persistent** - Timer survives system shutdown

Key File Paths

```
/etc/crontab                # System-wide cron table
/etc/cron.d/                # System cron job directory
/etc/cron.daily/            # Daily cron jobs
/etc/cron.weekly/           # Weekly cron jobs
/etc/cron.monthly/          # Monthly cron jobs
/etc/cron.hourly/           # Hourly cron jobs
/var/spool/cron/            # User crontabs
/etc/cron.allow             # Users allowed to use cron
/etc/cron.deny              # Users denied cron access
/etc/at.allow               # Users allowed to use at
/etc/at.deny                # Users denied at access
/var/spool/at/              # At job queue
/etc/anacrontab             # Anacron configuration
```

Essential Commands

```
# Crontab management
crontab -e                          # Edit user's crontab
crontab -l                          # List user's crontab
crontab -r                          # Remove user's crontab
crontab -l -u username              # List another user's crontab (root only)
crontab -e -u username              # Edit another user's crontab (root only)

# At command (one-time scheduling)
at 15:30                            # Schedule job for 3:30 PM today
at 15:30 tomorrow                   # Schedule for tomorrow
at 15:30 12/25/2023                 # Specific date
at now + 5 minutes                  # Relative time
atq                                 # List pending at jobs
atrm job_number                     # Remove at job
at -f script.sh now + 1 hour        # Run script file

# Batch command (load-dependent scheduling)
batch                               # Schedule when load average < 1.5
batch -f script.sh                  # Schedule script file

# Systemd timers
systemctl list-timers               # List active timers
systemctl list-timers --all         # List all timers
systemctl status timer_name         # Timer status
systemctl enable --now timer_name   # Enable and start timer
systemctl disable timer_name        # Disable timer
```

Cron Syntax

```
# Crontab format: minute hour day month weekday command
# * * * * * command
# ┬ ┬ ┬ ┬ ┬
# │ │ │ │ │
# │ │ │ │ └── Weekday (0-7, 0=Sunday)
# │ │ │ └──── Month (1-12)
# │ │ └────── Day (1-31)
# │ └──────── Hour (0-23)
# └────────── Minute (0-59)

# Special values:
# *           Any value
# ,           List separator (1,3,5)
# -           Range (1-5)
# /           Step values (*/5 = every 5)

# Examples:
0 2 * * *           # Daily at 2:00 AM
30 14 * * 1-5       # Weekdays at 2:30 PM
0 */4 * * *         # Every 4 hours
15 3 1 * *          # First day of month at 3:15 AM
0 6 * * 0           # Sundays at 6:00 AM
*/10 * * * *        # Every 10 minutes

# Special keywords (system cron only):
@reboot             # At startup
@yearly or @annually # Once a year (0 0 1 1 *)
@monthly            # Once a month (0 0 1 * *)
@weekly             # Once a week (0 0 * * 0)
@daily or @midnight # Once a day (0 0 * * *)
@hourly             # Once an hour (0 * * * *)
```

Systemd Timers

```
# Create timer unit file (/etc/systemd/system/backup.timer)
[Unit]
Description=Daily backup timer
Requires=backup.service

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target

# Create corresponding service file (/etc/systemd/system/backup.service)
[Unit]
Description=Daily backup service
Wants=backup.timer

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

[Install]
WantedBy=multi-user.target

# OnCalendar examples:
OnCalendar=*-*-* 02:00:00      # Daily at 2:00 AM
OnCalendar=Mon *-*-* 09:00:00  # Mondays at 9:00 AM
OnCalendar=*-*-01 00:00:00     # First of month at midnight
OnCalendar=*-01,07 00:00:00    # January and July 1st
OnCalendar=hourly              # Every hour
OnCalendar=*:0/15              # Every 15 minutes
```

Common Tasks

```
# Daily log rotation at 3:00 AM
echo "0 3 * * * /usr/sbin/logrotate /etc/logrotate.conf" | crontab -

# Weekly system update on Sundays at 2:00 AM
echo "0 2 * * 0 /usr/bin/dnf update -y" > /etc/cron.d/system-update

# Backup script every 6 hours
echo "0 */6 * * * /home/user/backup.sh" | crontab -

# Send system report monthly
echo "0 9 1 * * /usr/local/bin/system-report.sh | mail admin@example.com" >> /etc/crontab

# One-time maintenance in 2 hours
echo "/usr/local/bin/maintenance.sh" | at now + 2 hours

# Create systemd timer for daily backup
cat > /etc/systemd/system/daily-backup.timer << 'EOF'
[Unit]
Description=Daily backup timer

[Timer]
OnCalendar=daily
Persistent=true

[Install]
WantedBy=timers.target
EOF

cat > /etc/systemd/system/daily-backup.service << 'EOF'
[Unit]
Description=Daily backup service

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

systemctl enable --now daily-backup.timer
```

Environment and Output Handling

```
# Cron environment variables (in crontab)
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@example.com

# Redirect output
0 2 * * * /script.sh > /var/log/script.log 2>&1    # Log output
0 2 * * * /script.sh >/dev/null 2>&1               # Suppress output
0 2 * * * /script.sh 2>&1 | logger -t "script"    # Send to syslog

# Environment in scripts
#!/bin/bash
# Set PATH and other variables explicitly in scripts
export PATH=/usr/local/bin:/usr/bin:/bin
cd /home/user
./script.sh
```

Access Control

```
# Allow specific users to use cron
echo "alice" >> /etc/cron.allow
echo "bob" >> /etc/cron.allow

# Deny specific users (if no cron.allow exists)
echo "baduser" >> /etc/cron.deny

# At command access control
echo "alice" >> /etc/at.allow
echo "bob" >> /etc/at.allow
```

Troubleshooting

```
# Check cron service status
systemctl status crond
journalctl -u crond                   # Cron daemon logs

# Verify cron jobs are running
grep CRON /var/log/cron              # Cron execution logs (if rsyslog configured)
journalctl -u crond --since "1 hour ago"

# Check at jobs
atq                                   # List pending jobs
ls -la /var/spool/at/                # At job files
systemctl status atd                 # At daemon status

# Timer troubleshooting
systemctl list-timers --failed       # Failed timers
systemctl status timer_name          # Specific timer status
journalctl -u timer_name             # Timer logs

# Common issues
# 1. Script not executable
chmod +x /path/to/script.sh

# 2. Wrong PATH in cron environment
# Add full paths or set PATH in crontab

# 3. Missing MAILTO causes local mail accumulation
# Set MAILTO="" to suppress or configure proper email

# 4. Scripts expecting interactive shell
# Use absolute paths and set environment variables
```

Monitoring Scheduled Tasks

```
# List all active cron jobs
for user in $(cut -f1 -d: /etc/passwd); do
    echo "User: $user"
    crontab -l -u $user 2>/dev/null
done

# Monitor cron execution
tail -f /var/log/cron                 # If rsyslog configured for cron
journalctl -u crond -f               # Follow cron daemon logs

# Check systemd timer last run
systemctl list-timers                # Shows next run and last run times
systemctl status timer_name          # Detailed timer status
```

Common Pitfalls

- **WRONG**: Using relative paths in cron jobs → **RIGHT**: Use absolute paths always
- **WRONG**: Not setting proper environment → **RIGHT**: Set PATH and other variables in crontab
- **WRONG**: Forgetting output redirection → **RIGHT**: Redirect stdout/stderr appropriately
- **WRONG**: Not making scripts executable → **RIGHT**: Use `chmod +x` on script files
- **WRONG**: Using interactive commands → **RIGHT**: Use non-interactive flags and options

---

Emergency Recovery Procedures

Key Terms & Acronyms

- **rd.break** - Kernel parameter for root password reset
- **rescue** - Systemd target for system recovery
- **emergency** - Minimal systemd target
- **chroot** - Change root directory
- **autorelabel** - Force SELinux relabeling
- **single** - Single user mode (legacy)
- **fsck** - File system check
- **grubby** - GRUB configuration tool

Boot Issues and Recovery

```
# GRUB rescue (password reset)
# At GRUB menu: e -> linux line -> add rd.break
# mount -o remount,rw /sysroot
# chroot /sysroot
# passwd root
# touch /.autorelabel  # If SELinux enabled
# exit; exit

# Systemd rescue mode
# At GRUB: add systemd.unit=rescue.target

# Emergency mode (minimal system)
# At GRUB: add systemd.unit=emergency.target

# Boot with SELinux disabled
# At GRUB: add selinux=0
```

Critical File Recovery

```
# Corrupted fstab
# Boot to rescue mode
# mount -o remount,rw /
# cp /etc/fstab.backup /etc/fstab  # If backup exists
# Or manually fix entries

# Corrupted sudoers
# Boot to rescue mode as root
# visudo  # Fix syntax
# Or: pkexec visudo  # If available
```

SSH & Remote Access

Key Terms & Acronyms

- **SSH** - Secure Shell (encrypted remote access protocol)
- **sshd** - SSH daemon (server process)
- **RSA** - Rivest-Shamir-Adleman (key algorithm)
- **DSA** - Digital Signature Algorithm (legacy key type)
- **ECDSA** - Elliptic Curve Digital Signature Algorithm
- **Ed25519** - Modern elliptic curve signature scheme
- **public key** - Cryptographic key for encryption/verification
- **private key** - Secret key for decryption/signing
- **known_hosts** - File storing server public keys
- **authorized_keys** - File storing allowed client public keys
- **port forwarding** - Tunnel network connections through SSH
- **Wayland forwarding** - Remote GUI application display (RHEL 10 uses Wayland, not X11)

Key File Paths

```
/etc/ssh/sshd_config          # SSH daemon configuration
/etc/ssh/ssh_config           # System-wide client configuration
~/.ssh/config                 # User SSH client configuration
~/.ssh/authorized_keys        # Allowed public keys for this user
~/.ssh/known_hosts            # Verified server public keys
~/.ssh/id_rsa                 # User's private key (RSA)
~/.ssh/id_rsa.pub             # User's public key (RSA)
~/.ssh/id_ed25519             # User's private key (Ed25519)
~/.ssh/id_ed25519.pub         # User's public key (Ed25519)
/var/log/secure               # SSH authentication logs
```

Essential Commands

```
# SSH connection basics
ssh user@hostname                     # Connect to remote host
ssh -p 2222 user@hostname            # Connect to custom port
ssh -l username hostname              # Alternative user specification
ssh user@hostname command             # Execute single command
ssh -t user@hostname 'sudo command'   # Force pseudo-terminal allocation

# Key generation
ssh-keygen -t rsa -b 4096             # Generate RSA 4096-bit key
ssh-keygen -t ed25519                 # Generate Ed25519 key (recommended)
ssh-keygen -t rsa -b 4096 -C "comment" -f ~/.ssh/custom_key  # Custom key
ssh-keygen -y -f ~/.ssh/id_rsa        # Display public key from private key
ssh-keygen -l -f ~/.ssh/id_rsa.pub    # Show key fingerprint

# Key distribution
ssh-copy-id user@hostname             # Copy public key to remote host
ssh-copy-id -i ~/.ssh/custom_key.pub user@hostname  # Copy specific key
scp ~/.ssh/id_rsa.pub user@hostname:~/.ssh/authorized_keys  # Manual copy

# File transfer
scp file.txt user@hostname:/path/     # Copy file to remote host
scp user@hostname:/path/file.txt .    # Copy file from remote host
scp -r directory user@hostname:/path  # Copy directory recursively
scp -P 2222 file.txt user@hostname:/path  # Custom port

# Advanced SSH features
ssh -L 8080:localhost:80 user@hostname     # Local port forwarding
ssh -R 8080:localhost:80 user@hostname     # Remote port forwarding
ssh -D 1080 user@hostname                  # SOCKS proxy
ssh -X user@hostname                       # X11/Wayland forwarding (GUI apps)
ssh -N -f -L 8080:localhost:80 user@hostname  # Background tunnel
```

SSH Configuration

```
# Server configuration (/etc/ssh/sshd_config)
Port 22                         # Default SSH port
PermitRootLogin no              # Disable root login
PasswordAuthentication no       # Disable password auth (keys only)
PubkeyAuthentication yes        # Enable public key auth
AuthorizedKeysFile .ssh/authorized_keys  # Location of authorized keys
MaxAuthTries 3                  # Maximum authentication attempts
ClientAliveInterval 300         # Keep connection alive (seconds)
ClientAliveCountMax 2           # Maximum client alive messages
AllowUsers alice bob            # Restrict users
DenyUsers baduser               # Deny specific users
AllowGroups sshusers            # Allow specific groups

# After editing sshd_config:
sshd -t                        # Test configuration syntax
systemctl reload sshd         # Apply changes

# Client configuration (~/.ssh/config)
Host webserver
    HostName 192.168.1.100
    User admin
    Port 2222
    IdentityFile ~/.ssh/webserver_key

Host *.example.com
    User myusername
    IdentityFile ~/.ssh/company_key

Host bastion
    HostName bastion.example.com
    User admin
    ProxyJump jumphost.example.com
```

Key-Based Authentication Setup

```
# Complete key-based authentication setup
# 1. Generate key pair on client
ssh-keygen -t ed25519 -C "user@client"

# 2. Copy public key to server (method 1 - preferred)
ssh-copy-id user@server

# 3. Or manually copy public key (method 2)
cat ~/.ssh/id_ed25519.pub | ssh user@server "mkdir -p ~/.ssh && cat >> ~/.ssh/authorized_keys"

# 4. Set proper permissions on server
ssh user@server "chmod 700 ~/.ssh && chmod 600 ~/.ssh/authorized_keys"

# 5. Test key-based login
ssh user@server

# 6. Disable password authentication (optional)
# Edit /etc/ssh/sshd_config: PasswordAuthentication no
sudo systemctl reload sshd
```

SSH Security Hardening

```
# Change default port
sed -i 's/^#Port 22/Port 2222/' /etc/ssh/sshd_config

# Disable root login
sed -i 's/^#PermitRootLogin yes/PermitRootLogin no/' /etc/ssh/sshd_config

# Disable password authentication (use keys only)
sed -i 's/^#PasswordAuthentication yes/PasswordAuthentication no/' /etc/ssh/sshd_config

# Enable fail2ban for SSH protection
dnf install -y fail2ban
systemctl enable --now fail2ban
# Configure in /etc/fail2ban/jail.local

# Configure firewall
firewall-cmd --remove-service=ssh --permanent    # Remove default SSH
firewall-cmd --add-port=2222/tcp --permanent     # Add custom port
firewall-cmd --reload

# Apply SSH configuration
sshd -t && systemctl reload sshd
```

Common Tasks

```
# Set up passwordless SSH between servers
ssh-keygen -t ed25519 -N "" -f ~/.ssh/server_key
ssh-copy-id -i ~/.ssh/server_key.pub user@target-server
ssh -i ~/.ssh/server_key user@target-server

# Create SSH tunnel for database access
ssh -N -f -L 5432:localhost:5432 user@database-server
# Now connect to localhost:5432 to reach remote database

# Execute commands on multiple servers
for host in server1 server2 server3; do
    ssh user@$host "uptime"
done

# Copy files between remote servers (via local machine)
scp user@server1:/path/file.txt user@server2:/path/

# Monitor SSH connections
ss -tuln | grep :22              # Check SSH listening
last | grep ssh                  # Recent SSH logins
journalctl -u sshd -f           # Follow SSH logs in real-time
```

Troubleshooting SSH

```
# Connection troubleshooting
ssh -v user@hostname             # Verbose output (debug)
ssh -vv user@hostname            # More verbose
ssh -vvv user@hostname           # Maximum verbosity

# Test SSH configuration
sshd -t                          # Test sshd config syntax
sshd -T                          # Dump effective configuration

# Check SSH service status
systemctl status sshd            # Service status
journalctl -u sshd --no-pager    # SSH daemon logs
grep "ssh" /var/log/secure       # SSH authentication attempts

# Key problems
ssh-keygen -R hostname           # Remove host from known_hosts
chmod 600 ~/.ssh/id_rsa          # Fix private key permissions
chmod 644 ~/.ssh/id_rsa.pub      # Fix public key permissions
chmod 700 ~/.ssh                # Fix .ssh directory permissions
chmod 600 ~/.ssh/authorized_keys # Fix authorized_keys permissions

# Connection issues
telnet hostname 22               # Test basic connectivity
nmap -p 22 hostname              # Check if port is open
firewall-cmd --list-all          # Check firewall rules
semanage port -l | grep ssh      # Check SELinux port contexts
```

SSH Agent and Key Management

```
# Start SSH agent
eval $(ssh-agent)                # Start agent and set environment
ssh-agent bash                   # Start new shell with agent

# Add keys to agent
ssh-add ~/.ssh/id_rsa          # Add specific key
ssh-add                        # Add default keys
ssh-add -l                     # List loaded keys
ssh-add -D                     # Remove all keys from agent

# Key forwarding
ssh -A user@hostname           # Enable agent forwarding
# Add to ~/.ssh/config:
# ForwardAgent yes

# Multiple key management
ssh-add ~/.ssh/personal_key
ssh-add ~/.ssh/work_key
ssh -i ~/.ssh/specific_key user@hostname  # Use specific key
```

Common Pitfalls

- **WRONG**: Using weak RSA 1024-bit keys → **RIGHT**: Use RSA 4096-bit or Ed25519 keys
- **WRONG**: Leaving default SSH port 22 → **RIGHT**: Change to non-standard port for security
- **WRONG**: Wrong file permissions on SSH keys → **RIGHT**: Use 600 for private keys, 644 for public keys
- **WRONG**: Not testing SSH config before applying → **RIGHT**: Always use `sshd -t` to test configuration
- **WRONG**: Disabling password auth without testing key auth → **RIGHT**: Test key authentication before disabling passwords

---

Shell Environment & Scripting Basics

Key Terms & Acronyms

- **bash** - Bourne Again Shell (default RHEL shell)
- **env** - Environment (shell variables and settings)
- **PATH** - Executable search path
- **PS1** - Primary prompt string
- **stdin** - Standard input (file descriptor 0)
- **stdout** - Standard output (file descriptor 1)
- **stderr** - Standard error (file descriptor 2)
- **shebang** - #! interpreter directive
- **exit status** - Return code of command (0=success)
- **alias** - Command shortcut
- **function** - Shell function definition
- **source/dot** - Execute script in current shell

Key File Paths

```
/etc/profile                 # System-wide shell initialization
/etc/profile.d/              # System-wide shell scripts
/etc/bashrc                  # System-wide bash configuration
~/.bash_profile              # User bash login script
~/.bashrc                    # User bash non-login script
~/.bash_logout               # User logout script
~/.bash_history              # Command history file
/etc/environment             # System environment variables
```

Essential Commands

```
# Environment variables
export VAR=value                      # Set and export variable
env                                   # Show all environment variables
printenv VAR                          # Show specific variable
unset VAR                            # Remove variable
echo $VAR                            # Display variable value
env VAR=value command                # Set variable for single command

# Command history
history                              # Show command history
history -c                           # Clear history
!!                                   # Repeat last command
!n                                   # Repeat command number n
!string                              # Repeat last command starting with string
^old^new                             # Replace and re-run last command

# Aliases and functions
alias ll='ls -la'                    # Create alias
alias                                # List all aliases
unalias ll                           # Remove alias
function myfunc() { echo $1; }       # Define function
type command                         # Show command type (alias/function/builtin)
which command                        # Show executable location

# Process substitution and pipes
command1 | command2                  # Pipe output to input
command > file                       # Redirect stdout to file
command >> file                      # Append stdout to file
command 2> file                      # Redirect stderr to file
command &> file                      # Redirect both stdout and stderr
command < file                       # Use file as stdin
command1 && command2                 # Run command2 if command1 succeeds
command1 || command2                 # Run command2 if command1 fails
command1; command2                   # Run commands sequentially
```

Basic Scripting Constructs

```
#!/bin/bash
# Shebang - interpreter specification

# Variables (no spaces around =)
name="John"
count=10
readonly PI=3.14159                  # Read-only variable

# Command substitution
current_date=$(date)                 # Modern syntax
current_date=`date`                  # Legacy syntax (avoid)

# Conditional statements
if [ condition ]; then
    commands
elif [ condition ]; then
    commands
else
    commands
fi

# Test conditions
[ -f file ]                          # File exists and is regular file
[ -d directory ]                     # Directory exists
[ -r file ]                          # File is readable
[ -w file ]                          # File is writable
[ -x file ]                          # File is executable
[ -z string ]                        # String is empty
[ -n string ]                        # String is not empty
[ string1 = string2 ]                # Strings are equal
[ $num1 -eq $num2 ]                 # Numbers are equal
[ $num1 -gt $num2 ]                 # num1 greater than num2

# Loops
for file in *.txt; do
    echo "Processing $file"
done

for i in {1..10}; do
    echo "Number: $i"
done

while [ condition ]; do
    commands
done

# Case statement
case $variable in
    pattern1)
        commands;;
    pattern2)
        commands;;
    *)
```

```
        default commands;;
esac
```

Positional Parameters and Special Variables

```
$0                              # Script name
$1, $2, $3...                   # Positional parameters (arguments)
$#                              # Number of arguments
$@                              # All arguments (individually quoted)
$*                              # All arguments (as single string)
$?                              # Exit status of last command
$$                              # Current process ID
shift                           # Shift positional parameters left ($2→$1)
```

Common Scripting Patterns

```
# Check if script has arguments
if [ $# -eq 0 ]; then
    echo "Usage: $0 <argument>"
    exit 1
fi

# Process script arguments
while [ $# -gt 0 ]; do
    case $1 in
        -h|--help)
            echo "Help message"
            exit 0;;
        -v|--verbose)
            verbose=true;;
        *)
            echo "Unknown option: $1"
            exit 1;;
    esac
    shift
done

# Function with return value
calculate_sum() {
    local num1=$1
    local num2=$2
    echo $((num1 + num2))
}
result=$(calculate_sum 5 3)

# Error handling
command || {
    echo "Command failed" >&2
    exit 1
}

# Log function
log() {
    echo "$(date): $*" >> /var/log/script.log
}
```

Environment Configuration

```
# System-wide environment (/etc/profile)
export PATH="/usr/local/bin:$PATH"
export EDITOR=vim
export HISTSIZE=1000

# User environment (~/.bashrc)
# Add to end of file:
export PATH="$HOME/bin:$PATH"
alias ll='ls -la'
alias grep='grep --color=auto'

# Load custom configurations
if [ -f ~/.bash_aliases ]; then
    source ~/.bash_aliases
fi

# Custom prompt
export PS1='\u@\h:\w\$ '            # user@host:directory$
export PS1='\[\e[32m\]\u@\h\[\e[0m\]:\[\e[34m\]\w\[\e[0m\]\$ '  # Colored

# Apply changes
source ~/.bashrc                     # Reload configuration
```

Common Tasks

```
# Create a backup script
#!/bin/bash
BACKUP_DIR="/backup"
SOURCE_DIR="/home/user"
TIMESTAMP=$(date +%Y%m%d_%H%M%S)

mkdir -p "$BACKUP_DIR"
tar -czf "$BACKUP_DIR/backup_$TIMESTAMP.tar.gz" "$SOURCE_DIR"
echo "Backup completed: backup_$TIMESTAMP.tar.gz"

# System monitoring script
#!/bin/bash
echo "=== System Status ==="
echo "Date: $(date)"
echo "Uptime: $(uptime)"
echo "Disk Usage:"
df -h
echo "Memory Usage:"
free -h
echo "Load Average:"
cat /proc/loadavg

# User management script
#!/bin/bash
create_user() {
    local username=$1
    if id "$username" &>/dev/null; then
        echo "User $username already exists"
        return 1
    fi
    useradd -m "$username"
    echo "User $username created"
}

# Process multiple users
for user in alice bob charlie; do
    create_user "$user"
done
```

Advanced Shell Features

```
# Parameter expansion
${var}                          # Variable value
${var:-default}                 # Use default if var is empty
${var:=default}                 # Set var to default if empty
${#var}                         # Length of variable
${var%pattern}                  # Remove shortest match from end
${var%%pattern}                 # Remove longest match from end
${var#pattern}                  # Remove shortest match from beginning
${var##pattern}                 # Remove longest match from beginning

# Arrays (basic)
arr=("item1" "item2" "item3")
echo ${arr[0]}                  # First element
echo ${arr[@]}                  # All elements
echo ${#arr[@]}                 # Array length

# Here documents
cat << 'EOF' > config.txt
Setting1=value1
Setting2=value2
EOF

# Job control
command &                       # Run in background
jobs                            # List background jobs
fg %1                           # Bring job 1 to foreground
bg %1                           # Send job 1 to background
nohup command &                 # Run command immune to hangups
```

Debugging and Troubleshooting

```
# Script debugging
bash -x script.sh                # Execute with debug output
bash -n script.sh                # Check syntax without execution
set -x                           # Enable debug mode in script
set +x                           # Disable debug mode

# Error handling
set -e                           # Exit on any error
set -u                           # Exit on undefined variable
set -o pipefail                  # Exit on pipeline failure

# Debug information in script
echo "Debug: Variable value is $var" >&2
printf "Debug: Processing file %s\n" "$filename" >&2

# Validate script arguments
if [ ! -f "$1" ]; then
    echo "Error: File $1 does not exist" >&2
    exit 1
fi
```

Common Pitfalls

- **WRONG**: `VAR = value` → **RIGHT**: `VAR=value` (no spaces around =)
- **WRONG**: Not quoting variables → **RIGHT**: Use `"$var"` to prevent word splitting
- **WRONG**: Using `[ $var = value ]` with empty var → **RIGHT**: Use `[ "$var" = "value" ]`
- **WRONG**: Forgetting executable permissions → **RIGHT**: Use `chmod +x script.sh`
- **WRONG**: Not handling script arguments → **RIGHT**: Check `$#` and validate `$1`, `$2`, etc.

Quick Verification Commands

```
# System overview
lsblk && df -h                       # Storage status
systemctl --failed && getenforce     # Services and SELinux
firewall-cmd --list-all              # Firewall rules
ss -tuln | grep LISTEN               # Listening services
mount | grep -v tmpfs                # Active mounts

# Service verification
systemctl is-active httpd && systemctl is-enabled httpd
ping -c 1 8.8.8.8 && echo "Network OK"
ausearch -m AVC -ts recent | wc -l    # Count SELinux denials
```

Last-Minute Exam Reminders

Must-Verify Checklist

- [ ] Services enabled AND started: `systemctl is-enabled service && systemctl is-active service`
- [ ] Firewall rules applied: `firewall-cmd --list-all`
- [ ] SELinux not blocking: `ausearch -m AVC -ts recent`
- [ ] Mounts persistent: Check `/etc/fstab` and `mount -a`
- [ ] Users can login: Test with `su - username`
- [ ] Network connectivity: `ping 8.8.8.8`

Emergency Commands

```
# If something breaks during exam:
systemctl status service_name        # Check service status
journalctl -u service_name --no-pager # Check logs
getenforce && ausearch -m AVC -ts recent  # SELinux issues
firewall-cmd --list-all              # Firewall blocking?
lsof -i :port                        # What's using the port?
ss -tuln | grep :port                # Is service listening?
```

Final Strategy

Accuracy over speed. Verify everything. Use man pages when uncertain.

Time allocation: 40% basic tasks (users, basic services), 40% intermediate (storage, network), 20% advanced (containers, troubleshooting).

3.2 RHCSA Command Reference by Topic Area

Organized command reference extracted from both study guides, designed for quick lookup during exam preparation.

System Information and Management

Hardware and System Information

```
# System information
uname -a                    # System kernel and architecture
uname -r                    # Kernel version
hostnamectl                 # Hostname and system info
hostnamectl set-hostname NAME  # Set system hostname
uptime                      # System uptime and load
who                         # Currently logged in users
w                           # Detailed user activity
id                          # Current user ID and groups
whoami                      # Current username

# Hardware information
lscpu                       # CPU information
lsmem                       # Memory information
lsblk                       # Block device information
lspci                       # PCI device information
lsusb                       # USB device information
lshw                        # Detailed hardware information
dmidecode                   # Hardware DMI information

# Memory and disk usage
free -h                     # Memory usage (human readable)
df -h                       # Disk space usage (human readable)
du -sh /path                # Directory size
du -h --max-depth=1 /path   # Subdirectory sizes
```

Date and Time Management

```
# Time and timezone
date                        # Current date and time
timedatectl                 # System time settings
timedatectl set-time TIME   # Set system time
timedatectl set-timezone ZONE  # Set timezone
timedatectl list-timezones  # List available timezones
hwclock                     # Hardware clock
chrony                      # Time synchronization service
systemctl status chronyd   # Check time sync status
```

File System and Storage Management

Basic File Operations

```
# Directory navigation
pwd                         # Print working directory
cd /path                    # Change directory
cd ~                        # Go to home directory
cd -                        # Go to previous directory

# File listing
ls                          # List files
ls -l                       # Long format listing
ls -la                      # Include hidden files
ls -lh                      # Human readable sizes
ls -ltr                     # Sort by time, newest last
ls -Z                       # Show SELinux contexts

# File operations
cp source dest              # Copy file
cp -r source dest           # Copy directory recursively
cp -p source dest           # Preserve permissions and timestamps
mv source dest              # Move/rename file
rm file                     # Remove file
rm -r directory             # Remove directory recursively
rm -rf directory            # Force remove directory
mkdir directory             # Create directory
mkdir -p path/to/dir        # Create parent directories
rmdir directory             # Remove empty directory
touch file                  # Create empty file or update timestamp
```

File Searching and Finding

```
# Find files and directories
find /path -name "pattern"  # Find by name
find /path -type f          # Find files only
find /path -type d          # Find directories only
find /path -user username   # Find by owner
find /path -group groupname # Find by group
find /path -perm 755        # Find by permissions
find /path -size +100M      # Find large files (>100MB)
find /path -mtime -7        # Modified in last 7 days
find /path -atime +30       # Accessed more than 30 days ago
find /path -exec command {} \;  # Execute command on results

# Alternative find commands
locate filename             # Fast file location (updatedb)
which command               # Find command location
whereis command             # Find command, source, manual
type command                # Show command type and location
```

File Linking

```
# Hard and soft links
ln source hard_link         # Create hard link
ln -s target soft_link      # Create symbolic link
readlink link               # Show link target
stat file                   # Show file statistics and links
```

File Compression and Archives

```
# tar archives
tar -czf archive.tar.gz files/     # Create gzipped tar
tar -xzf archive.tar.gz            # Extract gzipped tar
tar -cjf archive.tar.bz2 files/    # Create bzip2 tar
tar -xjf archive.tar.bz2           # Extract bzip2 tar
tar -tf archive.tar.gz             # List archive contents
tar -czf archive.tar.gz --exclude=pattern files/  # Exclude pattern

# Individual compression
gzip file                   # Compress with gzip
gunzip file.gz              # Decompress gzip
bzip2 file                  # Compress with bzip2
bunzip2 file.bz2            # Decompress bzip2
zip archive.zip files       # Create zip archive
unzip archive.zip           # Extract zip archive
```

Text Processing and File Content

Viewing File Contents

```
# Display file contents
cat file                    # Display entire file
cat -n file                 # Display with line numbers
tac file                    # Display in reverse order
head file                   # First 10 lines
head -n 20 file             # First 20 lines
tail file                   # Last 10 lines
tail -n 20 file             # Last 20 lines
tail -f file                # Follow file changes
less file                   # Page through file
more file                   # Page through file (simpler)
```

Text Processing Tools

```
# Search and filter
grep pattern file           # Search for pattern
grep -i pattern file        # Case insensitive search
grep -v pattern file        # Invert match (exclude pattern)
grep -r pattern directory   # Recursive search
grep -n pattern file        # Show line numbers
grep -E "regex" file        # Extended regex
grep -A 3 pattern file      # Show 3 lines after match
grep -B 3 pattern file      # Show 3 lines before match
grep -C 3 pattern file      # Show 3 lines before and after

# Text manipulation
sort file                   # Sort lines
sort -n file                # Numeric sort
sort -r file                # Reverse sort
sort -k 2 file              # Sort by second field
uniq file                   # Remove duplicate lines
uniq -c file                # Count occurrences
cut -d: -f1 file            # Extract first field (: delimiter)
cut -c1-10 file             # Extract characters 1-10
awk '{print $1}' file       # Print first field
awk -F: '{print $1}' file   # Custom field separator
sed 's/old/new/g' file      # Replace all occurrences
sed '1,10d' file            # Delete lines 1-10
tr 'a-z' 'A-Z' < file       # Convert lowercase to uppercase

# Line and word counts
wc file                     # Lines, words, characters
wc -l file                  # Line count only
wc -w file                  # Word count only
wc -c file                  # Character count only
```

Text Editors

```
# vim editor
vim file                    # Open file in vim
# vim modes: i (insert), ESC (normal), :wq (save and quit)
# :q! (quit without saving), :w (save), /pattern (search)

# nano editor
nano file                   # Open file in nano
# Ctrl+X to exit, Ctrl+O to save, Ctrl+K to cut line
```

Permissions and Security

File Permissions

```
# View permissions
ls -l file                  # Show permissions
stat file                   # Detailed file information
getfacl file                # Show ACL permissions

# Change permissions (symbolic)
chmod u+x file              # Add execute for user
chmod g-w file              # Remove write for group
chmod o=r file              # Set other to read only
chmod a+r file              # Add read for all
chmod +x file               # Add execute for all

# Change permissions (octal)
chmod 755 file              # rwxr-xr-x
chmod 644 file              # rw-r--r--
chmod 600 file              # rw-------
chmod 777 file              # rwxrwxrwx

# Change ownership
chown user file             # Change owner
chown user:group file       # Change owner and group
chgrp group file            # Change group only
chown -R user:group directory  # Recursive ownership change

# Default permissions
umask                       # Show current umask
umask 022                   # Set umask (755 for dirs, 644 for files)
umask 077                   # Set umask (700 for dirs, 600 for files)
```

Special Permissions

```
# Special permission bits
chmod +s file               # Set setuid/setgid
chmod +t directory          # Set sticky bit
chmod 4755 file             # setuid (4000 + 755)
chmod 2755 directory        # setgid (2000 + 755)
chmod 1755 directory        # sticky bit (1000 + 755)

# Find special permissions
find / -perm -4000 2>/dev/null  # Find setuid files
find / -perm -2000 2>/dev/null  # Find setgid files
find / -perm -1000 2>/dev/null  # Find sticky bit files
```

Access Control Lists (ACLs) — Supplementary (not on RHEL 10 exam)

```
# Manage ACLs
setfacl -m u:username:rwx file      # Set user ACL
setfacl -m g:groupname:rx file      # Set group ACL
setfacl -m d:u:username:rwx dir     # Set default ACL
setfacl -x u:username file          # Remove user ACL
setfacl -b file                     # Remove all ACLs
getfacl file                        # Display ACLs
```

User and Group Management

User Account Management

```
# Create users
useradd username            # Create user with defaults
useradd -u 1001 username    # Specify UID
useradd -g group username   # Specify primary group
useradd -G groups username  # Specify supplementary groups
useradd -s /bin/bash username  # Specify shell
useradd -d /home/user username  # Specify home directory
useradd -c "Full Name" username  # Add comment
useradd -m username         # Create home directory
useradd -r username         # Create system account

# Modify users
usermod -aG group username  # Add to supplementary group
usermod -g group username   # Change primary group
usermod -s /bin/bash username  # Change shell
usermod -d /new/home username  # Change home directory
usermod -c "New Name" username  # Change comment
usermod -L username         # Lock account
usermod -U username         # Unlock account
usermod -e 2024-12-31 username  # Set expiration date

# Delete users
userdel username            # Delete user (keep home)
userdel -r username         # Delete user and home directory

# User information
id username                 # Show user ID and groups
groups username             # Show user groups
finger username             # User information (if available)
```

Password Management

```
# Password operations
passwd username             # Set user password
passwd -l username          # Lock password
passwd -u username          # Unlock password
passwd -d username          # Delete password
passwd -e username          # Expire password (force change)

# Password aging
chage username              # Interactive password aging
chage -l username           # List password aging info
chage -M 90 username        # Max password age (90 days)
chage -m 7 username         # Min password age (7 days)
chage -W 7 username         # Warning period (7 days)
chage -d 0 username         # Force password change on next login
chage -E 2024-12-31 username  # Account expiration date
```

Group Management

```
# Create groups
groupadd groupname          # Create group
groupadd -g 1001 groupname  # Specify GID
groupadd -r groupname       # Create system group

# Modify groups
groupmod -n newname oldname # Rename group
groupmod -g 1002 groupname  # Change GID
gpasswd -a user group       # Add user to group
gpasswd -d user group       # Remove user from group
gpasswd -A admin group      # Set group administrator

# Delete groups
groupdel groupname          # Delete group

# Group information
groups                      # Show current user groups
getent group groupname      # Show group information
```

User Information and Login History

```
# Current activity
who                         # Currently logged in users
w                           # Detailed user activity
users                       # Simple list of logged in users
last                        # Login history
lastb                       # Failed login attempts
lastlog                     # Last login for all users
```

Process and Job Management

Process Monitoring

```
# View processes
ps                          # Current session processes
ps aux                      # All processes (BSD style)
ps -ef                      # All processes (Unix style)
ps -u username              # Processes by user
ps -C processname           # Processes by name
pstree                      # Process tree
top                         # Real-time process monitor
htop                        # Enhanced process monitor
```

Process Control

```
# Find processes
pgrep processname           # Find process IDs by name
pgrep -u username           # Find processes by user
pidof processname           # Find PID of running process

# Kill processes
kill PID                    # Terminate process by PID
kill -9 PID                 # Force kill process
kill -15 PID                # Graceful termination (default)
killall processname         # Kill all processes by name
pkill processname           # Kill processes by name
pkill -u username           # Kill processes by user

# Process priority
nice -n 10 command          # Start with priority 10
renice 5 PID                # Change priority of running process
renice -5 -u username       # Change priority for user processes
```

Job Control

```
# Background jobs
command &                   # Run in background
jobs                        # List active jobs
bg %1                       # Put job 1 in background
fg %1                       # Bring job 1 to foreground
disown %1                   # Remove job from shell
nohup command &             # Run immune to hangups

# Job control signals
Ctrl+Z                      # Suspend current job
Ctrl+C                      # Interrupt current job
```

System Services and Systemd

Service Management

```
# Service operations
systemctl start service     # Start service
systemctl stop service      # Stop service
systemctl restart service   # Restart service
systemctl reload service    # Reload configuration
systemctl enable service    # Enable at boot
systemctl disable service   # Disable at boot
systemctl enable --now service  # Enable and start
systemctl mask service      # Mask service (prevent start)
systemctl unmask service    # Unmask service

# Service status
systemctl status service    # Service status
systemctl is-active service # Check if running
systemctl is-enabled service # Check if enabled
systemctl is-failed service # Check if failed
systemctl list-units --type=service  # List all services
systemctl list-units --state=failed  # List failed services
systemctl --failed          # Show failed services
```

Systemd Targets

```
# Target management
systemctl get-default       # Show default target
systemctl set-default multi-user.target  # Set default target
systemctl isolate rescue.target  # Switch to target
systemctl list-units --type=target  # List targets
systemctl list-dependencies target  # Show target dependencies
```

Unit Files and Configuration

```
# Unit file management
systemctl daemon-reload     # Reload unit files
systemctl cat service       # Show unit file content
systemctl edit service      # Edit unit file (override)
systemctl revert service    # Revert unit file changes
systemctl show service      # Show unit properties
```

Logging and Monitoring

Journal (systemd logs)

```
# View logs
journalctl                  # All journal entries
journalctl -u service       # Service-specific logs
journalctl -f               # Follow (tail) logs
journalctl -n 50            # Last 50 entries
journalctl -p err           # Error priority and above
journalctl --since "1 hour ago"  # Recent entries
journalctl --since "2024-01-01"  # Since date
journalctl --until "2024-01-31"  # Until date
journalctl -b               # Current boot logs
journalctl --list-boots     # List boot sessions
journalctl -k               # Kernel messages
journalctl --disk-usage     # Journal disk usage
```

Traditional Logs

```
# Log files
tail -f /var/log/messages   # Follow system messages
tail -f /var/log/secure     # Follow security logs
tail -f /var/log/maillog    # Follow mail logs
less /var/log/cron          # Cron job logs
logger "test message"       # Send message to syslog
```

Log Rotation

```
# Logrotate
logrotate -d /etc/logrotate.conf  # Debug/test rotation
logrotate -f /etc/logrotate.conf  # Force rotation
```

Network Configuration and Management

Network Information

```
# Network interfaces
ip addr show                # Show IP addresses
ip link show                # Show network interfaces
ip route show               # Show routing table
ip route get 8.8.8.8        # Show route to destination

# Legacy commands (still available)
ifconfig                    # Show interfaces (deprecated)
route -n                    # Show routing table (deprecated)
```

NetworkManager with nmcli

```
# Connection management
nmcli device status         # Device status
nmcli connection show       # Show connections
nmcli con show "connection" # Show connection details
nmcli device wifi list     # List WiFi networks

# Create connections
nmcli con add type ethernet con-name "conn1" ifname eth0
nmcli con add type wifi con-name "wifi1" ifname wlan0 ssid "SSID"

# Modify connections
nmcli con modify "conn1" ipv4.addresses "192.168.1.100/24"
nmcli con modify "conn1" ipv4.gateway "192.168.1.1"
nmcli con modify "conn1" ipv4.dns "8.8.8.8,8.8.4.4"
nmcli con modify "conn1" ipv4.method manual
nmcli con modify "conn1" autoconnect yes

# Control connections
nmcli con up "conn1"        # Activate connection
nmcli con down "conn1"      # Deactivate connection
nmcli con reload            # Reload configurations
nmcli con delete "conn1"    # Delete connection
```

Network Testing and Troubleshooting

```
# Connectivity testing
ping host                   # Test connectivity
ping -c 4 host              # Send 4 packets
traceroute host             # Trace network path
mtr host                    # Real-time traceroute

# DNS resolution
nslookup hostname           # DNS lookup
dig hostname                # Detailed DNS lookup
dig @server hostname        # Query specific DNS server
host hostname               # Simple DNS lookup

# Network connections
ss -tuln                    # Show listening ports
ss -tupln                   # Show all connections with PIDs
netstat -tuln               # Legacy network connections
lsof -i :80                 # Show what's using port 80
lsof -i tcp:22              # Show SSH connections
```

Network File System (NFS) and AutoFS

NFS Client Operations

```
# NFS package installation
dnf install -y nfs-utils       # Install NFS client utilities

# Discovering NFS shares
showmount -e server.example.com  # List exports from NFS server
showmount -e 192.168.1.100      # List exports using IP address
showmount -a server             # Show all client connections
showmount -d server             # Show directories being accessed

# Manual NFS mounting
mkdir /mnt/nfs-share            # Create mount point
mount -t nfs server:/export/share /mnt/nfs-share  # Mount NFS share
mount -t nfs -o nfsvers=4.2 server:/share /mnt    # Specify NFS version
mount -o rw,intr server:/data /mnt/data            # Mount with options

# NFS mount options
# rw/ro                - read-write/read-only
# hard/soft           - retry behavior on failure
# intr                - allow interrupts
# rsize=8192          - read buffer size
# wsize=8192          - write buffer size
# timeo=14            - timeout (1/10 second)
# retrans=3           - retry attempts
# _netdev             - wait for network

# NFS unmounting
umount /mnt/nfs-share          # Unmount NFS share
umount -l /mnt/nfs-share       # Lazy unmount (when busy)
umount -f /mnt/nfs-share       # Force unmount

# Testing NFS connectivity
ping nfs-server                # Basic connectivity
telnet nfs-server 2049         # Test NFS port
rpcinfo -p nfs-server          # Show RPC services
```

NFS Server Management

```
# NFS server package installation
dnf install -y nfs-utils       # Install NFS server utilities

# Export configuration (/etc/exports)
/export/share *(rw,sync)       # Export to all hosts
/data 192.168.1.0/24(rw,sync) # Export to specific network
/home server1(rw) server2(ro)  # Different permissions per host

# Export management commands
exportfs -avr                  # Export all shares with verbose output
exportfs -v                    # Show current exports
exportfs -u /export/share      # Unexport specific share
exportfs -ra                   # Re-export all shares

# NFS service management
systemctl enable --now nfs-server  # Start and enable NFS server
systemctl enable --now rpcbind     # Start and enable RPC service
systemctl status nfs-server        # Check NFS server status

# Firewall configuration for NFS
firewall-cmd --add-service=nfs --permanent      # Allow NFS traffic
firewall-cmd --add-service=rpc-bind --permanent # Allow RPC bind
firewall-cmd --add-service=mountd --permanent   # Allow mountd
firewall-cmd --reload                          # Apply firewall changes
```

AutoFS Configuration and Management

```
# AutoFS installation
dnf install -y autofs          # Install AutoFS package

# Master map configuration (/etc/auto.master)
/mnt/auto /etc/auto.nfs --timeout=60    # Indirect map with timeout
/- /etc/auto.direct                     # Direct map entry

# Indirect map configuration
# Format: key [options] server:/path
shared -rw server.example.com:/export/shared
data -ro,intr server:/export/data

# Direct map configuration
# Format: mount-point [options] server:/path
/mnt/shared-data -rw server:/export/data
/opt/software -ro server:/export/software

# Wildcard mapping for user directories
# Format: * [options] server:/path/&
* -rw server.example.com:/home/&        # Maps to server:/home/username

# AutoFS service management
systemctl enable --now autofs          # Start and enable AutoFS
systemctl status autofs                # Check AutoFS status
systemctl reload autofs                # Reload configuration
systemctl restart autofs               # Restart AutoFS service

# AutoFS monitoring and troubleshooting
automount -f -v                        # Run in foreground with verbose
tail -f /var/log/messages | grep automount  # Watch AutoFS logs
ls -la /etc/auto.*                     # Check map file permissions
mount | grep autofs                    # Show active automounts
```

fstab Integration for NFS

```
# fstab entry format for NFS
# device mount-point type options dump fsck
server:/export/share /mnt/nfs nfs defaults,_netdev 0 0

# Common fstab NFS options
defaults,_netdev                # Standard options with network dependency
rw,_netdev,soft,intr           # Read-write, soft mount, interruptible
ro,_netdev,hard,retrans=3      # Read-only, hard mount, 3 retries

# Testing fstab entries
mount -a                       # Mount all fstab entries
umount /mnt/nfs && mount /mnt/nfs  # Test specific entry
findmnt /mnt/nfs              # Show mount details
```

NFS and AutoFS Troubleshooting

```
# NFS client troubleshooting
showmount -e server           # Test server connectivity
rpcinfo -p server            # Check RPC services on server
mount -v -t nfs server:/share /mnt  # Verbose mount output
ping server && telnet server 2049  # Test basic and NFS connectivity

# NFS server troubleshooting
exportfs -v                   # Show current exports
rpcinfo -p localhost          # Check local RPC services
systemctl status nfs-server rpcbind  # Check service status
showmount -e localhost        # Test local NFS exports
netstat -tuln | grep :2049   # Check NFS port listening

# AutoFS troubleshooting
systemctl status autofs       # Check AutoFS service
automount -f -v              # Run AutoFS in foreground
ls -la /etc/auto.*           # Check map file permissions
cat /etc/auto.master         # Verify master map syntax
tail -f /var/log/messages | grep automount  # Watch logs

# General network filesystem troubleshooting
mount | grep nfs             # Show NFS mounts
df -t nfs                    # Show NFS filesystem usage
fuser -mv /mnt/nfs-share     # Show processes using mount point
lsof +D /mnt/nfs-share       # Show open files in NFS mount
```

Package Management

DNF Package Manager

```
# Package operations
dnf install package         # Install package
dnf update package          # Update package
dnf remove package          # Remove package
dnf upgrade                 # Update all packages
dnf downgrade package       # Downgrade package
dnf reinstall package       # Reinstall package

# Package information
dnf search keyword          # Search packages
dnf info package            # Package information
dnf list installed          # List installed packages
dnf list available          # List available packages
dnf list "pattern*"         # List packages matching pattern
dnf provides /path/file     # Find package providing file
dnf repoquery --requires package  # Show dependencies

# Package groups
dnf group list              # List package groups
dnf group info "group"      # Group information
dnf group install "group"   # Install package group
dnf group remove "group"    # Remove package group

# Repository management
dnf repolist                # List repositories
dnf repolist all            # List all repositories
dnf config-manager --add-repo URL  # Add repository
dnf config-manager --enable repo   # Enable repository
dnf config-manager --disable repo  # Disable repository

# History and cleanup
dnf history                 # Transaction history
dnf history info ID         # History transaction details
dnf history undo ID         # Undo transaction
dnf clean all               # Clean package cache
dnf autoremove              # Remove unneeded packages
```

RPM Package Manager

```
# RPM queries
rpm -qa                     # List all installed packages
rpm -qi package             # Package information
rpm -ql package             # List package files
rpm -qf /path/file          # Find package owning file
rpm -qd package             # List package documentation
rpm -qc package             # List package configuration files
rpm -q --requires package   # Show package dependencies
rpm -q --provides package   # Show what package provides

# RPM installation/removal
rpm -ivh package.rpm        # Install package
rpm -Uvh package.rpm        # Upgrade package
rpm -e package              # Remove package
rpm --import GPG-KEY        # Import GPG key

# RPM verification
rpm -V package              # Verify package integrity
rpm -Va                     # Verify all packages
```

Storage Management and File Systems

Disk and Partition Management

```
# Disk information
lsblk                       # List block devices
lsblk -f                    # Show file systems
blkid                       # Show UUIDs and file systems
fdisk -l                    # List disk partitions
df -h                       # Show mounted file systems
du -sh directory            # Directory size

# Partition management
fdisk /dev/device           # Create/modify partitions
parted /dev/device          # Alternative partitioning tool
partprobe                   # Re-read partition table
mkfs.ext4 /dev/partition    # Create ext4 file system
mkfs.xfs /dev/partition     # Create XFS file system
tune2fs -L label /dev/partition  # Set file system label
```

LVM (Logical Volume Management)

```
# Physical volumes
pvcreate /dev/device        # Create physical volume
pvdisplay                   # Show PV details
pvs                         # Show PV summary
pvremove /dev/device        # Remove physical volume

# Volume groups
vgcreate vg_name /dev/device  # Create volume group
vgdisplay                   # Show VG details
vgs                         # Show VG summary
vgextend vg_name /dev/device  # Extend volume group
vgreduce vg_name /dev/device  # Reduce volume group
vgremove vg_name            # Remove volume group

# Logical volumes
lvcreate -L 1G -n lv_name vg_name      # Create LV (size)
lvcreate -l 100%FREE -n lv_name vg_name  # Use all free space
lvcreate -l 50%VG -n lv_name vg_name   # Use 50% of VG
lvdisplay                   # Show LV details
lvs                         # Show LV summary
lvextend -L +1G /dev/vg/lv  # Extend logical volume
lvextend -l +100%FREE /dev/vg/lv  # Extend to use all space
lvreduce -L -1G /dev/vg/lv  # Reduce logical volume
lvremove /dev/vg/lv         # Remove logical volume
```

File System Operations

```
# Mounting
mount /dev/device /mountpoint     # Mount file system
mount -t xfs /dev/device /mnt     # Mount with type
mount -o ro /dev/device /mnt      # Mount read-only
umount /mountpoint                # Unmount file system
umount -l /mountpoint             # Lazy unmount
mount -a                          # Mount all in /etc/fstab

# File system resize
xfs_growfs /mountpoint      # Grow XFS file system
resize2fs /dev/device       # Resize ext4 file system
e2fsck -f /dev/device       # Check ext4 file system
xfs_repair /dev/device      # Repair XFS file system

# Swap management
mkswap /dev/device          # Create swap space
swapon /dev/device          # Enable swap
swapoff /dev/device         # Disable swap
swapon --show               # Show active swap
swapon -a                   # Enable all swap in fstab
```

fstab Configuration

```
# /etc/fstab format:
# device mountpoint fstype options dump pass
/dev/sda1 / ext4 defaults 0 1
UUID=xxx /home xfs defaults 0 2
/dev/vg/lv /data ext4 defaults 0 2
/dev/sdb1 swap swap defaults 0 0

# fstab options
defaults        # rw,suid,dev,exec,auto,nouser,async
ro              # Read-only
rw              # Read-write
noauto          # Don't mount automatically
user            # Allow users to mount
noexec          # Don't allow execution
nosuid          # Ignore setuid bits
```

Firewall Management

firewalld Configuration

```
# Firewall status
firewall-cmd --state        # Check if running
firewall-cmd --get-active-zones  # Show active zones
firewall-cmd --list-all     # Show all rules
firewall-cmd --list-services  # Show allowed services
firewall-cmd --list-ports   # Show allowed ports

# Zone management
firewall-cmd --get-default-zone  # Show default zone
firewall-cmd --set-default-zone=public  # Set default zone
firewall-cmd --get-zones    # List all zones
firewall-cmd --zone=work --list-all  # Show zone rules

# Service management
firewall-cmd --add-service=http  # Allow HTTP (temporary)
firewall-cmd --add-service=ssh --permanent  # Allow SSH (permanent)
firewall-cmd --remove-service=http  # Remove HTTP
firewall-cmd --list-services  # Show allowed services

# Port management
firewall-cmd --add-port=8080/tcp  # Allow port (temporary)
firewall-cmd --add-port=443/tcp --permanent  # Allow port (permanent)
firewall-cmd --remove-port=8080/tcp  # Remove port
firewall-cmd --list-ports   # Show allowed ports

# Rich rules
firewall-cmd --add-rich-rule='rule family="ipv4" source address="192.168.1.0/24" accept'
firewall-cmd --add-rich-rule='rule family="ipv4" source address="10.0.0.5" port port="22"
protocol="tcp" accept'

# Apply changes
firewall-cmd --reload       # Reload configuration
firewall-cmd --runtime-to-permanent  # Make runtime rules permanent
```

SELinux Management

SELinux Status and Modes

```
# SELinux status
getenforce                  # Current mode (Enforcing/Permissive/Disabled)
sestatus                    # Detailed status
setenforce 0                # Set permissive mode (temporary)
setenforce 1                # Set enforcing mode (temporary)

# Permanent mode change (edit /etc/selinux/config)
SELINUX=enforcing          # or permissive, disabled
```

File Contexts

```
# View contexts
ls -Z file                  # Show file context
ps -eZ                      # Show process contexts
id -Z                       # Show user context

# Manage contexts
restorecon file             # Restore default context
restorecon -R directory     # Restore recursively
restorecon -v file          # Verbose output

# Set custom contexts
semanage fcontext -a -t httpd_exec_t "/web(/.*)?"
restorecon -R /web          # Apply new context
semanage fcontext -l        # List file contexts
semanage fcontext -d "/web(/.*)?"  # Delete context rule
```

SELinux Booleans

```
# View booleans
getsebool -a                # List all booleans
getsebool httpd_can_network_connect  # Check specific boolean
setsebool httpd_can_network_connect on  # Set boolean (temporary)
setsebool -P httpd_can_network_connect on  # Set boolean (permanent)
```

Port Contexts

```
# Manage port contexts
semanage port -l            # List port contexts
semanage port -a -t http_port_t -p tcp 8080  # Add port context
semanage port -d -p tcp 8080  # Delete port context
semanage port -l | grep http  # Show HTTP ports
```

SELinux Troubleshooting

```
# Check for denials
ausearch -m AVC -ts recent  # Recent AVC denials
ausearch -m AVC -ts today   # Today's denials
sealert -a /var/log/audit/audit.log  # Analyze denials
sealert -l UUID             # Detailed denial analysis

# Generate policies
audit2allow -a              # Generate policy from all denials
audit2allow -a -M mypolicy  # Generate policy module
semodule -i mypolicy.pp     # Install policy module
```

Boot Process and GRUB

GRUB Configuration

```
# GRUB management
grub2-editenv list          # List GRUB environment
grub2-mkconfig -o /boot/grub2/grub.cfg  # Generate GRUB config
grub2-set-default "menu entry"  # Set default boot entry
grub2-reboot "menu entry"   # Boot specific entry once

# Kernel parameters (persistent)
grub2-editenv - set "kernelopts=root=/dev/sda1 quiet"
grub2-mkconfig -o /boot/grub2/grub.cfg
```

Boot Targets and Runlevels

```
# Systemd targets
systemctl get-default       # Show default target
systemctl set-default multi-user.target  # Set default target
systemctl isolate rescue.target  # Switch to rescue mode
systemctl isolate emergency.target  # Switch to emergency mode

# Target management
systemctl list-units --type=target  # List all targets
systemctl list-dependencies graphical.target  # Show dependencies
```

Scheduled Tasks

Cron Jobs

```
# User crontab
crontab -e                  # Edit user crontab
crontab -l                  # List user crontab
crontab -r                  # Remove user crontab
crontab -u username -e      # Edit another user's crontab

# System crontab
vim /etc/crontab            # System-wide crontab
ls /etc/cron.d/             # Additional cron files
ls /etc/cron.{hourly,daily,weekly,monthly}/  # Cron directories

# Cron format: minute hour day month weekday command
# Examples:
0 2 * * * /path/script      # Daily at 2 AM
*/15 * * * * /path/script   # Every 15 minutes
0 0 * * 0 /path/script      # Weekly on Sunday
0 3 1 * * /path/script      # Monthly on 1st at 3 AM
```

At Jobs

```
# Schedule one-time jobs
at now + 5 minutes          # Schedule for 5 minutes from now
at 15:30                    # Schedule for 3:30 PM today
at 15:30 tomorrow           # Schedule for 3:30 PM tomorrow
at -f script.sh now + 1 hour  # Run script in 1 hour

# Manage at jobs
atq                         # List scheduled jobs
atrm job_number             # Remove scheduled job
at -c job_number            # Show job details
```

Systemd Timers

```
# Timer management
systemctl list-timers       # List all timers
systemctl list-timers --all # List all timers (including inactive)
systemctl enable timer.timer  # Enable timer
systemctl start timer.timer   # Start timer
systemctl status timer.timer  # Check timer status
```

Flatpak Software Management

Remote and Application Management

```
# Remote (repository) management
flatpak remote-add --if-not-exists flathub https://flathub.org/repo/flathub.flatpakrepo
flatpak remote-add --user flathub URL  # User-level remote
flatpak remote-delete flathub          # Remove remote
flatpak remotes                        # List configured remotes
flatpak remote-ls flathub              # List available apps

# Application management
flatpak search keyword                 # Search for apps
flatpak install flathub org.example.App  # Install app
flatpak install --user flathub org.example.App  # User-level install
flatpak uninstall org.example.App      # Remove app
flatpak uninstall --unused             # Remove unused runtimes
flatpak list --app                     # List installed apps
flatpak list --runtime                 # List installed runtimes
flatpak run org.example.App            # Run app
flatpak update                         # Update all
flatpak info org.example.App           # Show app details
```

SSH and Remote Access

SSH Client

```
# SSH connections
ssh user@hostname           # Connect to remote host
ssh -p 2222 user@hostname   # Connect to custom port
ssh -i keyfile user@hostname  # Use specific key
ssh -L 8080:localhost:80 user@host  # Local port forwarding
ssh -R 8080:localhost:80 user@host  # Remote port forwarding
ssh -X user@hostname        # X11/Wayland forwarding

# Key management
ssh-keygen -t rsa           # Generate RSA key pair
ssh-keygen -t ed25519       # Generate Ed25519 key pair
ssh-copy-id user@hostname   # Copy public key to remote
ssh-add keyfile             # Add key to SSH agent
ssh-agent bash              # Start SSH agent
```

SSH Server Configuration

```
# SSH daemon configuration (/etc/ssh/sshd_config)
Port 22                     # Change SSH port
PermitRootLogin no          # Disable root login
PasswordAuthentication no   # Disable password auth
PubkeyAuthentication yes    # Enable key-based auth
AllowUsers user1 user2      # Restrict users

# Apply SSH configuration
systemctl reload sshd       # Reload SSH daemon
sshd -t                     # Test configuration
```

File Transfer

```
# SCP (Secure Copy)
scp file user@host:/path    # Copy file to remote
scp user@host:/path/file .  # Copy file from remote
scp -r directory user@host:/path  # Copy directory

# RSYNC
rsync -av source/ destination/  # Sync directories
rsync -av --delete source/ dest/  # Sync and delete extras
rsync -av user@host:/path/ local/  # Sync from remote
```

This comprehensive command reference covers all major RHCSA topics with practical command examples organized by functional area for efficient study and quick reference during exam preparation.

3.3 RHCSA Acronyms and Glossary

Comprehensive list of acronyms, abbreviations, and key terms for RHCSA exam preparation

Quick Navigation

- Alphabetical Index
- Acronyms by Category
- Certification & System
- Hardware & Boot
- File Systems & Storage
- Networking
- Security
- Services & Process Management
- Flatpak & Software Distribution
- Package Management

Alphabetical Index

A

- **ACL** - Access Control List - Extended file permissions beyond standard Unix permissions
- **API** - Application Programming Interface - Set of protocols for building software applications
- **ARP** - Address Resolution Protocol - Maps IP addresses to MAC addresses
- **AVC** - Access Vector Cache - SELinux component that caches access decisions
- **AWK** - Aho, Weinberger, and Kernighan - Pattern scanning and processing language

B

- **BASH** - Bourne Again Shell - Default command interpreter in RHEL
- **BIOS** - Basic Input/Output System - Legacy firmware interface for PC boot
- **BIND** - Berkeley Internet Name Domain - DNS server software
- **BTRFS** - B-tree File System - Copy-on-write filesystem (not default in RHEL)

C

- **CD** - Change Directory - Command to navigate filesystem
- **CIDR** - Classless Inter-Domain Routing - IP addressing and routing methodology

- **CLI** - Command Line Interface - Text-based interface for system interaction
- **CPU** - Central Processing Unit - Main processor of a computer
- **CRON** - Command Run On - Time-based job scheduler
- **CSV** - Comma-Separated Values - File format for tabular data

D

- **DAC** - Discretionary Access Control - Standard Linux permissions model
- **DHCP** - Dynamic Host Configuration Protocol - Automatic IP address assignment
- **DMZ** - Demilitarized Zone - Network security zone
- **DNF** - Dandified YUM - Modern package manager for RHEL 8/9
- **DNS** - Domain Name System - Translates domain names to IP addresses
- **DOS** - Disk Operating System - Legacy operating system

E

- **EFI** - Extensible Firmware Interface - See UEFI
- **EOF** - End of File - Marker indicating no more data
- **EPM** - Enterprise Package Manager - Package management tool
- **EXT4** - Fourth Extended Filesystem - Linux filesystem type

F

- **FDISK** - Fixed Disk - Disk partitioning utility
- **FIFO** - First In, First Out - Named pipe for inter-process communication
- **FQDN** - Fully Qualified Domain Name - Complete domain name including hostname
- **FSCK** - File System Check - Filesystem integrity checker
- **FTP** - File Transfer Protocol - Network protocol for file transfer

G

- **GCC** - GNU Compiler Collection - Compiler system
- **GID** - Group Identifier - Numeric group identification
- **GNU** - GNU's Not Unix - Free software project
- **GPG** - GNU Privacy Guard - Encryption and signing tool
- **GPL** - General Public License - Free software license
- **GPT** - GUID Partition Table - Modern disk partitioning scheme
- **GRUB** - Grand Unified Bootloader - Linux boot loader
- **GUI** - Graphical User Interface - Visual interface for system interaction
- **GUID** - Globally Unique Identifier - Unique reference number

H

- **HDD** - Hard Disk Drive - Mechanical storage device
- **HOME** - Home Directory - User's personal directory
- **HTML** - HyperText Markup Language - Web page formatting language
- **HTTP** - HyperText Transfer Protocol - Web communication protocol
- **HTTPS** - HTTP Secure - Encrypted web communication protocol

I

- **ICMP** - Internet Control Message Protocol - Network diagnostic protocol (ping)
- **IDE** - Integrated Drive Electronics - Legacy disk interface
- **I/O** - Input/Output - Data transfer operations
- **IP** - Internet Protocol - Network layer protocol
- **IPC** - Inter-Process Communication - Process interaction mechanisms
- **IPv4** - Internet Protocol version 4 - 32-bit IP addressing
- **IPv6** - Internet Protocol version 6 - 128-bit IP addressing
- **ISO** - International Organization for Standardization - Standards body

J

- **JSON** - JavaScript Object Notation - Data interchange format
- **JVM** - Java Virtual Machine - Java runtime environment

K

- **KB** - Kilobyte - 1024 bytes
- **KVM** - Kernel-based Virtual Machine - Linux virtualization technology

L

- **LAMP** - Linux, Apache, MySQL, PHP - Web application stack
- **LDAP** - Lightweight Directory Access Protocol - Directory service protocol
- **LIFO** - Last In, First Out - Stack data structure
- **LTS** - Long Term Support - Extended support release
- **LUKS** - Linux Unified Key Setup - Disk encryption specification
- **LV** - Logical Volume - LVM storage unit
- **LVM** - Logical Volume Manager - Flexible disk management system

M

- **MAC** - Mandatory Access Control - SELinux security model
- **MAC** - Media Access Control - Hardware network address
- **MB** - Megabyte - 1024 kilobytes
- **MBR** - Master Boot Record - Legacy boot sector (2TB limit)
- **MCS** - Multi-Category Security - SELinux security model
- **MD5** - Message Digest 5 - Cryptographic hash function
- **MLS** - Multi-Level Security - SELinux security model
- **MTU** - Maximum Transmission Unit - Largest packet size

N

- **NAT** - Network Address Translation - IP address remapping
- **NFS** - Network File System - Distributed filesystem protocol (NFS 4.2 default in RHEL 10)
- **NIC** - Network Interface Card - Network adapter hardware
- **NIS** - Network Information Service - Directory service
- **NMCLI** - NetworkManager Command Line Interface
- **NMTUI** - NetworkManager Text User Interface
- **NTP** - Network Time Protocol - Time synchronization protocol

O

- **OSTree** - Content-addressable filesystem used by Flatpak for efficient storage
- **OS** - Operating System - System software managing hardware/software
- **OSS** - Open Source Software - Publicly accessible source code

P

- **PAM** - Pluggable Authentication Modules - Authentication framework
- **PATH** - Path Variable - Executable search directories
- **PCI** - Peripheral Component Interconnect - Expansion card standard
- **PE** - Physical Extent - LVM allocation unit
- **PID** - Process Identifier - Unique process number
- **PING** - Packet Internet Groper - Network connectivity test
- **PPID** - Parent Process Identifier - Parent process number
- **PS** - Process Status - Process listing command
- **PV** - Physical Volume - LVM physical storage

Q

- **QEMU** - Quick Emulator - Machine emulator and virtualizer

R

- **RAID** - Redundant Array of Independent Disks - Data redundancy/performance
- **RAM** - Random Access Memory - System memory
- **RBAC** - Role-Based Access Control - Security model
- **REGEX** - Regular Expression - Pattern matching syntax
- **RHCSA** - Red Hat Certified System Administrator - Entry-level certification
- **RHCE** - Red Hat Certified Engineer - Advanced certification
- **RHEL** - Red Hat Enterprise Linux - Enterprise Linux distribution
- **RPM** - Red Hat Package Manager - Package format and tool
- **RSA** - Rivest-Shamir-Adleman - Public-key cryptosystem
- **RSS** - Resident Set Size - Memory usage metric
- **RSYNC** - Remote Sync - File synchronization tool

S

- **SAMBA** - Server Message Block - Windows file sharing on Linux
- **SATA** - Serial Advanced Technology Attachment - Disk interface
- **SCSI** - Small Computer System Interface - Storage interface standard
- **SED** - Stream Editor - Text processing tool
- **SELinux** - Security-Enhanced Linux - Mandatory access control system
- **SFTP** - SSH File Transfer Protocol - Secure file transfer
- **SHA** - Secure Hash Algorithm - Cryptographic hash functions
- **SHELL** - Command interpreter - User interface for OS access
- **SKEL** - Skeleton - Template directory for new users
- **SMTP** - Simple Mail Transfer Protocol - Email transmission protocol
- **SNMP** - Simple Network Management Protocol - Network management
- **SSD** - Solid State Drive - Flash-based storage device
- **SSH** - Secure Shell - Encrypted remote access protocol
- **SSL** - Secure Sockets Layer - Deprecated encryption protocol
- **STDIN** - Standard Input - Input stream (file descriptor 0)
- **STDOUT** - Standard Output - Output stream (file descriptor 1)
- **STDERR** - Standard Error - Error stream (file descriptor 2)
- **SUDO** - Superuser Do - Privilege escalation command
- **SWAP** - Swap Space - Disk-based virtual memory

- **SYSTEMD** - System Daemon - Init system and service manager

T

- **TAR** - Tape Archive - Archive file format and tool
- **TCP** - Transmission Control Protocol - Reliable network protocol
- **TLS** - Transport Layer Security - Modern encryption protocol
- **TMP** - Temporary - Temporary files directory
- **TTY** - Teletypewriter - Terminal device
- **TUI** - Text User Interface - Text-based graphical interface
- **TZ** - Time Zone - Timezone environment variable

U

- **UDP** - User Datagram Protocol - Connectionless network protocol
- **UEFI** - Unified Extensible Firmware Interface - Modern firmware standard
- **UID** - User Identifier - Numeric user identification
- **UMASK** - User Mask - Default permission mask
- **URI** - Uniform Resource Identifier - Resource identifier
- **URL** - Uniform Resource Locator - Web address
- **USB** - Universal Serial Bus - Peripheral connection standard
- **UTF** - Unicode Transformation Format - Character encoding
- **UUID** - Universally Unique Identifier - Unique device/filesystem ID

V

- **VG** - Volume Group - LVM storage pool
- **VI/VIM** - Visual/Vi Improved - Text editor
- **VLAN** - Virtual Local Area Network - Network segmentation
- **VM** - Virtual Machine - Virtualized computer system
- **VNC** - Virtual Network Computing - Remote desktop protocol
- **VPN** - Virtual Private Network - Secure network connection

W

- **WAN** - Wide Area Network - Large geographic network
- **WWW** - World Wide Web - Global information system

X

- **Wayland** - Modern display protocol replacing X11 in RHEL 10
- **XFS** - X File System - High-performance filesystem (RHEL default)
- **XML** - Extensible Markup Language - Data format

Y

- **YAML** - YAML Ain't Markup Language - Data serialization format
- **YUM** - Yellowdog Updater Modified - Legacy package manager

Z

- **ZLIB** - Compression Library - Data compression library
- **ZFS** - Zettabyte File System - Advanced filesystem (not in RHEL)

Acronyms by Category

Certification & System

- **RHCSA** - Red Hat Certified System Administrator
- **RHCE** - Red Hat Certified Engineer
- **RHEL** - Red Hat Enterprise Linux
- **OS** - Operating System
- **CLI** - Command Line Interface
- **GUI** - Graphical User Interface
- **API** - Application Programming Interface
- **FQDN** - Fully Qualified Domain Name
- **TTY** - Teletypewriter (terminal)
- **EOF** - End of File
- **STDIN/STDOUT/STDERR** - Standard input/output/error
- **BASH** - Bourne Again Shell
- **GNU** - GNU's Not Unix
- **GPL** - General Public License
- **OSS** - Open Source Software
- **LTS** - Long Term Support

Hardware & Boot

- **CPU** - Central Processing Unit
- **RAM** - Random Access Memory
- **BIOS** - Basic Input/Output System
- **UEFI** - Unified Extensible Firmware Interface
- **EFI** - Extensible Firmware Interface
- **GRUB** - Grand Unified Bootloader
- **MBR** - Master Boot Record (2TB limit)
- **GPT** - GUID Partition Table
- **GUID** - Globally Unique Identifier
- **UUID** - Universally Unique Identifier
- **HDD** - Hard Disk Drive
- **SSD** - Solid State Drive
- **SATA** - Serial Advanced Technology Attachment
- **SCSI** - Small Computer System Interface
- **IDE** - Integrated Drive Electronics
- **PCI** - Peripheral Component Interconnect
- **USB** - Universal Serial Bus
- **I/O** - Input/Output

File Systems & Storage

- **XFS** - X File System (RHEL default)
- **EXT4** - Fourth Extended Filesystem
- **BTRFS** - B-tree File System
- **ZFS** - Zettabyte File System (not in RHEL)
- **LVM** - Logical Volume Manager
- **PV** - Physical Volume
- **VG** - Volume Group
- **LV** - Logical Volume
- **PE** - Physical Extent
- **LUKS** - Linux Unified Key Setup
- **RAID** - Redundant Array of Independent Disks
- **NFS** - Network File System
- **FSCK** - File System Check
- **FDISK** - Fixed Disk
- **SWAP** - Swap Space

Networking

- **IP** - Internet Protocol
- **IPv4** - Internet Protocol version 4
- **IPv6** - Internet Protocol version 6
- **TCP** - Transmission Control Protocol
- **UDP** - User Datagram Protocol
- **ICMP** - Internet Control Message Protocol
- **DHCP** - Dynamic Host Configuration Protocol
- **DNS** - Domain Name System
- **BIND** - Berkeley Internet Name Domain
- **HTTP** - HyperText Transfer Protocol
- **HTTPS** - HTTP Secure
- **FTP** - File Transfer Protocol
- **SFTP** - SSH File Transfer Protocol
- **SSH** - Secure Shell
- **SSL** - Secure Sockets Layer (deprecated)
- **TLS** - Transport Layer Security
- **SMTP** - Simple Mail Transfer Protocol
- **SNMP** - Simple Network Management Protocol
- **CIDR** - Classless Inter-Domain Routing
- **MAC** - Media Access Control
- **ARP** - Address Resolution Protocol
- **NAT** - Network Address Translation
- **NIC** - Network Interface Card
- **MTU** - Maximum Transmission Unit
- **VLAN** - Virtual Local Area Network
- **VPN** - Virtual Private Network
- **WAN** - Wide Area Network
- **DMZ** - Demilitarized Zone
- **NMCLI** - NetworkManager Command Line Interface
- **NMTUI** - NetworkManager Text User Interface

Security

- **SELinux** - Security-Enhanced Linux
- **MAC** - Mandatory Access Control
- **DAC** - Discretionary Access Control
- **RBAC** - Role-Based Access Control

- **AVC** - Access Vector Cache
- **MLS** - Multi-Level Security
- **MCS** - Multi-Category Security
- **PAM** - Pluggable Authentication Modules
- **ACL** - Access Control List
- **RSA** - Rivest-Shamir-Adleman
- **SHA** - Secure Hash Algorithm
- **MD5** - Message Digest 5
- **GPG** - GNU Privacy Guard

Services & Process Management

- **SYSTEMD** - System Daemon
- **PID** - Process Identifier
- **PPID** - Parent Process Identifier
- **PS** - Process Status
- **RSS** - Resident Set Size
- **CRON** - Command Run On
- **NTP** - Network Time Protocol
- **LDAP** - Lightweight Directory Access Protocol
- **NIS** - Network Information Service
- **IPC** - Inter-Process Communication
- **FIFO** - First In, First Out
- **LIFO** - Last In, First Out

Flatpak & Software Distribution

- **Flatpak** - Application distribution framework with sandboxing
- **Flathub** - Public Flatpak application repository
- **OSTree** - Content-addressable storage system
- **VM** - Virtual Machine
- **KVM** - Kernel-based Virtual Machine
- **QEMU** - Quick Emulator
- **VNC** - Virtual Network Computing

Package Management

- **DNF** - Dandified YUM
- **YUM** - Yellowdog Updater Modified

- **RPM** - Red Hat Package Manager
- **EPM** - Enterprise Package Manager

Other Common Acronyms

- **AWK** - Aho, Weinberger, and Kernighan
- **SED** - Stream Editor
- **TAR** - Tape Archive
- **REGEX** - Regular Expression
- **CSV** - Comma-Separated Values
- **HTML** - HyperText Markup Language
- **XML** - Extensible Markup Language
- **YAML** - YAML Ain't Markup Language
- **JSON** - JavaScript Object Notation
- **JVM** - Java Virtual Machine
- **GCC** - GNU Compiler Collection
- **LAMP** - Linux, Apache, MySQL, PHP
- **RSYNC** - Remote Sync
- **SAMBA** - Server Message Block
- **UTF** - Unicode Transformation Format
- **URI** - Uniform Resource Identifier
- **URL** - Uniform Resource Locator
- **WWW** - World Wide Web
- **TUI** - Text User Interface
- **TZ** - Time Zone
- **ZLIB** - Compression Library
- **KB** - Kilobyte
- **MB** - Megabyte
- **CD** - Change Directory
- **DOS** - Disk Operating System
- **ISO** - International Organization for Standardization
- **HOME** - Home Directory
- **PATH** - Path Variable
- **SKEL** - Skeleton
- **TMP** - Temporary
- **UMASK** - User Mask
- **GID** - Group Identifier
- **UID** - User Identifier
- **VI/VIM** - Visual/Vi Improved

- **Wayland** - Modern Display Protocol
- **PING** - Packet Internet Groper

Key Non-Acronym Terms

System Components

- **daemon** - Background service process
- **kernel** - Core operating system component
- **shell** - Command interpreter
- **terminal** - Text interface for system access
- **console** - Physical or virtual system interface

File System Terms

- **inode** - Index node storing file metadata
- **block** - Basic storage allocation unit
- **sector** - Physical disk storage unit
- **mount** - Attach filesystem to directory tree
- **umount** - Detach filesystem from directory tree

Process Terms

- **fork** - Create child process
- **exec** - Execute new program
- **zombie** - Defunct process awaiting cleanup
- **orphan** - Process whose parent has terminated
- **thread** - Lightweight process component

Network Terms

- **socket** - Network communication endpoint
- **port** - Network service identifier
- **interface** - Network connection point
- **gateway** - Network routing point
- **netmask** - Network address mask

Security Terms

- **context** - SELinux security label
- **boolean** - SELinux policy toggle
- **permission** - File access rights
- **capability** - Fine-grained privilege
- **audit** - Security event logging

Storage Terms

- **partition** - Disk subdivision
- **filesystem** - Data organization method
- **journal** - Filesystem transaction log
- **quota** - Storage usage limit
- **snapshot** - Point-in-time copy

Service Terms

- **unit** - Systemd management object
- **target** - Systemd state grouping
- **service** - Systemd daemon unit
- **timer** - Systemd scheduled task
- **socket** - Systemd activation unit

Flatpak Terms

- **remote** - Flatpak repository source
- **runtime** - Shared base libraries for Flatpak apps
- **app ID** - Reverse-DNS application identifier
- **sandbox** - Isolated app execution environment
- **portal** - D-Bus interface for controlled host access

This glossary covers essential acronyms and terms for RHCSA exam success. Review regularly to ensure familiarity with technical vocabulary.

3.4 RHCSA Study Guide Summary: Topics and Commands from Both EPUBs

Based on analysis of "RHCSA Red Hat Enterprise Linux 10" by Asghar Ghori (Dec 2025 edition) and "Red Hat RHCSA 9 Cert Guide" by Sander van Vugt.

Enhanced Study Resource: This summary has been expanded into the comprehensive RHCSA Synthesis knowledge base, which provides detailed modules for each topic with hands-on labs, troubleshooting guides, and exam strategies. **Start with the synthesis modules for the most comprehensive exam preparation.**

Book Structure Overview

Asghar Ghori RHCSA Book Structure (RHEL 10 Edition, Dec 2025)

22 Chapters with comprehensive exercises and labs

Chapters 1-4: Foundation Skills

- Chapter 1: Local Installation
- Chapter 2: Initial Interaction with the System
- Chapter 3: Working with Files and File Permissions
- Chapter 4: Basic File Permissions

Chapters 5-8: User and System Management

- Chapter 5: Basic User Management
- Chapter 6: Advanced User Management
- Chapter 7: The Bash Shell
- Chapter 8: Shell Scripting

Chapters 9-12: System Operations

- Chapter 9: Managing Services and Processes
- Chapter 10: System Processes and Job Control
- Chapter 11: Package Management
- Chapter 12: Flatpak Software Management

Chapters 13-16: Storage

- Chapter 13: Storage Management (Partitions and File Systems)
- Chapter 14: Advanced Storage (LVM)
- Chapter 15: Advanced Storage (LVM Thin Provisioning, Swap)
- Chapter 16: Boot Process, GRUB2, and the Linux Kernel

Chapters 17-22: Networking and Security

- Chapter 17: Networking, Network Devices, and Network Connections
- Chapter 18: Hostname Resolution and Time Synchronization
- Chapter 19: NFS and AutoFS
- Chapter 20: Firewall and System Security
- Chapter 21: SELinux
- Chapter 22: SSH and Remote Access

Note: The RHEL 10 edition replaces the Podman/containers chapter with Flatpak, adds LVM thin provisioning, elevates shell scripting to its own chapter, and merges NFS+AutoFS into a single chapter.

Sander van Vugt RHCSA Book Structure

26 Chapters organized in 5 parts

Part I: Basic System Management Tasks (Chapters 1-8)

- Chapter 1: Installing Red Hat Enterprise Linux Server
- Chapter 2: Using Essential Tools
- Chapter 3: Essential File Management Tools
- Chapter 4: Working with Text Files
- Chapter 5: Connecting to Red Hat Enterprise Linux Server
- Chapter 6: User and Group Management
- Chapter 7: Permissions Management
- Chapter 8: Configuring Networking

Part II: Operating Running Systems (Chapters 9-13)

- Chapter 9: Software Management
- Chapter 10: Managing Processes
- Chapter 11: Working with Systemd Services
- Chapter 12: Scheduling Tasks
- Chapter 13: Configuring Logging

Part III: Advanced System Administration Tasks (Chapters 14-20)

- Chapter 14: Managing Storage
- Chapter 15: Advanced Storage Management
- Chapter 16: Basic Kernel Management
- Chapter 17: Managing and Understanding the Boot Procedure
- Chapter 18: Essential Troubleshooting Skills
- Chapter 19: An Introduction to Bash Shell Scripting
- Chapter 20: Managing Software

Part IV: Managing Network Services (Chapters 21-25)

- Chapter 21: Configuring SSH
- Chapter 22: Managing SELinux
- Chapter 23: Configuring a Firewall
- Chapter 24: Accessing Network Storage
- Chapter 25: Configuring Time Services

Part V: RHCSA Practice Exams (Chapter 26)

- Chapter 26: Managing Containers

Topic-by-Topic Breakdown with Lab Exercises

1. System Installation and Initial Setup

Asghar Ghori Labs:

- Exercise 1-1: Download and Install VirtualBox Software
- Exercise 1-2: Download and Install RHEL
- Exercise 1-3: Logging In from Windows

- Lab 1-1: Build RHEL9-VM2 (server2)

Sander van Vugt Labs:

- Focus on automated installation methods
- Kickstart configuration
- Initial system configuration

Key Commands:

```
# System information
hostnamectl
uname -a
cat /etc/redhat-release
lscpu
free -h
df -h

# Initial configuration
nmtui
timedatectl set-timezone
localectl set-locale
```

2. File Management and Text Processing

Asghar Ghori Labs:

- Exercise 3-1: Create Compressed Archives
- Exercise 3-2: Create and Manage Hard Links
- Exercise 3-3: Create and Manage Soft Links
- Lab 3-1: Archive, List, and Restore Files
- Lab 3-2: Practice the vim Editor
- Lab 3-3: File and Directory Operations

Sander van Vugt Labs:

- Working with tar archives
- Text file manipulation with sed, awk, grep
- File linking and copying strategies

Key Commands:

```
# File operations
ls -la
cp -r source destination
mv source destination
rm -rf directory
find / -name "filename" -type f
locate filename

# Archives and compression
tar -czf archive.tar.gz directory/
tar -xzf archive.tar.gz
gzip file
gunzip file.gz

# Links
ln source hard_link
ln -s target symbolic_link

# Text processing
grep pattern file
sed 's/old/new/g' file
awk '{print $1}' file
sort file
uniq file
wc -l file

# Text editors
vim filename
nano filename
```

3. File Permissions and Security

Asghar Ghori Labs:

- Exercise 4-1: Modify Permission Bits Using Symbolic Form
- Exercise 4-2: Modify Permission Bits Using Octal Form
- Exercise 4-3: Test the Effect of setuid Bit on Executable Files
- Exercise 4-4: Test the Effect of setgid Bit on Executable Files
- Exercise 4-5: Set up Shared Directory for Group Collaboration
- Exercise 4-6: Test the Effect of Sticky Bit
- Lab 4-1: Manipulate File Permissions
- Lab 4-2: Configure Group Collaboration and Prevent File Deletion
- Lab 4-3: Find Files
- Lab 4-4: Find Files Using Different Criteria

Sander van Vugt Labs:

- Advanced permission scenarios
- ACL implementation
- Special permissions in practice

Key Commands:

```
# Basic permissions
chmod 755 file
chmod u+x,g+r,o-w file
chown user:group file
chgrp group file

# Special permissions
chmod +s file              # setuid/setgid
chmod +t directory         # sticky bit
chmod 4755 file           # setuid
chmod 2755 directory      # setgid
chmod 1755 directory      # sticky bit

# ACLs
setfacl -m u:username:rwx file
getfacl file
setfacl -x u:username file

# Finding files
find / -perm -4000         # setuid files
find / -perm -2000         # setgid files
find / -user username
find / -group groupname
```

4. User and Group Management

Asghar Ghori Labs:

- Exercise 5-1: Create a User Account with Default Attributes
- Exercise 5-2: Create a User Account with Custom Values
- Exercise 5-3: Modify and Delete a User Account
- Exercise 5-4: Create a User Account with No-Login Access
- Lab 5-1: Check User Login Attempts
- Lab 5-2: Verify User and Group Identity
- Exercise 6-3: Lock and Unlock a User Account with usermod and passwd

Sander van Vugt Labs:

- User account policies
- Group membership management
- Password aging configuration

Key Commands:

```
# User management
useradd -u UID -g GROUP -G GROUPS -s SHELL -d HOME username
usermod -aG group username
usermod -L username        # lock account
usermod -U username        # unlock account
userdel -r username

# Password management
passwd username
chage -M 90 -m 7 -W 7 username
chage -l username

# Group management
groupadd -g GID groupname
groupmod -n newname oldname
groupdel groupname
gpasswd -a username groupname

# User information
id username
groups username
who
w
last
lastb
```

5. Process and Service Management

Asghar Ghori Labs:

- Systemctl command exercises
- Service configuration labs
- Process monitoring and control

Sander van Vugt Labs:

- Systemd service creation
- Timer configuration

- Process priority management

Key Commands:

```
# Process management
ps aux
ps -ef
top
htop
pgrep process_name
pkill process_name
kill PID
killall process_name
jobs
bg
fg
nohup command &

# Process priority
nice -n 10 command
renice 5 PID

# System services
systemctl start service
systemctl stop service
systemctl restart service
systemctl reload service
systemctl enable service
systemctl disable service
systemctl status service
systemctl list-units --type=service
systemctl daemon-reload

# Systemd targets
systemctl set-default multi-user.target
systemctl get-default
systemctl isolate rescue.target
```

6. Package Management

Asghar Ghori Labs:

- DNF package management exercises
- Repository configuration
- RPM command usage

Sander van Vugt Labs:

- Advanced package queries

- Group package management
- Creating custom repositories

Key Commands:

```
# DNF package management
dnf install package
dnf update package
dnf remove package
dnf search keyword
dnf info package
dnf list installed
dnf list available
dnf group list
dnf group install "group name"
dnf history
dnf clean all

# Repository management
dnf config-manager --add-repo URL
dnf repolist
dnf config-manager --enable repo
dnf config-manager --disable repo

# RPM commands
rpm -qa                    # list all packages
rpm -qi package           # package info
rpm -ql package           # list files in package
rpm -qf /path/file        # which package owns file
rpm -ivh package.rpm      # install package
rpm -Uvh package.rpm      # upgrade package
```

7. Storage Management and LVM

Asghar Ghori Labs:

- LVM creation and management exercises
- File system creation and mounting
- Swap configuration labs

Sander van Vugt Labs:

- Advanced LVM scenarios
- Storage troubleshooting

Key Commands:

```
# Disk and partition management
lsblk
fdisk -l
fdisk /dev/device
parted /dev/device
partprobe

# LVM management
pvcreate /dev/device
pvdisplay
pvs
vgcreate vg_name /dev/device
vgdisplay
vgs
vgextend vg_name /dev/device
lvcreate -L size -n lv_name vg_name
lvdisplay
lvs
lvextend -L +size /dev/vg/lv
lvreduce -L -size /dev/vg/lv

# File systems
mkfs.xfs /dev/device
mkfs.ext4 /dev/device
mount /dev/device /mountpoint
umount /mountpoint
mount -a
df -h
du -sh directory

# File system resizing
xfs_growfs /mountpoint      # for XFS
resize2fs /dev/device       # for ext4

# Swap management
mkswap /dev/device
swapon /dev/device
swapoff /dev/device
swapon --show
```

8. Network Configuration

Asghar Ghori Labs:

- NetworkManager configuration with nmcli
- Static IP configuration
- Network troubleshooting exercises

Sander van Vugt Labs:

- Advanced networking scenarios
- Network bonding and teaming
- IPv6 configuration

Key Commands:

```
# Network information
ip addr show
ip route show
ip link show
nmcli device status
nmcli connection show

# Network configuration
nmcli con add type ethernet con-name NAME ifname DEVICE
nmcli con modify NAME ipv4.addresses IP/MASK
nmcli con modify NAME ipv4.gateway GATEWAY
nmcli con modify NAME ipv4.dns DNS1,DNS2
nmcli con modify NAME ipv4.method manual
nmcli con up NAME
nmcli con down NAME

# Network testing
ping host
traceroute host
nslookup hostname
dig hostname
ss -tuln
netstat -tuln
```

9. Firewall Configuration

Asghar Ghori Labs:

- firewall-cmd basic configuration
- Service and port management
- Rich rules implementation

Sander van Vugt Labs:

- Advanced firewall scenarios
- Custom service definitions
- Firewall troubleshooting

Key Commands:

```
# Firewall management
firewall-cmd --state
firewall-cmd --get-active-zones
firewall-cmd --list-all
firewall-cmd --add-service=SERVICE --permanent
firewall-cmd --remove-service=SERVICE --permanent
firewall-cmd --add-port=PORT/PROTOCOL --permanent
firewall-cmd --remove-port=PORT/PROTOCOL --permanent
firewall-cmd --reload

# Rich rules
firewall-cmd --add-rich-rule='rule family="ipv4" source address="IP" accept' --permanent

# Zone management
firewall-cmd --set-default-zone=ZONE
firewall-cmd --change-interface=INTERFACE --zone=ZONE --permanent
```

10. SELinux Management

Asghar Ghori Labs:

- SELinux mode configuration
- Context management exercises
- Boolean configuration
- Port labeling labs

Sander van Vugt Labs:

- Advanced SELinux troubleshooting
- Custom policy modules
- File context analysis

Key Commands:

```
# SELinux status and modes
getenforce
setenforce 0|1
sestatus

# File contexts
ls -Z file
restorecon -Rv /path
semanage fcontext -a -t TYPE "/path(/.*)?"
semanage fcontext -l | grep path

# Process contexts
ps -eZ
ps auxZ

# SELinux booleans
getsebool -a
getsebool boolean_name
setsebool -P boolean_name on|off

# Port contexts
semanage port -l
semanage port -a -t TYPE -p PROTOCOL PORT

# Troubleshooting
ausearch -m AVC -ts recent
sealert -a /var/log/audit/audit.log
```

11. Boot Process and GRUB

Asghar Ghori Labs:

- GRUB configuration modification
- Kernel parameter management
- Boot troubleshooting scenarios

Sander van Vugt Labs:

- Advanced boot procedures
- Systemd target management
- Recovery scenarios

Key Commands:

```
# GRUB management
grub2-editenv list
grub2-mkconfig -o /boot/grub2/grub.cfg
grub2-set-default "menu entry"

# Boot targets
systemctl get-default
systemctl set-default multi-user.target
systemctl list-units --type=target

# Kernel management
uname -r
rpm -qa kernel
dnf list installed kernel
```

12. Logging and Monitoring

Asghar Ghori Labs:

- Journald configuration
- Rsyslog setup
- Log rotation configuration

Sander van Vugt Labs:

- Advanced logging scenarios
- Remote logging setup
- Log analysis techniques

Key Commands:

```
# Journal management
journalctl
journalctl -u service
journalctl -f
journalctl --since "1 hour ago"
journalctl -p err
journalctl --list-boots

# Traditional logging
tail -f /var/log/messages
tail -f /var/log/secure
logger "test message"

# Log rotation
logrotate -d /etc/logrotate.conf
```

13. Scheduled Tasks

Asghar Ghori Labs:

- Crontab configuration
- At job scheduling
- Systemd timer creation

Sander van Vugt Labs:

- Advanced scheduling scenarios
- Timer unit configuration
- Anacron usage

Key Commands:

```
# Cron management
crontab -e
crontab -l
crontab -r
crontab -u username -e

# At scheduling
at now + 5 minutes
at 15:30
atq
atrm job_number

# Systemd timers
systemctl list-timers
systemctl enable timer.timer
systemctl start timer.timer
```

14. Container Management

Asghar Ghori Labs:

- Podman basic operations
- Container networking
- Container storage management

Sander van Vugt Labs:

- Advanced container scenarios
- Systemd integration
- Container image management

Key Commands:

```
# Container management
podman pull image
podman run -d --name NAME -p HOST:CONTAINER image
podman ps
podman ps -a
podman stop container
podman start container
podman rm container
podman rmi image

# Container systemd integration
podman generate systemd --new --files --name container
loginctl enable-linger username

# Container images
podman images
podman search term
podman inspect image
podman logs container
```

15. Network Services

Asghar Ghori Labs:

- NFS server and client configuration
- AutoFS implementation
- SSH configuration

Sander van Vugt Labs:

- Advanced NFS scenarios
- Time synchronization
- SSH key management

Key Commands:

```
# NFS management
systemctl enable --now nfs-server
exportfs -arv
exportfs -s
showmount -e server

# AutoFS
systemctl enable --now autofs
mount | grep autofs

# SSH configuration
ssh-keygen -t rsa
ssh-copy-id user@server
scp file user@server:/path
```

Command Summary by Category

User Management Commands

```
useradd, usermod, userdel, passwd, chage, chsh, chfn
groupadd, groupmod, groupdel, gpasswd
id, groups, who, w, last, lastb
```

File Management Commands

```
ls, cp, mv, rm, mkdir, rmdir, touch, find, locate
chmod, chown, chgrp, umask
setfacl, getfacl
tar, gzip, gunzip, zip, unzip
```

Process Management Commands

```
ps, top, htop, pgrep, pkill, kill, killall
jobs, bg, fg, nohup
nice, renice
```

System Service Commands

```
systemctl, journalctl
service, chkconfig (legacy)
```

Network Commands

```
nmcli, nmtui
ip, ifconfig (legacy)
ping, traceroute, nslookup, dig
ss, netstat (legacy)
```

Storage Commands

```
lsblk, fdisk, parted, partprobe
pvcreate, vgcreate, lvcreate, pvs, vgs, lvs
mkfs.xfs, mkfs.ext4, mount, umount
xfs_growfs, resize2fs
```

Package Management Commands

```
dnf, rpm, yum (legacy)
```

Security Commands

```
firewall-cmd
getenforce, setenforce, setsebool, restorecon, semanage
ausearch, sealert
```

Container Commands

```
podman, buildah, skopeo
```

This comprehensive summary covers all major topics and commands from both study guides, organized for efficient exam preparation.