📚 Linux & Networking Mastery Series - Complete Index

> **Your comprehensive guide from Linux newbie to network-aware power user**

🎯 Learning Path Overview

This series is designed to take you through a structured learning journey, building skills progressively from basic Linux commands to advanced networking and system administration.

📖 How to Use This Series

1. **Sequential Learning**: Follow chapters in order for best results
2. **Hands-on Practice**: Each chapter includes practical exercises
3. **Reference Material**: Use the cheat sheet for quick lookups
4. **Progressive Difficulty**: Concepts build upon previous chapters
5. **Real-world Focus**: Every topic includes practical applications

---

📋 Complete Chapter Index

- Special permissions (setuid, setgid, sticky bit)
- Access Control Lists (ACLs) and security best practices
- **Prerequisites**: Chapters 1-2
- **Time**: 2-3 hours
- **Difficulty**: Intermediate

04 - Process Management

- Understanding processes and system resources
- Process monitoring (`ps`, `top`, `htop`) and control
- Job control, signals, and background processes
- Service management with systemd
- **Prerequisites**: Chapters 1-3
- **Time**: 3-4 hours
- **Difficulty**: Intermediate

🔧 **System Administration** (Chapters 5-8)

05 - Networking Basics

- IP addressing (IPv4/IPv6) and subnetting fundamentals
- Network protocols (TCP/UDP, HTTP, DNS, SSH)
- OSI model and network components
- Home networking (NAT, DHCP, routing basics)
- **Prerequisites**: Chapters 1-4
- **Time**: 4-5 hours
- **Difficulty**: Intermediate

06 - System Administration

- User and group management at scale
- Package management (APT, YUM/DNF) and repositories
- System monitoring, logging, and maintenance
- Disk management, filesystems, and LVM
- **Prerequisites**: Chapters 1-5
- **Time**: 4-6 hours
- **Difficulty**: Intermediate-Advanced

07 - SSH & Remote Access

- SSH fundamentals and key-based authentication
- SSH configuration, hardening, and security
- Secure file transfers (SCP, SFTP, rsync)
- SSH tunneling and port forwarding techniques
- **Prerequisites**: Chapters 1-6
- **Time**: 3-4 hours
- **Difficulty**: Intermediate-Advanced

08 - Web Services

- HTTP/HTTPS protocols and web server fundamentals
- Apache and Nginx installation and configuration
- SSL/TLS certificates and HTTPS implementation
- Virtual hosts, load balancing, and performance optimization
- **Prerequisites**: Chapters 1-7
- **Time**: 4-5 hours
- **Difficulty**: Advanced

🌐 **Network Tools & Analysis** (Chapters 9-11)

09 - Basic Network Tools

- Essential diagnostic tools (`ping`, `traceroute`, `nslookup`)
- Network interface management (`ip`, `ifconfig`)
- Connection monitoring (`netstat`, `ss`) and analysis
- Practical troubleshooting workflows
- **Prerequisites**: Chapters 1-8
- **Time**: 3-4 hours
- **Difficulty**: Intermediate

10 - Advanced Network Tools

- Network discovery and scanning (`nmap`, `masscan`)
- Packet capture and analysis (`tcpdump`, `wireshark`)
- Firewall management (`iptables`, `ufw`) and traffic control
- Performance testing (`iperf3`, `mtr`) and optimization
- **Prerequisites**: Chapters 1-9
- **Time**: 4-6 hours
- **Difficulty**: Advanced

11 - DNS Deep Dive

- DNS hierarchy, resolution process, and record types
- DNS server configuration (BIND9) and management
- Advanced DNS features (load balancing, failover)
- DNS security (DNSSEC, filtering) and troubleshooting
- **Prerequisites**: Chapters 1-10
- **Time**: 4-5 hours
- **Difficulty**: Advanced

🔒 **Security & Advanced Topics** (Chapters 12-16)

12 - Security & Firewalls

- Network security fundamentals and threat landscape

- Firewall configuration (UFW, iptables) and rule management
- Intrusion detection systems (Fail2ban, OSSEC)
- Security scanning and vulnerability assessment
- **Prerequisites**: Chapters 1-11
- **Time**: 5-6 hours
- **Difficulty**: Advanced

13 - Performance Monitoring

- System resource monitoring (CPU, memory, disk, network)
- Performance monitoring tools (SAR, Nagios, Zabbix)
- Bottleneck identification and optimization strategies
- Automated monitoring and alerting systems
- **Prerequisites**: Chapters 1-12
- **Time**: 4-5 hours
- **Difficulty**: Advanced

14 - Automation & Scripting

- Bash scripting fundamentals and best practices
- Task automation (cron, systemd timers)
- Configuration management (Ansible) and Infrastructure as Code
- Log management and backup automation
- **Prerequisites**: Chapters 1-13
- **Time**: 5-7 hours
- **Difficulty**: Advanced

15 - Troubleshooting & Debugging

- Systematic troubleshooting methodology
- Advanced debugging tools (`strace`, `gdb`, `lsof`)
- Network debugging and packet analysis
- Log analysis, correlation, and root cause analysis
- **Prerequisites**: Chapters 1-14
- **Time**: 4-5 hours
- **Difficulty**: Advanced

16 - Best Practices

- Industry best practices and professional workflows
- Security hardening and compliance frameworks
- Performance optimization and capacity planning
- Career development and certification paths
- **Prerequisites**: All previous chapters
- **Time**: 3-4 hours
- **Difficulty**: Expert

📝 Reference Materials

Networking Cheat Sheet

- Quick reference for all essential commands
- Common network ports and protocols
- Troubleshooting workflows and decision trees
- Pro tips and advanced techniques
- **Type**: Reference
- **Use**: Throughout the series

🎯 Learning Tracks

Track 1: Linux System Administrator (8-10 weeks)

```
Chapters: 1 → 2 → 3 → 4 → 6 → 14 → 15 → 16
Focus: System administration, automation, troubleshooting
Career: Linux System Administrator, DevOps Engineer
```

Track 2: Network Engineer (10-12 weeks)

```
Chapters: 1 → 5 → 9 → 10 → 11 → 12 → 13 → 15 → 16
Focus: Networking, security, performance monitoring
Career: Network Engineer, Network Security Specialist
```

Track 3: Security Specialist (12-14 weeks)

```
Chapters: 1 → 2 → 3 → 5 → 7 → 9 → 10 → 12 → 15 → 16
Focus: Security hardening, monitoring, incident response
Career: Security Analyst, Penetration Tester
```

Track 4: Full-Stack (Complete) (16-20 weeks)

```
Chapters: 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 13 → 14 → 15 → 16
Focus: Comprehensive Linux and networking mastery
Career: Senior System Administrator, DevOps Architect, Technical Lead
```

📊 Progress Tracking

Beginner Level (Chapters 1-4)

- [ ] Chapter 1: Linux Basics
- [ ] Chapter 2: File Operations
- [ ] Chapter 3: Permissions & Ownership
- [ ] Chapter 4: Process Management

Milestone: Can navigate Linux systems confidently and manage basic system tasks

Intermediate Level (Chapters 5-8)

- [ ] Chapter 5: Networking Basics
- [ ] Chapter 6: System Administration
- [ ] Chapter 7: SSH & Remote Access
- [ ] Chapter 8: Web Services

Milestone: Can administer Linux systems and understand networking fundamentals

Advanced Level (Chapters 9-12)

- [ ] Chapter 9: Basic Network Tools
- [ ] Chapter 10: Advanced Network Tools
- [ ] Chapter 11: DNS Deep Dive
- [ ] Chapter 12: Security & Firewalls

Milestone: Can troubleshoot complex network issues and implement security measures

Expert Level (Chapters 13-16)

- [ ] Chapter 13: Performance Monitoring
- [ ] Chapter 14: Automation & Scripting
- [ ] Chapter 15: Troubleshooting & Debugging
- [ ] Chapter 16: Best Practices

Milestone: Can design, implement, and maintain enterprise-grade Linux and networking solutions

---

🛠 Prerequisites and Setup

System Requirements

- Linux system (Ubuntu 20.04+ recommended) or Linux VM
- At least 4GB RAM and 20GB disk space
- Internet connection for package installation
- Administrative (sudo) access

Recommended Lab Setup

```
# Virtual machines for practice:
# 1. Ubuntu Server (main learning environment)
# 2. CentOS/RHEL (for package management differences)
# 3. Network simulation tools (GNS3, Packet Tracer)
# 4. Monitoring tools (Nagios, Zabbix)
```

Essential Tools to Install

```
# Basic tools
sudo apt update && sudo apt install -y \
    curl wget vim htop tree git \
    net-tools iputils-ping traceroute \
    nmap tcpdump wireshark-cli \
    fail2ban ufw iptables-persistent

# Monitoring tools
sudo apt install -y \
    sysstat iotop nload \
    prometheus-node-exporter \
    rsyslog logrotate

# Development tools
sudo apt install -y \
    build-essential python3-pip \
    ansible docker.io \
    nginx apache2-utils
```

🎓 Certification Preparation

This series prepares you for:

Linux Certifications

- **CompTIA Linux+**: Chapters 1-6, 14
- **LPIC-1**: Chapters 1-8, 13-14
- **RHCSA**: Chapters 1-6, 14 (with RHEL-specific practice)

Networking Certifications

- **CompTIA Network+**: Chapters 5, 9-12
- **Cisco CCNA**: Chapters 5, 9-11 (with Cisco-specific practice)

Security Certifications

- **CompTIA Security+**: Chapters 7, 12, 16
- **CEH**: Chapters 10, 12, 15

📞 Support and Community

Getting Help

- Review the troubleshooting sections in each chapter
- Check the networking cheat sheet for quick references
- Practice in a safe lab environment before production
- Join Linux and networking communities for support

Contributing

- Report errors or suggest improvements
- Share your learning experiences and tips
- Create additional practice scenarios
- Help others in their learning journey

🚀 Ready to Start?

Begin your journey with Chapter 1: Linux Basics

Remember:

- Take your time with each chapter
- Practice extensively in lab environments
- Don't skip the hands-on exercises
- Build real projects to reinforce learning
- Stay curious and keep exploring!

> **"The expert in anything was once a beginner."** - Start your Linux and networking mastery journey today! 🐧🌐

🚀 Linux & Networking Command Cheat Sheet

> **Quick reference for essential Linux and networking commands - your go-to guide for daily operations**

📋 Table of Contents

🖥 Basic Linux Commands

Navigation & File System

```
# Directory navigation
pwd                     # Show current directory
ls -la                  # List files with details
cd /path/to/dir         # Change directory
cd ~                    # Go to home directory
cd -                    # Go to previous directory

# File operations
cp file1 file2          # Copy file
mv file1 file2          # Move/rename file
rm file                 # Delete file
rm -rf directory        # Delete directory recursively
mkdir directory         # Create directory
rmdir directory         # Remove empty directory

# File viewing
cat file                # Display file content
less file               # View file with pagination
head -n 10 file         # Show first 10 lines
tail -n 10 file         # Show last 10 lines
tail -f file            # Follow file changes

# File searching
find /path -name "*.txt" # Find files by name
grep "pattern" file      # Search text in file
grep -r "pattern" /dir   # Recursive text search
locate filename         # Find file by name (fast)
```

Permissions & Ownership

```
# View permissions
ls -l file              # Show file permissions
stat file               # Detailed file info

# Change permissions
chmod 755 file          # rwxr-xr-x
chmod u+x file          # Add execute for user
chmod g-w file          # Remove write for group
chmod o=r file          # Set read-only for others

# Change ownership
```

```
chown user:group file   # Change owner and group
chown user file         # Change owner only
chgrp group file        # Change group only
sudo chown -R user:group /dir  # Recursive ownership change
```

🗁 File Operations

Text Processing

```
# Text manipulation
cut -d':' -f1 /etc/passwd     # Extract first field
awk '{print $1}' file         # Print first column
sed 's/old/new/g' file        # Replace text
sort file                     # Sort lines
uniq file                     # Remove duplicates
wc -l file                    # Count lines

# File comparison
diff file1 file2              # Compare files
comm file1 file2              # Compare sorted files

# Archives
tar -czf archive.tar.gz dir   # Create compressed archive
tar -xzf archive.tar.gz       # Extract archive
zip -r archive.zip dir        # Create zip archive
unzip archive.zip             # Extract zip
```

File Transfer

```
# Local transfer
cp -r source/ dest/           # Copy directory
rsync -av source/ dest/       # Sync directories

# Remote transfer
scp file user@host:/path      # Copy file to remote
scp user@host:/path file      # Copy file from remote
rsync -av dir/ user@host:/path # Sync to remote

# Download
wget https://example.com/file # Download file
curl -O https://example.com/file # Download with curl
```

🖥 System Information

Hardware & System

```
# System info
uname -a                # System information
hostname                # System hostname
uptime                  # System uptime
whoami                  # Current user
id                      # User and group IDs

# Hardware info
lscpu                   # CPU information
lsmem                   # Memory information
lsblk                   # Block devices
lsusb                   # USB devices
lspci                   # PCI devices

# Memory & disk
free -h                 # Memory usage
df -h                   # Disk usage
du -sh /path            # Directory size
```

Performance Monitoring

```
# Process monitoring
top                     # Real-time processes
htop                    # Enhanced process viewer
ps aux                  # All running processes
ps -ef | grep process   # Find specific process

# System load
w                       # Who is logged in and load
vmstat 1                # Virtual memory statistics
iostat 1                # I/O statistics
sar -u 1 10             # CPU usage over time
```

⚙ Process Management

Process Control

```
# Start/stop processes
nohup command &         # Run in background
command &               # Run in background
jobs                    # List background jobs
fg %1                   # Bring job to foreground
bg %1                   # Send job to background

# Kill processes
kill PID                # Terminate process
kill -9 PID             # Force kill process
```

```
killall process_name    # Kill by name
pkill -f pattern        # Kill by pattern

# Process priority
nice -n 10 command      # Start with lower priority
renice 10 PID           # Change process priority
```

Service Management

```
# Systemd services
sudo systemctl start service    # Start service
sudo systemctl stop service     # Stop service
sudo systemctl restart service  # Restart service
sudo systemctl enable service   # Enable at boot
sudo systemctl disable service  # Disable at boot
sudo systemctl status service   # Check service status
sudo systemctl list-units       # List all services
```

🌐 Network Interface Management

Interface Information

```
# Modern commands (ip)
ip addr show            # Show all interfaces
ip addr show eth0       # Show specific interface
ip -4 addr show         # Show IPv4 only
ip -6 addr show         # Show IPv6 only
ip link show            # Show link status

# Legacy commands (ifconfig)
ifconfig                # Show all interfaces
ifconfig eth0           # Show specific interface
ifconfig -a             # Show all interfaces (including down)
```

Interface Configuration

```
# Bring interface up/down
sudo ip link set eth0 up
sudo ip link set eth0 down
sudo ifconfig eth0 up
sudo ifconfig eth0 down

# Configure IP address
sudo ip addr add 192.168.1.100/24 dev eth0
sudo ip addr del 192.168.1.100/24 dev eth0
sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0

# Configure routes
ip route show           # Show routing table
sudo ip route add 10.0.0.0/8 via 192.168.1.1
sudo ip route del 10.0.0.0/8
sudo ip route add default via 192.168.1.1
```

🏓 Connectivity Testing

Basic Connectivity

```
# Ping tests
ping google.com         # Basic connectivity test
ping -c 4 google.com    # Ping 4 times
ping -i 0.5 google.com  # Ping every 0.5 seconds
ping -s 1000 google.com # Ping with 1000 byte packets
ping6 google.com        # IPv6 ping

# Path tracing
traceroute google.com   # Trace route to destination
traceroute -I google.com # Use ICMP instead of UDP
traceroute -T google.com # Use TCP
traceroute6 google.com  # IPv6 traceroute
mtr google.com          # Real-time traceroute
```

Port Testing

```
# Test port connectivity
telnet google.com 80    # Test TCP port
nc -zv google.com 80    # Test port with netcat
nc -zvu google.com 53   # Test UDP port
timeout 5 bash -c '</dev/tcp/google.com/80' # Bash TCP test

# HTTP testing
curl -I http://google.com       # HTTP headers only
curl -w "%{time_total}" google.com # Show timing
wget --spider google.com        # Test without downloading
```

🔍 DNS Operations

DNS Queries

```
# Basic DNS lookup
nslookup google.com     # Basic DNS query
dig google.com          # Detailed DNS query
dig +short google.com   # Short answer only
host google.com         # Simple DNS lookup

# Specific record types
dig A google.com        # IPv4 address
dig AAAA google.com     # IPv6 address
dig MX google.com       # Mail exchange
dig NS google.com       # Name servers
dig TXT google.com      # Text records
dig SOA google.com      # Start of authority

# Reverse DNS
dig -x 8.8.8.8          # Reverse lookup
nslookup 8.8.8.8        # Reverse lookup
```

Advanced DNS

```
# DNS tracing
dig +trace google.com   # Trace DNS resolution
dig +trace +additional google.com # Include additional records

# Query specific server
dig @8.8.8.8 google.com # Query Google DNS
dig @1.1.1.1 google.com # Query Cloudflare DNS

# DNS cache management
sudo systemd-resolve --flush-caches # Clear DNS cache
sudo systemctl restart systemd-resolved # Restart DNS service
systemd-resolve --status # Show DNS status
```

📊 Network Monitoring

Connection Monitoring

```
# Active connections
netstat -tulpn          # All listening ports with processes
netstat -an             # All connections (numerical)
netstat -i              # Interface statistics
netstat -r              # Routing table

# Modern alternative (ss)
ss -tulpn               # All listening ports
ss -an                  # All connections
ss -o state established # Established connections
ss -s                   # Socket statistics
```

Traffic Analysis

```
# Real-time monitoring
iftop                   # Real-time bandwidth usage
nload                   # Network load monitor
bmon                    # Bandwidth monitor
watch -n 1 'cat /proc/net/dev' # Interface statistics

# Packet capture
sudo tcpdump -i eth0    # Capture all traffic
sudo tcpdump -i eth0 port 80 # Capture HTTP traffic
sudo tcpdump -i eth0 host 192.168.1.1 # Capture specific host
sudo tcpdump -w capture.pcap # Save to file
tcpdump -r capture.pcap # Read from file
```

🛡 Firewall Management

iptables

```
# View rules
sudo iptables -L        # List all rules
sudo iptables -L -n -v  # Numerical and verbose
sudo iptables -L INPUT  # List INPUT chain

# Basic rules
sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT # Allow SSH
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT # Allow HTTP
sudo iptables -A INPUT -s 192.168.1.100 -j ACCEPT # Allow IP
sudo iptables -A INPUT -j DROP # Drop everything else

# Delete rules
sudo iptables -D INPUT 1 # Delete rule number 1
sudo iptables -F        # Flush all rules

# Save/restore
sudo iptables-save > rules.txt
sudo iptables-restore < rules.txt
```

ufw (Ubuntu Firewall)

```
# Basic operations
sudo ufw enable         # Enable firewall
sudo ufw disable        # Disable firewall
sudo ufw status         # Show status
sudo ufw status numbered # Show numbered rules

# Allow/deny rules
sudo ufw allow 22       # Allow SSH
sudo ufw allow ssh      # Allow SSH (by name)
sudo ufw allow 80/tcp   # Allow HTTP
sudo ufw allow from 192.168.1.0/24 # Allow subnet
sudo ufw deny 23        # Deny telnet

# Delete rules
sudo ufw delete allow 80
sudo ufw delete 1       # Delete rule number 1
sudo ufw --force reset  # Reset all rules
```

🔬 Advanced Network Tools

Network Scanning (nmap)

```
# Host discovery
nmap -sn 192.168.1.0/24 # Ping scan
nmap 192.168.1.1-10     # Scan range
nmap -F 192.168.1.1     # Fast scan (top 100 ports)

# Port scanning
nmap -p 22,80,443 192.168.1.1 # Specific ports
nmap -p- 192.168.1.1    # All ports
nmap -sS 192.168.1.1    # SYN scan
nmap -sU 192.168.1.1    # UDP scan

# Service detection
nmap -sV 192.168.1.1    # Version detection
nmap -O 192.168.1.1     # OS detection
nmap -A 192.168.1.1     # Aggressive scan
nmap --script vuln 192.168.1.1 # Vulnerability scan
```

Packet Analysis

```
# tcpdump filters
sudo tcpdump -i eth0 'tcp port 80' # HTTP traffic
sudo tcpdump -i eth0 'udp port 53'  # DNS traffic
sudo tcpdump -i eth0 'icmp'         # ICMP traffic
sudo tcpdump -i eth0 'host 192.168.1.1' # Specific host
sudo tcpdump -i eth0 'net 192.168.1.0/24' # Subnet

# Wireshark command line (tshark)
tshark -i eth0          # Live capture
tshark -r capture.pcap  # Read file
tshark -Y "http"        # Display filter
tshark -T fields -e ip.src -e ip.dst # Extract fields
```

📈 Performance Testing

Bandwidth Testing

```
# iperf3 (requires server)
iperf3 -s               # Server mode
iperf3 -c server_ip     # Client mode
iperf3 -c server_ip -u  # UDP test
iperf3 -c server_ip -R  # Reverse test
iperf3 -c server_ip -P 4 # Parallel streams

# Speed test
speedtest-cli           # Internet speed test
wget -O /dev/null http://speedtest.wdc01.softlayer.com/downloads/test100.zip
```

Latency Testing

```
# Continuous monitoring
ping -i 0.2 google.com  # High frequency ping
mtr --report-cycles 100 google.com # MTR report

# Jitter testing
ping -c 100 google.com | grep 'time=' | awk -F'time=' '{print $2}' | awk -F' '
'{print $1}'
```

🔒 Security & Scanning

Security Scanning

```
# Port scanning
nmap -sS -O target      # Stealth scan with OS detection
nmap --script vuln target # Vulnerability scan
nmap --script safe target # Safe scripts only

# SSL/TLS testing
nmap --script ssl-cert target
nmap --script ssl-enum-ciphers target
openssl s_client -connect target:443

# Web security
```

```
nmap --script http-enum target
nmap --script http-vuln* target
curl -I https://target  # Check headers
```

Network Security

```
# Monitor connections
ss -tulpn | grep LISTEN # Listening services
lsof -i                 # Open network files
netstat -an | grep ESTABLISHED # Active connections

# Check for suspicious activity
last                    # Login history
who                     # Currently logged in
w                       # What users are doing
journalctl -f           # System logs
```

🔧 Troubleshooting Workflows

Network Connectivity Issues

```
# Step 1: Check local interface
ip addr show
ping 127.0.0.1

# Step 2: Check local network
ping $(ip route | grep default | awk '{print $3}')

# Step 3: Check DNS
nslookup google.com
ping 8.8.8.8

# Step 4: Check internet
ping google.com
curl -I http://google.com

# Step 5: Check specific service
telnet target 80
nc -zv target 80
```

DNS Issues

```
# Check DNS configuration
cat /etc/resolv.conf
systemd-resolve --status
```

```
# Test different DNS servers
dig @8.8.8.8 domain.com
dig @1.1.1.1 domain.com

# Clear DNS cache
sudo systemd-resolve --flush-caches
sudo systemctl restart systemd-resolved

# Check DNS propagation
dig +trace domain.com
```

Performance Issues

```
# Check bandwidth
iperf3 -c speedtest.server.com
speedtest-cli

# Check latency
ping -c 10 target
mtr --report target

# Check packet loss
ping -c 100 target | grep 'packet loss'

# Check interface errors
ip -s link show
cat /proc/net/dev
```

🔌 Common Network Ports

Well-Known Ports

```
20/21   FTP (File Transfer Protocol)
22      SSH (Secure Shell)
23      Telnet
25      SMTP (Simple Mail Transfer Protocol)
53      DNS (Domain Name System)
67/68   DHCP (Dynamic Host Configuration Protocol)
80      HTTP (Hypertext Transfer Protocol)
110     POP3 (Post Office Protocol v3)
143     IMAP (Internet Message Access Protocol)
443     HTTPS (HTTP Secure)
993     IMAPS (IMAP Secure)
995     POP3S (POP3 Secure)
```

Application Ports

```
3306    MySQL
5432    PostgreSQL
6379    Redis
27017   MongoDB
3389    RDP (Remote Desktop Protocol)
5900    VNC (Virtual Network Computing)
8080    HTTP Alternative
8443    HTTPS Alternative
9200    Elasticsearch
5601    Kibana
```

🌐 IP Address Ranges

Private IP Ranges (RFC 1918)

```
10.0.0.0/8        (10.0.0.0 - 10.255.255.255)
172.16.0.0/12     (172.16.0.0 - 172.31.255.255)
192.168.0.0/16    (192.168.0.0 - 192.168.255.255)
```

Special IP Addresses

```
127.0.0.0/8       Loopback (localhost)
169.254.0.0/16    Link-local (APIPA)
224.0.0.0/4       Multicast
0.0.0.0/0         Default route (any address)
255.255.255.255   Broadcast
```

Subnet Masks (CIDR)

```
/8  = 255.0.0.0       (16,777,214 hosts)
/16 = 255.255.0.0     (65,534 hosts)
/24 = 255.255.255.0   (254 hosts)
/25 = 255.255.255.128 (126 hosts)
/26 = 255.255.255.192 (62 hosts)
/27 = 255.255.255.224 (30 hosts)
/28 = 255.255.255.240 (14 hosts)
/29 = 255.255.255.248 (6 hosts)
/30 = 255.255.255.252 (2 hosts)
```

🎯 Quick Reference Scripts

Network Health Check

```bash
#!/bin/bash
# Quick network health check
echo "=== Network Health Check ==="
echo "Interface Status:"
ip addr show | grep -E '^[0-9]+:|inet '
echo "\nDefault Gateway:"
ip route | grep default
echo "\nDNS Servers:"
cat /etc/resolv.conf | grep nameserver
echo "\nConnectivity Test:"
ping -c 1 8.8.8.8 > /dev/null && echo "✅ Internet OK" || echo "❌ Internet FAIL"
echo "\nDNS Test:"
nslookup google.com > /dev/null && echo "✅ DNS OK" || echo "❌ DNS FAIL"
```

Port Scanner

```bash
#!/bin/bash
# Simple port scanner
HOST="$1"
for PORT in 22 23 25 53 80 110 143 443 993 995; do
    timeout 1 bash -c "</dev/tcp/$HOST/$PORT" 2>/dev/null &&
    echo "Port $PORT: Open" || echo "Port $PORT: Closed"
done
```

Bandwidth Monitor

```bash
#!/bin/bash
# Simple bandwidth monitor
INTERFACE="eth0"
while true; do
    RX1=$(cat /sys/class/net/$INTERFACE/statistics/rx_bytes)
    TX1=$(cat /sys/class/net/$INTERFACE/statistics/tx_bytes)
    sleep 1
    RX2=$(cat /sys/class/net/$INTERFACE/statistics/rx_bytes)
    TX2=$(cat /sys/class/net/$INTERFACE/statistics/tx_bytes)

    RX_RATE=$(((RX2-RX1)/1024))
    TX_RATE=$(((TX2-TX1)/1024))

    echo "$(date): RX: ${RX_RATE} KB/s, TX: ${TX_RATE} KB/s"
done
```

💡 Pro Tips

Command Aliases

```
# Add to ~/.bashrc or ~/.zshrc
alias ll='ls -la'
alias la='ls -la'
alias myip='curl -s ifconfig.me'
alias ports='netstat -tulpn'
alias listening='ss -tulpn'
alias connections='ss -tuln'
alias flushdns='sudo systemd-resolve --flush-caches'
alias netcheck='ping -c 4 8.8.8.8 && nslookup google.com'
```

Useful One-liners

```
# Find your public IP
curl -s ifconfig.me
curl -s ipinfo.io/ip

# Show all listening ports
ss -tulpn | grep LISTEN

# Find process using specific port
lsof -i :80
ss -tulpn | grep :80

# Monitor network connections
watch -n 1 'ss -tuln'

# Check if port is open
nc -zv google.com 80

# Quick HTTP test
curl -w "@curl-format.txt" -o /dev/null -s http://example.com

# DNS lookup with timing
time nslookup google.com

# Show routing table
ip route show
route -n

# Monitor interface traffic
watch -n 1 'cat /proc/net/dev'
```

> 🎓 **Congratulations!** You now have a comprehensive reference for Linux and networking commands. Bookmark this page and refer to it whenever you need quick command syntax or troubleshooting

> workflows. Practice these commands regularly to build muscle memory and become a networking power user! 🚀

Remember: Always test commands in a safe environment first, especially those that modify system configuration. When in doubt, check the man pages (`man command`) for detailed documentation.

🐧 Linux Basics: Foundation Skills

> **Master the terminal, file system navigation, and essential commands**

📖 What You'll Learn

Linux is the backbone of the internet, powering everything from web servers to smartphones. Understanding Linux fundamentals is essential for any technical professional. In this chapter, you'll master:

- Terminal navigation and basic commands
- Linux file system structure
- File and directory operations
- Basic permissions and ownership
- Essential shortcuts and productivity tips

🌍 Why This Matters

Real-world applications:

- **Web Development**: Deploy applications on Linux servers
- **DevOps**: Automate infrastructure and deployments
- **System Administration**: Manage servers and services
- **Cybersecurity**: Analyze systems and investigate incidents
- **Data Science**: Process large datasets on Linux clusters

💻 Getting Started: The Terminal

Opening the Terminal

Ubuntu/Debian:

```
# Keyboard shortcut
Ctrl + Alt + T

# Or search for "Terminal" in applications
```

Your First Command:

```
$ whoami
username
```

```
$ pwd
/home/username
```

Understanding the Prompt

```
username@hostname:~$
#   ^        ^     ^  ^
#   |        |     |  |
#   user   computer |  command prompt
#                  |
#                current directory (~ = home)
```

🗂 Linux File System Structure

The Root Directory (`/`)

```
$ ls /
bin   dev  home  lib64  mnt  proc  run   srv  tmp  var
boot  etc  lib   media  opt  root  sbin  sys  usr
```

Key Directories Explained

Directory	Purpose	Example Contents
`/home`	User home directories	`/home/john`, `/home/mary`
`/etc`	System configuration files	`/etc/passwd`, `/etc/hosts`
`/var`	Variable data (logs, databases)	`/var/log`, `/var/www`
`/usr`	User programs and libraries	`/usr/bin`, `/usr/local`
`/tmp`	Temporary files	Session files, cache
`/bin`	Essential system binaries	`ls`, `cp`, `mv`
`/sbin`	System administration binaries	`iptables`, `mount`

Visualizing the Structure

```
$ tree -L 2 /
/
├── bin -> usr/bin
├── boot
│   ├── grub
│   └── vmlinuz
├── dev
│   ├── sda1
│   └── tty1
├── etc
│   ├── passwd
│   └── hosts
├── home
│   ├── john
│   └── mary
└── var
    ├── log
    └── www
```

🧭 Navigation Commands

Basic Navigation

```
# Print working directory
$ pwd
/home/username

# List directory contents
$ ls
Documents  Downloads  Pictures  Videos

# List with details
$ ls -l
total 16
drwxr-xr-x 2 user user 4096 Dec 15 10:30 Documents
drwxr-xr-x 2 user user 4096 Dec 15 10:30 Downloads
drwxr-xr-x 2 user user 4096 Dec 15 10:30 Pictures
drwxr-xr-x 2 user user 4096 Dec 15 10:30 Videos

# List including hidden files
$ ls -la
total 24
drwxr-xr-x 3 user user 4096 Dec 15 10:30 .
drwxr-xr-x 3 root root 4096 Dec 15 10:00 ..
-rw-r--r-- 1 user user  220 Dec 15 10:00 .bash_logout
-rw-r--r-- 1 user user 3771 Dec 15 10:00 .bashrc
drwxr-xr-x 2 user user 4096 Dec 15 10:30 Documents
```

Change Directory

```
# Go to home directory
$ cd
$ cd ~
$ cd /home/username

# Go to specific directory
$ cd /var/log
$ pwd
/var/log

# Go back one level
$ cd ..
$ pwd
/var

# Go back to previous directory
$ cd -
/var/log

# Relative vs Absolute paths
$ cd Documents            # Relative path
$ cd /home/user/Documents  # Absolute path
```

Path Shortcuts

Symbol	Meaning	Example
`~`	Home directory	`cd ~/Documents`
`.`	Current directory	`./script.sh`
`..`	Parent directory	`cd ../..`
`-`	Previous directory	`cd -`
`/`	Root directory	`cd /`

🗀 File and Directory Operations

Creating Files and Directories

```
# Create empty file
$ touch newfile.txt
$ ls -l newfile.txt
-rw-r--r-- 1 user user 0 Dec 15 11:00 newfile.txt

# Create multiple files
$ touch file1.txt file2.txt file3.txt

# Create directory
$ mkdir projects
$ ls -ld projects
drwxr-xr-x 2 user user 4096 Dec 15 11:01 projects

# Create nested directories
$ mkdir -p projects/web/frontend
$ tree projects
projects
```

```
└── web
    └── frontend
```

Copying Files and Directories

```
# Copy file
$ cp file1.txt file1_backup.txt

# Copy file to directory
$ cp file1.txt projects/

# Copy directory recursively
$ cp -r projects projects_backup

# Copy with verbose output
$ cp -v file1.txt file1_copy.txt
'file1.txt' -> 'file1_copy.txt'

# Copy preserving attributes
$ cp -p file1.txt file1_preserve.txt
```

Moving and Renaming

```
# Rename file
$ mv file1.txt renamed_file.txt

# Move file to directory
$ mv renamed_file.txt projects/

# Move and rename simultaneously
$ mv file2.txt projects/project_file.txt

# Move directory
$ mv projects_backup archive/
```

Removing Files and Directories

```
# Remove file
$ rm file3.txt

# Remove multiple files
$ rm *.txt

# Remove directory (empty)
$ rmdir empty_directory
```

```
# Remove directory and contents
$ rm -r projects_backup

# Remove with confirmation
$ rm -i important_file.txt
rm: remove regular file 'important_file.txt'? y

# Force remove (be careful!)
$ rm -rf dangerous_directory
```

🔍 Viewing File Contents

Basic File Viewing

```
# Display entire file
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...

# Display with line numbers
$ cat -n /etc/passwd
     1  root:x:0:0:root:/root:/bin/bash
     2  daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
...

# View first 10 lines
$ head /var/log/syslog
Dec 15 10:00:01 hostname systemd[1]: Started Daily apt download activities.
Dec 15 10:00:01 hostname systemd[1]: Starting Daily apt upgrade and clean
activities...
...

# View last 10 lines
$ tail /var/log/syslog
Dec 15 11:45:01 hostname CRON[1234]: (root) CMD (command -v debian-sa1 > /dev/null
&& debian-sa1 1 1)
...

# Follow file changes (useful for logs)
$ tail -f /var/log/syslog
# Press Ctrl+C to stop
```

Paging Through Files

```
# View file page by page
$ less /var/log/syslog
# Navigation:
# Space: next page
# b: previous page
# /search_term: search
# q: quit

# Alternative pager
$ more /var/log/syslog
```

🔧 Essential Commands Reference

File Information

```
# File type and permissions
$ file /bin/ls
/bin/ls: ELF 64-bit LSB executable, x86-64

# Disk usage of files/directories
$ du -h projects/
4.0K    projects/web/frontend
8.0K    projects/web
12K projects/

# File size
$ ls -lh file1.txt
-rw-r--r-- 1 user user 1.2K Dec 15 11:00 file1.txt

# Count lines, words, characters
$ wc /etc/passwd
  45   65 2419 /etc/passwd
# lines words chars filename
```

System Information

```
# Current user
$ whoami
username

# User ID information
$ id
uid=1000(username) gid=1000(username)
groups=1000(username),4(adm),24(cdrom),27(sudo)

# Current date and time
$ date
Fri Dec 15 11:30:45 UTC 2023

# System uptime
$ uptime
 11:30:45 up  2:15,  1 user,  load average: 0.08, 0.03, 0.01
```

```
# Disk space usage
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        20G  5.5G   14G  30% /
/dev/sda2       100G   45G   50G  48% /home
```

⚠ Common Pitfalls and Tips

1. Case Sensitivity

```
# Linux is case-sensitive!
$ ls Documents  # ✅ Correct
$ ls documents  # ❌ Error if directory is "Documents"
ls: cannot access 'documents': No such file or directory
```

2. Spaces in Filenames

```
# Use quotes or escape spaces
$ touch "my file.txt"        # ✅ Correct
$ touch my\ file.txt          # ✅ Correct
$ touch my file.txt          # ❌ Creates two files: "my" and "file.txt"
```

3. Hidden Files

```
# Files starting with . are hidden
$ ls               # Won't show .bashrc
$ ls -a            # Shows all files including hidden ones
```

4. Tab Completion

```
# Use Tab to auto-complete
$ cd Doc<Tab>      # Completes to "Documents"
$ ls /etc/pas<Tab> # Completes to "/etc/passwd"
```

🎯 Practical Scenario: Setting Up a Project

Let's create a typical project structure:

```
# 1. Create project directory
$ mkdir -p ~/projects/my-website
$ cd ~/projects/my-website
```

```
# 2. Create project structure
$ mkdir -p {src,docs,tests,config}
$ touch README.md
$ touch src/{index.html,style.css,script.js}
$ touch docs/setup.md
$ touch config/settings.conf

# 3. Verify structure
$ tree
.
├── README.md
├── config
│   └── settings.conf
├── docs
│   └── setup.md
├── src
│   ├── index.html
│   ├── script.js
│   └── style.css
└── tests

# 4. Add some content
$ echo "# My Website Project" > README.md
$ echo "<!DOCTYPE html><html><head><title>My Site</title></head><body><h1>Hello
World!</h1></body></html>" > src/index.html

# 5. Check our work
$ cat README.md
# My Website Project

$ cat src/index.html
<!DOCTYPE html><html><head><title>My Site</title></head><body><h1>Hello World!
</h1></body></html>
```

Productivity Tips

Command History

```
# View command history
$ history
  1  ls
  2  cd projects
  3  mkdir my-website
  ...

# Repeat last command
$ !!

# Repeat command by number
$ !3
```

```
# Search history
$ Ctrl+R
(reverse-i-search)`cd': cd projects
```

Keyboard Shortcuts

Shortcut	Action
`Ctrl+C`	Cancel current command
`Ctrl+D`	Exit terminal/logout
`Ctrl+L`	Clear screen
`Ctrl+A`	Move to beginning of line
`Ctrl+E`	Move to end of line
`Ctrl+U`	Delete from cursor to beginning
`Ctrl+K`	Delete from cursor to end
`Tab`	Auto-complete
`↑/↓`	Navigate command history

Aliases for Efficiency

```
# Add to ~/.bashrc for permanent aliases
$ echo 'alias ll="ls -la"' >> ~/.bashrc
$ echo 'alias la="ls -la"' >> ~/.bashrc
$ echo 'alias ..="cd .."' >> ~/.bashrc
$ source ~/.bashrc

# Now you can use:
$ ll        # Instead of ls -la
$ ..        # Instead of cd ..
```

🧠 Knowledge Check

Quick Quiz

1. **What command shows your current directory?**

   ▶ Answer

   ```
   pwd
   ```

2. **How do you create a directory called "test" and navigate into it in one line?**

▶ Answer

```
mkdir test && cd test
```

3. **What's the difference between `rm file.txt` and `rm -r directory/`?**

▶ Answer

- `rm file.txt` removes a single file
- `rm -r directory/` removes a directory and all its contents recursively

4. **How do you view the last 20 lines of a file called "logfile.txt"?**

▶ Answer

```
tail -n 20 logfile.txt
# or
tail -20 logfile.txt
```

Hands-On Challenges

Challenge 1: File System Explorer

```
# Navigate to root directory and explore
# Find the largest directory in /var
# List all files in /etc that contain "conf" in their name
```

Challenge 2: Project Setup

```
# Create a project structure for a blog:
# blog/
# ├── posts/
# ├── images/
# ├── css/
# ├── js/
# └── index.html
```

Challenge 3: File Operations

```
# Create 5 text files with different extensions
# Copy all .txt files to a backup directory
# Rename all .log files to have today's date in the filename
```

🚀 Next Steps

Congratulations! You've mastered Linux basics. You can now:

- Navigate the Linux file system confidently
- Create, copy, move, and delete files and directories
- View file contents and system information
- Use essential productivity shortcuts

Ready for the next level? Continue to 02-file-operations.md to master advanced file manipulation, text processing, and powerful command-line editors.

> **Pro Tip**: The best way to learn Linux is by using it daily. Set up a Linux virtual machine or use WSL on Windows to practice these commands regularly. Remember: every expert was once a beginner! 🐧

🗀 File Operations: Text Processing Mastery

> **Master file manipulation, text processing, and command-line editors**

📖 What You'll Learn

File operations are the heart of Linux system administration and development. This chapter covers advanced file manipulation, powerful text processing tools, and essential editors that every Linux user must know:

- Advanced file searching and filtering
- Text processing with `grep`, `sed`, and `awk`
- File comparison and merging
- Command-line editors (`nano`, `vim`)
- File permissions and ownership
- Symbolic and hard links

🌍 Why This Matters

Real-world applications:

- **Log Analysis**: Parse and analyze system logs for troubleshooting
- **Configuration Management**: Edit config files on remote servers
- **Data Processing**: Clean and transform text data
- **Automation**: Create scripts that process files automatically
- **Security**: Analyze files for suspicious content or changes

🔍 Advanced File Searching

Finding Files with `find`

```
# Find files by name
$ find /home -name "*.txt"
/home/user/documents/notes.txt
```

```
/home/user/projects/readme.txt

# Find files by type
$ find /var/log -type f -name "*.log"
/var/log/syslog
/var/log/auth.log
/var/log/kern.log

# Find directories
$ find /etc -type d -name "*conf*"
/etc/apparmor.d
/etc/systemd/system.conf.d

# Find files by size
$ find /var -size +100M
/var/log/huge_log.log
/var/cache/large_file.cache

# Find files modified in last 7 days
$ find /home -mtime -7

# Find files by permissions
$ find /usr/bin -perm 755
```

Advanced `find` Examples

```
# Find and execute command on results
$ find /tmp -name "*.tmp" -exec rm {} \;

# Find large files and show their sizes
$ find /var -size +50M -exec ls -lh {} \;

# Find files owned by specific user
$ find /home -user john -type f

# Find files with specific permissions
$ find /etc -perm 644 -type f

# Find empty files
$ find /tmp -empty -type f

# Find files modified more than 30 days ago
$ find /var/log -mtime +30 -name "*.log"
```

Using `locate` for Fast Searches

```
# Update locate database (run as root)
$ sudo updatedb
```

```
# Fast file search
$ locate nginx.conf
/etc/nginx/nginx.conf
/usr/share/doc/nginx/examples/nginx.conf

# Case-insensitive search
$ locate -i README
/home/user/projects/readme.md
/usr/share/doc/README
```

🔎 Text Processing with `grep`

Basic `grep` Usage

```
# Search for pattern in file
$ grep "error" /var/log/syslog
Dec 15 10:30:15 server kernel: [12345.678] USB disconnect, address 1
Dec 15 10:31:22 server systemd[1]: Failed to start service

# Case-insensitive search
$ grep -i "ERROR" /var/log/syslog

# Show line numbers
$ grep -n "failed" /var/log/auth.log
23:Dec 15 10:25:30 server sshd[1234]: Failed password for user from 192.168.1.100
45:Dec 15 10:26:15 server sshd[1235]: Failed password for admin from 192.168.1.101

# Count matches
$ grep -c "ssh" /var/log/auth.log
127

# Show context around matches
$ grep -A 2 -B 2 "error" /var/log/syslog
# -A 2: show 2 lines after
# -B 2: show 2 lines before
# -C 2: show 2 lines before and after
```

Advanced `grep` Patterns

```
# Regular expressions
$ grep "^Dec 15" /var/log/syslog          # Lines starting with "Dec 15"
$ grep "error$" /var/log/syslog           # Lines ending with "error"
$ grep "[0-9]\{1,3\}\.[0-9]\{1,3\}" /var/log/auth.log  # IP addresses

# Multiple patterns
$ grep -E "error|warning|critical" /var/log/syslog

# Exclude patterns
$ grep -v "info" /var/log/syslog          # Show all lines except those with
```

```
"info"

# Recursive search in directories
$ grep -r "TODO" /home/user/projects/
/home/user/projects/app.js:15:// TODO: Add error handling
/home/user/projects/README.md:23:TODO: Update documentation

# Search in specific file types
$ grep -r --include="*.py" "import" /home/user/projects/
```

Practical `grep` Examples

```
# Find failed login attempts
$ grep "Failed password" /var/log/auth.log

# Find all IP addresses in logs
$ grep -oE "[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}" /var/log/auth.log

# Find processes using specific port
$ netstat -tulpn | grep ":80 "

# Search for configuration errors
$ grep -i "error\|warning\|fail" /etc/nginx/nginx.conf
```

✂ Text Manipulation with `sed`

Basic `sed` Operations

```
# Replace first occurrence per line
$ sed 's/old/new/' file.txt

# Replace all occurrences
$ sed 's/old/new/g' file.txt

# Replace and save to new file
$ sed 's/old/new/g' file.txt > newfile.txt

# Edit file in place
$ sed -i 's/old/new/g' file.txt

# Replace with backup
$ sed -i.bak 's/old/new/g' file.txt
```

Advanced `sed` Examples

```
# Delete lines containing pattern
$ sed '/pattern/d' file.txt

# Delete empty lines
$ sed '/^$/d' file.txt

# Print specific lines
$ sed -n '10,20p' file.txt            # Print lines 10-20
$ sed -n '/pattern/p' file.txt        # Print lines matching pattern

# Insert text before/after pattern
$ sed '/pattern/i\New line before' file.txt
$ sed '/pattern/a\New line after' file.txt

# Multiple operations
$ sed -e 's/old1/new1/g' -e 's/old2/new2/g' file.txt
```

Practical sed Use Cases

```
# Remove comments from config file
$ sed '/^#/d' /etc/ssh/sshd_config

# Change IP address in config
$ sed -i 's/192.168.1.100/192.168.1.200/g' /etc/hosts

# Add line numbers
$ sed = file.txt | sed 'N;s/\n/\t/'

# Convert DOS line endings to Unix
$ sed -i 's/\r$//' file.txt

# Extract email addresses
$ sed -n 's/.*\([a-zA-Z0-9._%+-]\+@[a-zA-Z0-9.-]\+\.[a-zA-Z]\{2,\}\).*/\1/p'
file.txt
```

🔧 Text Processing with awk

Basic awk Concepts

```
# Print specific columns
$ awk '{print $1, $3}' /etc/passwd       # Print 1st and 3rd columns

# Print with custom separator
$ awk -F: '{print $1, $3}' /etc/passwd   # Use : as field separator

# Print lines matching condition
$ awk '$3 >= 1000' /etc/passwd           # Users with UID >= 1000
```

```
# Count lines
$ awk 'END {print NR}' file.txt

# Sum numbers in column
$ awk '{sum += $1} END {print sum}' numbers.txt
```

Advanced `awk` Examples

```
# Process log files
$ awk '$9 == 404 {print $1, $7}' /var/log/apache2/access.log

# Calculate averages
$ awk '{sum += $3; count++} END {print sum/count}' data.txt

# Format output
$ awk -F: '{printf "%-20s %s\n", $1, $5}' /etc/passwd

# Conditional processing
$ awk '$3 > 1000 {print $1 " is a regular user"}' /etc/passwd

# Multiple conditions
$ awk '$3 >= 1000 && $7 != "/usr/sbin/nologin" {print $1}' /etc/passwd
```

Real-World `awk` Scripts

```
# Analyze web server logs
$ awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr
# Shows most frequent IP addresses

# Process CSV files
$ awk -F, '{print $2, $4}' data.csv

# Calculate disk usage summary
$ df -h | awk 'NR>1 {gsub(/%/, "", $5); if($5 > 80) print $6 " is " $5 "% full"}'

# Monitor system load
$ uptime | awk '{print "Load average: " $(NF-2) " " $(NF-1) " " $NF}'
```

📝 Command-Line Editors

`nano` - Beginner-Friendly Editor

```
# Open file in nano
$ nano filename.txt

# Nano shortcuts (shown at bottom of screen):
```

```
# Ctrl+O: Save (WriteOut)
# Ctrl+X: Exit
# Ctrl+W: Search
# Ctrl+K: Cut line
# Ctrl+U: Paste
# Ctrl+G: Help
```

Nano Configuration:

```
# Create ~/.nanorc for custom settings
$ cat > ~/.nanorc << EOF
set linenumbers
set mouse
set smooth
set tabsize 4
include /usr/share/nano/*.nanorc
EOF
```

`vim` - Powerful Modal Editor

Vim Modes

- **Normal Mode**: Navigation and commands (default)
- **Insert Mode**: Text editing
- **Visual Mode**: Text selection
- **Command Mode**: Execute commands

Basic Vim Commands

```
# Open file
$ vim filename.txt

# Mode switching:
# i: Enter insert mode
# Esc: Return to normal mode
# v: Enter visual mode
# :: Enter command mode

# Navigation (Normal mode):
# h,j,k,l: Left, down, up, right
# w: Next word
# b: Previous word
# 0: Beginning of line
# $: End of line
# gg: First line
# G: Last line
# :n: Go to line n
# Editing (Normal mode):
# x: Delete character
# dd: Delete line
# yy: Copy line
# p: Paste
# u: Undo
# Ctrl+r: Redo

# Search and replace:
# /pattern: Search forward
# ?pattern: Search backward
# n: Next match
# N: Previous match
# :%s/old/new/g: Replace all

# Save and exit:
# :w: Save
# :q: Quit
# :wq: Save and quit
# :q!: Quit without saving
```

Vim Configuration

```
# Create ~/.vimrc for custom settings
$ cat > ~/.vimrc << EOF
set number          " Show line numbers
set tabstop=4       " Tab width
set shiftwidth=4    " Indent width
set expandtab       " Use spaces instead of tabs
set hlsearch        " Highlight search results
set ignorecase      " Case-insensitive search
set smartcase       " Smart case search
syntax on           " Enable syntax highlighting
set mouse=a         " Enable mouse support
EOF
```

🔐 File Permissions and Ownership

Understanding Permissions

```
# View permissions
$ ls -l file.txt
-rw-r--r-- 1 user group 1024 Dec 15 10:30 file.txt
# ^^^^^^^^^^
# |||||||++-- Other permissions (r--)
# ||||+++---- Group permissions (r--)
# |+++------- User permissions (rw-)
# +---------- File type (-=file, d=directory, l=link)
```

Permission Types

Symbol	Permission	Numeric	Meaning
`r`	Read	4	View file contents
`w`	Write	2	Modify file
`x`	Execute	1	Run file as program
`-`	No permission	0	No access

Changing Permissions

```
# Symbolic method
$ chmod u+x script.sh              # Add execute for user
$ chmod g-w file.txt               # Remove write for group
$ chmod o+r file.txt               # Add read for others
$ chmod a+x script.sh              # Add execute for all

# Numeric method
$ chmod 755 script.sh              # rwxr-xr-x
$ chmod 644 file.txt               # rw-r--r--
$ chmod 600 private.txt            # rw-------
$ chmod 777 public_dir/            # rwxrwxrwx (dangerous!)

# Recursive permissions
$ chmod -R 755 /var/www/html/
```

Common Permission Patterns

Permissions	Numeric	Use Case
`rwx------`	700	Private executable
`rwxr-xr-x`	755	Public executable
`rw-------`	600	Private file
`rw-r--r--`	644	Public readable file
`rw-rw-r--`	664	Group writable file

Changing Ownership

```
# Change owner
$ sudo chown newuser file.txt

# Change owner and group
$ sudo chown newuser:newgroup file.txt
```

```
# Change only group
$ sudo chgrp newgroup file.txt

# Recursive ownership change
$ sudo chown -R www-data:www-data /var/www/

# Copy permissions from another file
$ chmod --reference=source.txt target.txt
```

🔗 Links: Symbolic and Hard

Hard Links

```
# Create hard link
$ ln original.txt hardlink.txt

# Both files point to same data
$ ls -li original.txt hardlink.txt
123456 -rw-r--r-- 2 user group 1024 Dec 15 10:30 original.txt
123456 -rw-r--r-- 2 user group 1024 Dec 15 10:30 hardlink.txt
#      ^
#      Same inode number

# Deleting original doesn't affect hard link
$ rm original.txt
$ cat hardlink.txt  # Still works
```

Symbolic Links

```
# Create symbolic link
$ ln -s /path/to/original.txt symlink.txt

# View symbolic link
$ ls -l symlink.txt
lrwxrwxrwx 1 user group 20 Dec 15 10:30 symlink.txt -> /path/to/original.txt

# Create relative symbolic link
$ ln -s ../config/settings.conf current_settings.conf

# Update symbolic link
$ ln -sfn /new/path/file.txt symlink.txt
```

Link Differences

Feature	Hard Link	Symbolic Link

Feature	Hard Link	Symbolic Link
Cross filesystems	No	Yes
Link to directories	No	Yes
Original deleted	Still works	Broken link
Shows in `ls -l`	No	Yes
Disk space	None	Minimal

📊 File Comparison and Merging

Comparing Files

```
# Simple comparison
$ diff file1.txt file2.txt
2c2
< This is line 2 in file1
---
> This is line 2 in file2

# Side-by-side comparison
$ diff -y file1.txt file2.txt
Line 1 is same                    Line 1 is same
This is line 2 in file1         | This is line 2 in file2
Line 3 is same                    Line 3 is same

# Unified diff format
$ diff -u file1.txt file2.txt
--- file1.txt   2023-12-15 10:30:00.000000000 +0000
+++ file2.txt   2023-12-15 10:31:00.000000000 +0000
@@ -1,3 +1,3 @@
 Line 1 is same
-This is line 2 in file1
+This is line 2 in file2
 Line 3 is same

# Compare directories
$ diff -r dir1/ dir2/
```

Advanced Comparison Tools

```
# Compare ignoring whitespace
$ diff -w file1.txt file2.txt

# Compare ignoring case
$ diff -i file1.txt file2.txt

# Show only if files differ
```

```
$ diff -q file1.txt file2.txt
Files file1.txt and file2.txt differ

# Context diff
$ diff -c file1.txt file2.txt
```

🎯 Practical Scenario: Log Analysis

Let's analyze a web server log file:

```
# 1. Create sample log file
$ cat > access.log << EOF
192.168.1.100 - - [15/Dec/2023:10:30:15 +0000] "GET /index.html HTTP/1.1" 200 1024
192.168.1.101 - - [15/Dec/2023:10:30:16 +0000] "GET /about.html HTTP/1.1" 200 2048
192.168.1.100 - - [15/Dec/2023:10:30:17 +0000] "GET /contact.php HTTP/1.1" 404 512
192.168.1.102 - - [15/Dec/2023:10:30:18 +0000] "POST /login.php HTTP/1.1" 200 256
192.168.1.100 - - [15/Dec/2023:10:30:19 +0000] "GET /admin.php HTTP/1.1" 403 128
EOF

# 2. Find all 404 errors
$ grep " 404 " access.log
192.168.1.100 - - [15/Dec/2023:10:30:17 +0000] "GET /contact.php HTTP/1.1" 404 512

# 3. Count requests per IP
$ awk '{print $1}' access.log | sort | uniq -c
      3 192.168.1.100
      1 192.168.1.101
      1 192.168.1.102

# 4. Find most requested pages
$ awk '{print $7}' access.log | sort | uniq -c | sort -nr
      1 /index.html
      1 /contact.php
      1 /admin.php
      1 /about.html
      1 /login.php

# 5. Calculate total bytes transferred
$ awk '{sum += $10} END {print "Total bytes:", sum}' access.log
Total bytes: 3968

# 6. Find suspicious activity (multiple failed attempts)
$ awk '$9 >= 400 {print $1, $7, $9}' access.log
192.168.1.100 /contact.php 404
192.168.1.100 /admin.php 403

# 7. Extract unique IP addresses
$ awk '{print $1}' access.log | sort -u
192.168.1.100
192.168.1.101
192.168.1.102
```

⚠ Common Pitfalls and Best Practices

1. Backup Before Editing

```
# Always backup important files
$ cp /etc/important.conf /etc/important.conf.backup
$ sed -i.bak 's/old/new/g' /etc/important.conf
```

2. Test Regular Expressions

```
# Test grep patterns before using in scripts
$ echo "test string" | grep "pattern"
```

3. Use Quotes with Special Characters

```
# Protect special characters
$ grep "\$variable" file.txt
$ find . -name "*.tmp"
```

4. Understand File Permissions

```
# Check permissions before changing
$ ls -l file.txt
$ chmod 644 file.txt  # Don't use 777 unless necessary
```

🧠 Knowledge Check

Quick Quiz

1. **How do you search for all files larger than 100MB in /var?**

   ▶ Answer

   ```
   find /var -size +100M
   ```

2. **What's the difference between `>` and `>>` when redirecting output?**

   ▶ Answer
   - `>` overwrites the file
   - `>>` appends to the file

3. **How do you replace all occurrences of "old" with "new" in a file using sed?**

   ▶ Answer

   ```
   sed -i 's/old/new/g' filename
   ```

4. **What permission number gives read and write to owner, read to group, and no access to others?**

   ▶ Answer

   ```
   640 (rw-r-----)
   ```

Hands-On Challenges

Challenge 1: Log Processing

```
# Create a script that:
# 1. Finds all ERROR entries in /var/log/syslog
# 2. Counts how many errors occurred today
# 3. Saves unique error messages to a file
```

Challenge 2: Configuration Management

```
# 1. Create a backup of /etc/hosts
# 2. Add a new entry: "127.0.0.1 myapp.local"
# 3. Verify the change was made correctly
```

Challenge 3: File Cleanup

```
# Create a script that:
# 1. Finds all .tmp files older than 7 days
# 2. Lists them with their sizes
# 3. Asks for confirmation before deleting
```

🚀 Next Steps

Excellent work! You've mastered advanced file operations and text processing. You can now:

- Search and filter files efficiently
- Process text data with grep, sed, and awk
- Edit files with nano and vim
- Manage file permissions and ownership

- Compare and analyze files

Ready to dive deeper? Continue to 03-process-management.md to learn about managing processes, services, and system resources.

> **Pro Tip**: Text processing skills are invaluable for automation and data analysis. Practice these commands with real log files and configuration files to build muscle memory. The combination of grep, sed, and awk can solve most text processing challenges! 🔧

🔐 File Permissions & Ownership: Security Fundamentals

> **Master Linux file permissions, ownership, and access control for secure system administration**

📖 What You'll Learn

File permissions and ownership are the foundation of Linux security. This chapter covers everything you need to know about controlling access to files and directories:

- Understanding Linux permission model
- Reading and interpreting permission strings
- Changing permissions with `chmod`
- Managing ownership with `chown` and `chgrp`
- Special permissions (setuid, setgid, sticky bit)
- Access Control Lists (ACLs)
- Security best practices
- Troubleshooting permission issues

🌍 Why This Matters

Critical applications:

- **Security**: Protect sensitive files and system resources
- **Multi-user Systems**: Control access in shared environments
- **Web Servers**: Secure web applications and content
- **System Administration**: Maintain proper system security
- **Compliance**: Meet security requirements and standards

🏗️ Understanding Linux Permissions

Permission Model Overview

```
# Linux uses a simple but powerful permission model:
# Every file/directory has:
# 1. Owner (user who owns the file)
# 2. Group (group that owns the file)
# 3. Others (everyone else)
# Each category has three permissions:
# r = read (4)
# w = write (2)
# x = execute (1)

# Example permission string:
# -rwxr-xr--
# ||||||||||
# |||||||||+-- Others: read only (r--)
# ||||||||+--- Others: no write
# |||||||+---- Others: no execute
# ||||||+----- Group: read and execute (r-x)
# |||||+------ Group: no write
# ||||+------- Group: execute
# |||+-------- Owner: read, write, execute (rwx)
# ||+--------- Owner: write
# +----------- File type: - (regular file)
```

Viewing Permissions

```
# Basic permission viewing
$ ls -l
total 12
-rw-r--r-- 1 user group  1024 Dec 15 10:30 document.txt
drwxr-xr-x 2 user group  4096 Dec 15 10:31 directory
-rwxr-xr-x 1 user group  2048 Dec 15 10:32 script.sh

# Detailed breakdown:
# -rw-r--r-- 1 user group  1024 Dec 15 10:30 document.txt
# |          | |    |     |    |           |
# |          | |    |     |    |           +-- Filename
# |          | |    |     |    +-- Modification time
# |          | |    |     +-- File size
# |          | |    +-- Group owner
# |          | +-- User owner
# |          +-- Number of hard links
# +-- Permissions

# Show permissions in octal format
$ stat -c "%a %n" *
644 document.txt
755 directory
755 script.sh

# Show detailed file information
$ stat document.txt
  File: document.txt
  Size: 1024        Blocks: 8          IO Block: 4096   regular file
Device: 801h/2049d  Inode: 123456      Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/    user)   Gid: ( 1000/   group)
```

```
Access: 2023-12-15 10:30:00.000000000 +0000
Modify: 2023-12-15 10:30:00.000000000 +0000
Change: 2023-12-15 10:30:00.000000000 +0000
```

File Type Indicators

```
# First character indicates file type:
-  Regular file
d  Directory
l  Symbolic link
c  Character device
b  Block device
p  Named pipe (FIFO)
s  Socket

# Examples:
$ ls -l /dev/ | head -5
crw-rw-rw- 1 root tty     5,   0 Dec 15 10:30 tty
brw-rw---- 1 root disk    8,   0 Dec 15 10:30 sda
lrwxrwxrwx 1 root root        15 Dec 15 10:30 stdout -> /proc/self/fd/1
prw-r--r-- 1 root root         0 Dec 15 10:30 mypipe
srwxrwxrwx 1 root root         0 Dec 15 10:30 socket
```

🔧 Changing Permissions with chmod

Symbolic Mode

```
# Symbolic mode uses letters and symbols:
# u = user (owner)
# g = group
# o = others
# a = all (user + group + others)

# + = add permission
# - = remove permission
# = = set exact permission

# Examples:
$ chmod u+x script.sh         # Add execute for user
$ chmod g-w document.txt      # Remove write for group
$ chmod o=r document.txt      # Set others to read-only
$ chmod a+r document.txt      # Add read for all
$ chmod u+rwx,g+rx,o+r file  # Multiple permissions

# Before and after examples:
$ ls -l script.sh
-rw-r--r-- 1 user group 1024 Dec 15 10:30 script.sh

$ chmod u+x script.sh
$ ls -l script.sh
-rwxr--r-- 1 user group 1024 Dec 15 10:30 script.sh

$ chmod g+w,o-r script.sh
$ ls -l script.sh
-rwxrw---- 1 user group 1024 Dec 15 10:30 script.sh
```

Numeric (Octal) Mode

```
# Numeric mode uses three digits (0-7):
# Each digit represents permissions for user, group, others
# 4 = read (r)
# 2 = write (w)
# 1 = execute (x)

# Common permission combinations:
# 0 = --- (no permissions)
# 1 = --x (execute only)
# 2 = -w- (write only)
# 3 = -wx (write + execute)
# 4 = r-- (read only)
# 5 = r-x (read + execute)
# 6 = rw- (read + write)
# 7 = rwx (read + write + execute)

# Examples:
$ chmod 755 script.sh        # rwxr-xr-x
$ chmod 644 document.txt     # rw-r--r--
$ chmod 600 private.txt      # rw-------
$ chmod 777 shared_dir       # rwxrwxrwx (dangerous!)
$ chmod 000 locked_file      # --------- (no access)

# Practical examples:
$ chmod 755 ~/bin/*          # Make all scripts executable
$ chmod 644 *.txt            # Standard file permissions
$ chmod 700 ~/.ssh           # Secure SSH directory
$ chmod 600 ~/.ssh/id_rsa    # Secure private key
```

Recursive Permission Changes

```
# Apply permissions recursively to directories
$ chmod -R 755 /var/www/html # Web directory permissions
$ chmod -R 644 /var/www/html/*.html # HTML files

# Set directory and file permissions differently:
$ find /path -type d -exec chmod 755 {} \;  # Directories: 755
$ find /path -type f -exec chmod 644 {} \;  # Files: 644

# One-liner for web directories:
```

```
$ find /var/www/html -type d -exec chmod 755 {} \; -o -type f -exec chmod 644 {} \;

# Using chmod with find for specific file types:
$ find . -name "*.sh" -exec chmod +x {} \;   # Make all .sh files executable
$ find . -name "*.txt" -exec chmod 644 {} \; # Set standard permissions for text files
```

👥 Managing Ownership

Changing File Ownership

```
# Change owner only
$ sudo chown newuser file.txt
$ ls -l file.txt
-rw-r--r-- 1 newuser group 1024 Dec 15 10:30 file.txt

# Change owner and group
$ sudo chown newuser:newgroup file.txt
$ ls -l file.txt
-rw-r--r-- 1 newuser newgroup 1024 Dec 15 10:30 file.txt

# Change ownership recursively
$ sudo chown -R apache:apache /var/www/html

# Copy ownership from another file
$ sudo chown --reference=template.txt file.txt

# Change ownership of symbolic links
$ sudo chown -h user:group symlink  # Change link itself, not target
```

Changing Group Ownership

```
# Change group only
$ sudo chgrp developers project_dir
$ ls -ld project_dir
drwxr-xr-x 2 user developers 4096 Dec 15 10:30 project_dir

# Change group recursively
$ sudo chgrp -R www-data /var/www

# Copy group from another file
$ sudo chgrp --reference=template.txt file.txt
```

Practical Ownership Examples

```
# Web server setup
$ sudo chown -R www-data:www-data /var/www/html
$ sudo chmod -R 755 /var/www/html

# Database files
$ sudo chown -R mysql:mysql /var/lib/mysql
$ sudo chmod -R 750 /var/lib/mysql

# Log files
$ sudo chown -R syslog:adm /var/log
$ sudo chmod -R 640 /var/log/*.log

# User home directory
$ sudo chown -R user:user /home/user
$ sudo chmod 755 /home/user
$ sudo chmod 700 /home/user/.ssh
```

🔒 Special Permissions

Setuid (Set User ID)

```
# Setuid allows a file to run with owner's privileges
# Represented by 's' in user execute position

# Set setuid bit
$ sudo chmod u+s /usr/bin/passwd
$ ls -l /usr/bin/passwd
-rwsr-xr-x 1 root root 68208 Dec 15 10:30 /usr/bin/passwd

# Numeric method (add 4000)
$ sudo chmod 4755 /usr/bin/passwd

# Remove setuid
$ sudo chmod u-s /usr/bin/passwd

# Find setuid files (security audit)
$ find / -type f -perm -4000 2>/dev/null
/usr/bin/passwd
/usr/bin/sudo
/usr/bin/su
```

Setgid (Set Group ID)

```
# Setgid on files: run with group's privileges
# Setgid on directories: new files inherit directory's group

# Set setgid on directory
$ sudo chmod g+s /shared/project
$ ls -ld /shared/project
drwxrwsr-x 2 user developers 4096 Dec 15 10:30 /shared/project

# Files created in this directory inherit the group
$ touch /shared/project/newfile.txt
$ ls -l /shared/project/newfile.txt
-rw-r--r-- 1 user developers 0 Dec 15 10:30 newfile.txt

# Numeric method (add 2000)
$ sudo chmod 2775 /shared/project

# Find setgid files and directories
$ find / -type f -perm -2000 2>/dev/null
$ find / -type d -perm -2000 2>/dev/null
```

Sticky Bit

```
# Sticky bit on directories: only owner can delete files
# Common on /tmp directory

$ ls -ld /tmp
drwxrwxrwt 10 root root 4096 Dec 15 10:30 /tmp
#         ^
#         +-- Sticky bit (t)

# Set sticky bit
$ sudo chmod +t /shared/temp
$ ls -ld /shared/temp
drwxrwxrwt 2 root root 4096 Dec 15 10:30 /shared/temp

# Numeric method (add 1000)
$ sudo chmod 1777 /shared/temp

# Remove sticky bit
$ sudo chmod -t /shared/temp
```

Combined Special Permissions

```
# All special permissions combined
$ sudo chmod 7755 special_file  # setuid + setgid + sticky + 755
$ ls -l special_file
-rwsrwsrwt 1 user group 1024 Dec 15 10:30 special_file

# Understanding the numbers:
# 7755 = 4000 (setuid) + 2000 (setgid) + 1000 (sticky) + 755 (rwxr-xr-x)
```

📋 Access Control Lists (ACLs)

Basic ACL Operations

```
# Check if filesystem supports ACLs
$ mount | grep acl
/dev/sda1 on / type ext4 (rw,relatime,acl,user_xattr)

# View ACLs
$ getfacl file.txt
# file: file.txt
# owner: user
# group: group
user::rw-
group::r--
other::r--

# Set ACL for specific user
$ setfacl -m u:alice:rw file.txt
$ getfacl file.txt
# file: file.txt
# owner: user
# group: group
user::rw-
user:alice:rw-
group::r--
mask::rw-
other::r--

# Set ACL for specific group
$ setfacl -m g:developers:rwx project_dir

# Set default ACLs for directories
$ setfacl -d -m g:developers:rwx project_dir
$ setfacl -d -m o::r-x project_dir
```

Advanced ACL Management

```
# Remove specific ACL entry
$ setfacl -x u:alice file.txt

# Remove all ACLs
$ setfacl -b file.txt

# Copy ACLs from one file to another
$ getfacl file1.txt | setfacl --set-file=- file2.txt

# Recursive ACL operations
$ setfacl -R -m g:developers:rwx /project

# Set ACLs with mask
$ setfacl -m m::rw file.txt  # Limit maximum permissions
```

```
# Backup and restore ACLs
$ getfacl -R /project > acl_backup.txt
$ setfacl --restore=acl_backup.txt
```

🛡 Security Best Practices

Principle of Least Privilege

```
# Give minimum necessary permissions

# Bad: World-writable files
$ chmod 777 file.txt  # DON'T DO THIS!

# Good: Specific permissions
$ chmod 644 file.txt  # Read-write for owner, read for others
$ chmod 755 script.sh # Executable for owner, read-execute for others

# Secure directories
$ chmod 700 ~/.ssh           # SSH directory
$ chmod 600 ~/.ssh/id_rsa    # Private key
$ chmod 644 ~/.ssh/id_rsa.pub # Public key
$ chmod 644 ~/.ssh/authorized_keys # Authorized keys

# Web application security
$ chmod 755 /var/www/html    # Web root
$ chmod 644 /var/www/html/*.html # HTML files
$ chmod 600 /var/www/html/config.php # Config files
```

Common Security Patterns

```
# System configuration files
$ sudo chmod 644 /etc/passwd     # World-readable
$ sudo chmod 600 /etc/shadow     # Root-only
$ sudo chmod 644 /etc/group      # World-readable
$ sudo chmod 600 /etc/gshadow    # Root-only

# Log files
$ sudo chmod 640 /var/log/*.log  # Owner read-write, group read
$ sudo chmod 600 /var/log/auth.log # Sensitive logs

# Temporary directories
$ sudo chmod 1777 /tmp           # Sticky bit for shared temp
$ chmod 700 ~/tmp                # Private temp directory

# Backup files
$ chmod 600 backup_*.tar.gz      # Secure backups
$ chmod 700 backup_scripts/      # Secure backup scripts
```

Permission Auditing

```bash
#!/bin/bash
# security-audit.sh - Basic permission audit

echo "=== Security Audit ==="
echo

# Find world-writable files (potential security risk)
echo "World-writable files:"
find / -type f -perm -002 2>/dev/null | head -10
echo

# Find setuid files
echo "Setuid files:"
find / -type f -perm -4000 2>/dev/null
echo

# Find setgid files
echo "Setgid files:"
find / -type f -perm -2000 2>/dev/null
echo

# Find files with no owner
echo "Files with no owner:"
find / -nouser 2>/dev/null | head -10
echo

# Find files with no group
echo "Files with no group:"
find / -nogroup 2>/dev/null | head -10
echo

echo "=== Audit Complete ==="
```

🔧 Troubleshooting Permission Issues

Common Permission Problems

```
# Problem: Permission denied
$ ./script.sh
bash: ./script.sh: Permission denied

# Solution: Add execute permission
$ chmod +x script.sh
$ ./script.sh
Hello World!

# Problem: Cannot write to file
```

```
$ echo "test" > file.txt
bash: file.txt: Permission denied

# Check permissions
$ ls -l file.txt
-r--r--r-- 1 user group 0 Dec 15 10:30 file.txt

# Solution: Add write permission
$ chmod u+w file.txt
$ echo "test" > file.txt

# Problem: Cannot access directory
$ cd /restricted
bash: cd: /restricted: Permission denied

# Check directory permissions
$ ls -ld /restricted
drw-r--r-- 2 root root 4096 Dec 15 10:30 /restricted

# Solution: Need execute permission on directories
$ sudo chmod +x /restricted
```

Permission Debugging Tools

```
# Check effective permissions
$ namei -l /path/to/file
f: /path/to/file
 drwxr-xr-x root root /
 drwxr-xr-x root root path
 drwxr-xr-x user user to
 -rw-r--r-- user user file

# Test file access
$ test -r file.txt && echo "Readable" || echo "Not readable"
$ test -w file.txt && echo "Writable" || echo "Not writable"
$ test -x file.txt && echo "Executable" || echo "Not executable"

# Check user groups
$ groups
user sudo developers

$ id
uid=1000(user) gid=1000(user) groups=1000(user),27(sudo),1001(developers)

# Check file ownership
$ stat -c "%U %G" file.txt
user group
```

Permission Recovery

```
# Reset home directory permissions
$ sudo chmod 755 /home/user
$ sudo chmod -R u+rwX,go-w /home/user
$ sudo chmod 700 /home/user/.ssh
$ sudo chmod 600 /home/user/.ssh/*

# Reset web directory permissions
$ sudo find /var/www/html -type d -exec chmod 755 {} \;
$ sudo find /var/www/html -type f -exec chmod 644 {} \;
$ sudo chown -R www-data:www-data /var/www/html

# Emergency permission reset script
#!/bin/bash
# reset-permissions.sh
DIR="$1"
if [ -z "$DIR" ]; then
    echo "Usage: $0 <directory>"
    exit 1
fi

echo "Resetting permissions for $DIR"
find "$DIR" -type d -exec chmod 755 {} \;
find "$DIR" -type f -exec chmod 644 {} \;
echo "Done!"
```

🧠 Knowledge Check

Quick Quiz

1. **What does the permission string `-rwxr-xr--` mean?**

   ▶ Answer

   Regular file with owner having read/write/execute, group having read/execute, and others having read-only permissions.

2. **How do you make a script executable for everyone?**

   ▶ Answer

   ```
   chmod +x script.sh
   # or
   chmod 755 script.sh
   ```

3. **What's the difference between `chmod 755` and `chmod 644`?**

   ▶ Answer

   755 includes execute permission (rwxr-xr-x), while 644 doesn't (rw-r--r--). 755 is for executables/directories, 644 is for regular files.

4. **How do you find all setuid files on the system?**

   ▶ Answer

   ```
   find / -type f -perm -4000 2>/dev/null
   ```

Hands-On Challenges

Challenge 1: Secure Web Directory

```
# Create a web directory structure with proper permissions
# - Web root: 755
# - HTML files: 644
# - CGI scripts: 755
# - Config files: 600
# - Log directory: 755 with www-data ownership
```

Challenge 2: Shared Project Directory

```
# Set up a shared directory where:
# - Multiple users can read/write
# - New files inherit the group
# - Only file owners can delete their files
# - Use ACLs for fine-grained control
```

Challenge 3: Permission Audit

```
# Write a script that:
# - Finds world-writable files
# - Identifies setuid/setgid binaries
# - Checks for files with unusual permissions
# - Reports potential security issues
```

🚀 Next Steps

Excellent! You've mastered Linux file permissions and ownership. You can now:

- Understand and interpret permission strings
- Set appropriate permissions for different scenarios
- Manage file ownership effectively
- Use special permissions securely
- Implement ACLs for complex access control
- Troubleshoot permission-related issues

Ready for process management? Continue to 04-process-management.md to learn about controlling and monitoring system processes.

> **Pro Tip**: Always test permission changes in a safe environment first. Use `ls -l` frequently to verify permissions, and remember that directories need execute permission to be accessible. When in doubt, start with restrictive permissions and add access as needed! 🔒

⚙️ Process Management: Controlling System Resources

> **Master process control, monitoring, and system resource management in Linux**

📖 What You'll Learn

Process management is essential for system administration and troubleshooting. This chapter covers everything you need to know about controlling processes and system resources:

- Understanding processes and process hierarchy
- Monitoring running processes with `ps`, `top`, and `htop`
- Starting, stopping, and controlling processes
- Background and foreground job management
- Process signals and communication
- System resource monitoring
- Service management with systemd
- Performance optimization and troubleshooting

🌍 Why This Matters

Critical applications:

- **System Administration**: Monitor and control system resources
- **Troubleshooting**: Identify and resolve performance issues
- **Security**: Detect suspicious processes and resource abuse
- **Automation**: Manage services and background tasks
- **Performance**: Optimize system resource utilization

🔍 Understanding Processes

Process Basics

```
# Every running program is a process
# Each process has:
# - PID (Process ID): Unique identifier
# - PPID (Parent Process ID): Parent process
# - UID/GID: User and group ownership
# - State: Running, sleeping, stopped, zombie
# - Priority: CPU scheduling priority
# - Memory usage: RAM and virtual memory

# View your current shell process
$ echo $$
1234

# View parent process ID
$ echo $PPID
1000

# Process hierarchy example:
# systemd (PID 1)
# ├── sshd (PID 500)
# │   └── sshd (PID 1000) - your SSH session
# │       └── bash (PID 1234) - your shell
# │           └── ps (PID 1500) - command you're running
```

Process States

```
# Process states in Linux:
# R - Running or runnable
# S - Interruptible sleep (waiting for event)
# D - Uninterruptible sleep (usually I/O)
# T - Stopped (by signal or debugger)
# Z - Zombie (finished but not cleaned up)
# X - Dead (should never be seen)

# View process states
$ ps aux | head -5
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 225316  9876 ?        Ss   10:00   0:01 /sbin/init
root         2  0.0  0.0      0     0 ?        S    10:00   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        I<   10:00   0:00 [rcu_gp]
user      1234  0.0  0.2  21532  5432 pts/0    Ss   10:30   0:00 -bash
```

📊 Monitoring Processes

Using ps Command

```
# Basic process listing
$ ps
  PID TTY          TIME CMD
 1234 pts/0    00:00:00 bash
 1567 pts/0    00:00:00 ps

# Show all processes
$ ps aux
$ ps -ef
```

```
# Show process tree
$ ps auxf
$ ps -ef --forest
$ pstree

# Show processes for specific user
$ ps -u username
$ ps aux | grep username

# Show processes by command name
$ ps aux | grep nginx
$ pgrep nginx
$ pgrep -l nginx  # Show PID and name

# Show process hierarchy
$ ps -ejH
$ ps axjf

# Custom output format
$ ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu
  PID  PPID CMD                         %MEM %CPU
 1500  1234 firefox                     15.2 25.3
 1600  1234 chrome                      12.1 18.7
 1234     1 bash                         0.2  0.1
```

Real-time Monitoring with top

```
# Basic top usage
$ top

top - 10:30:45 up 2 days,  3:15,  2 users,  load average: 0.15, 0.25, 0.30
Tasks: 125 total,   1 running, 124 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.3 us,  1.2 sy,  0.0 ni, 96.2 id,  0.3 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   8192.0 total,   2048.5 free,   3072.2 used,   3071.3 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   4608.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1500 user      20   0 2097152 524288  65536 S  25.3  15.2   5:23.45 firefox
 1600 user      20   0 1572864 393216  49152 S  18.7  12.1   3:45.67 chrome
 1234 user      20   0   21532   5432   3456 S   0.1   0.2   0:00.12 bash

# Top interactive commands:
# q - quit
# k - kill process (enter PID)
# r - renice process (change priority)
# u - show processes for specific user
# M - sort by memory usage
# P - sort by CPU usage
# T - sort by running time
# 1 - show individual CPU cores
# h - help
# Top with specific options
$ top -u username     # Show processes for specific user
$ top -p 1234,5678    # Monitor specific PIDs
$ top -n 1            # Run once and exit
$ top -b -n 1         # Batch mode (good for scripts)
```

Enhanced Monitoring with `htop`

```
# Install htop (if not available)
$ sudo apt install htop  # Ubuntu/Debian
$ sudo yum install htop  # CentOS/RHEL

# Run htop
$ htop

# htop features:
# - Color-coded display
# - Mouse support
# - Tree view (F5)
# - Search (F3)
# - Filter (F4)
# - Kill process (F9)
# - Nice/renice (F7/F8)
# - Setup (F2)

# htop command line options
$ htop -u username    # Show processes for specific user
$ htop -p 1234,5678   # Monitor specific PIDs
$ htop -t             # Tree view
$ htop -s PERCENT_CPU # Sort by CPU usage
```

System Resource Monitoring

```
# Memory usage
$ free -h
              total        used        free      shared  buff/cache   available
Mem:          8.0Gi       3.0Gi       2.0Gi       256Mi       3.0Gi       4.6Gi
Swap:         2.0Gi          0B       2.0Gi

# Detailed memory information
$ cat /proc/meminfo | head -10
MemTotal:        8388608 kB
MemFree:         2097152 kB
MemAvailable:    4718592 kB
Buffers:          524288 kB
Cached:          2621440 kB

# CPU information
```

```
$ lscpu
Architecture:        x86_64
CPU op-mode(s):      32-bit, 64-bit
Byte Order:          Little Endian
CPU(s):              4
On-line CPU(s) list: 0-3
Thread(s) per core:  2
Core(s) per socket:  2
Socket(s):           1

# Load average
$ uptime
 10:30:45 up 2 days,  3:15,  2 users,  load average: 0.15, 0.25, 0.30

$ cat /proc/loadavg
0.15 0.25 0.30 2/125 1567

# I/O statistics
$ iostat 1 3  # 1-second intervals, 3 times
Linux 5.4.0 (hostname)     12/15/2023      _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           2.50    0.00    1.25    0.25    0.00   96.00

Device             tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
sda               5.25       125.50        75.25     125500      75250
```

🎮 Process Control

Starting and Stopping Processes

```
# Start process in foreground
$ long_running_command

# Start process in background
$ long_running_command &
[1] 1234

# Start with nohup (survives logout)
$ nohup long_running_command &
[1] 1234
nohup: ignoring input and appending output to 'nohup.out'

# Start with custom output redirection
$ nohup long_running_command > output.log 2>&1 &

# Disown process (remove from job control)
$ long_running_command &
$ disown %1

# Start process with specific priority
```

```
$ nice -n 10 cpu_intensive_task  # Lower priority
$ nice -n -5 important_task      # Higher priority (requires sudo)
```

Job Control

```
# List active jobs
$ jobs
[1]+  Running                 long_running_command &
[2]-  Stopped                 vim file.txt

# Bring job to foreground
$ fg %1        # Bring job 1 to foreground
$ fg           # Bring most recent job to foreground

# Send job to background
$ bg %1        # Send job 1 to background
$ bg           # Send most recent job to background

# Suspend current foreground process
# Press Ctrl+Z
$ vim file.txt
^Z
[1]+  Stopped                 vim file.txt

# Resume suspended job in background
$ bg %1
[1]+ vim file.txt &

# Kill job
$ kill %1      # Kill job 1
$ kill %2      # Kill job 2
```

Process Signals

```
# Common signals:
# SIGTERM (15) - Graceful termination (default)
# SIGKILL (9)  - Force kill (cannot be caught)
# SIGSTOP (19) - Stop process (cannot be caught)
# SIGCONT (18) - Continue stopped process
# SIGHUP (1)   - Hangup (often used to reload config)
# SIGINT (2)   - Interrupt (Ctrl+C)
# SIGQUIT (3)  - Quit (Ctrl+\)

# Send signals to processes
$ kill 1234           # Send SIGTERM to PID 1234
$ kill -9 1234        # Send SIGKILL to PID 1234
$ kill -TERM 1234     # Send SIGTERM (same as default)
$ kill -HUP 1234      # Send SIGHUP (reload config)
$ kill -STOP 1234     # Stop process
$ kill -CONT 1234     # Continue process

# Kill processes by name
$ killall firefox     # Kill all firefox processes
$ killall -9 firefox  # Force kill all firefox processes
$ pkill firefox       # Kill processes matching pattern
$ pkill -f "python script.py"  # Kill by full command line

# Kill processes by user
$ pkill -u username   # Kill all processes by user
$ sudo pkill -u username  # Kill as root

# Interactive process killing
$ top                 # Press 'k' and enter PID
$ htop                # Press F9 and select signal
```

Process Priority Management

```
# View process priorities
$ ps -eo pid,ni,pri,pcpu,comm
  PID  NI PRI %CPU COMMAND
 1234   0  20  0.1 bash
 1500  10  30  5.2 backup_script
 1600 -10  10 15.3 important_app

# Change priority of running process
$ sudo renice 10 1234      # Lower priority (higher nice value)
$ sudo renice -5 1234      # Higher priority (lower nice value)
$ sudo renice 0 1234       # Normal priority

# Change priority by process name
$ sudo renice 10 $(pgrep firefox)

# Start process with specific priority
$ nice -n 19 backup_script.sh    # Lowest priority
$ nice -n 0 normal_script.sh     # Normal priority
$ sudo nice -n -20 critical_app  # Highest priority

# Real-time priority (use with caution)
$ sudo chrt -f 50 critical_realtime_app  # FIFO scheduling
$ sudo chrt -r 50 critical_realtime_app  # Round-robin scheduling
```

🔧 Service Management with systemd

Basic Service Operations

```
# Check service status
$ sudo systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
```

```
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset:
enabled)
   Active: active (running) since Mon 2023-12-15 10:00:00 UTC; 2h 30min ago
     Docs: man:nginx(8)
  Process: 1234 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process
on; (code=exited, status=0/SUCCESS)
  Process: 1235 ExecStart=/usr/sbin/nginx -g daemon on; master_process on;
(code=exited, status=0/SUCCESS)
 Main PID: 1236 (nginx)
    Tasks: 5 (limit: 4915)
   Memory: 15.2M
   CGroup: /system.slice/nginx.service
           ├─1236 nginx: master process /usr/sbin/nginx -g daemon on;
master_process on;
           ├─1237 nginx: worker process
           └─1238 nginx: worker process

# Start/stop/restart services
$ sudo systemctl start nginx
$ sudo systemctl stop nginx
$ sudo systemctl restart nginx
$ sudo systemctl reload nginx    # Reload config without restart

# Enable/disable services (auto-start at boot)
$ sudo systemctl enable nginx
$ sudo systemctl disable nginx

# Check if service is enabled
$ systemctl is-enabled nginx
enabled

# Check if service is active
$ systemctl is-active nginx
active
```

Service Management Commands

```
# List all services
$ systemctl list-units --type=service
$ systemctl list-units --type=service --state=running
$ systemctl list-units --type=service --state=failed

# List all unit files
$ systemctl list-unit-files --type=service

# Show service dependencies
$ systemctl list-dependencies nginx

# View service logs
$ sudo journalctl -u nginx
$ sudo journalctl -u nginx -f        # Follow logs
$ sudo journalctl -u nginx --since today
$ sudo journalctl -u nginx --since "2023-12-15 10:00:00"

# Mask/unmask services (prevent starting)
$ sudo systemctl mask nginx
$ sudo systemctl unmask nginx

# Edit service configuration
$ sudo systemctl edit nginx          # Create override file
$ sudo systemctl edit --full nginx   # Edit full unit file

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

Creating Custom Services

```
# Create a simple service file
$ sudo nano /etc/systemd/system/myapp.service

[Unit]
Description=My Application
After=network.target

[Service]
Type=simple
User=myuser
WorkingDirectory=/opt/myapp
ExecStart=/opt/myapp/start.sh
ExecReload=/bin/kill -HUP $MAINPID
Restart=always
RestartSec=10

[Install]
WantedBy=multi-user.target

# Enable and start the service
$ sudo systemctl daemon-reload
$ sudo systemctl enable myapp
$ sudo systemctl start myapp
$ sudo systemctl status myapp

# Service types:
# simple - Default, process doesn't fork
# forking - Process forks and parent exits
# oneshot - Process exits after completion
# notify - Process sends notification when ready
# idle - Waits for other jobs to complete
```

📈 Performance Monitoring and Optimization

System Performance Analysis

```
# CPU usage over time
$ sar -u 1 10         # CPU usage every 1 second, 10 times
Linux 5.4.0 (hostname)     12/15/2023      _x86_64_        (4 CPU)

10:30:01 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
10:30:02 AM     all      2.50      0.00      1.25      0.25      0.00     96.00
10:30:03 AM     all      3.75      0.00      1.50      0.50      0.00     94.25

# Memory usage over time
$ sar -r 1 5          # Memory usage every 1 second, 5 times

# I/O statistics
$ sar -b 1 5          # I/O statistics
$ sar -d 1 5          # Disk statistics

# Network statistics
$ sar -n DEV 1 5      # Network device statistics

# Generate system activity report
$ sar -A > system_report.txt
```

Process Resource Usage

```
# Detailed process information
$ cat /proc/1234/status
Name:   firefox
State:  S (sleeping)
Pid:    1234
PPid:   1000
VmPeak:  2097152 kB
VmSize:  1572864 kB
VmRSS:    524288 kB

# Process memory maps
$ cat /proc/1234/maps | head -5
7f8b4c000000-7f8b4c021000 rw-p 00000000 00:00 0
7f8b4c021000-7f8b50000000 ---p 00000000 00:00 0
7f8b50000000-7f8b50021000 rw-p 00000000 00:00 0

# Process file descriptors
$ ls -l /proc/1234/fd/
total 0
lrwx------ 1 user user 64 Dec 15 10:30 0 -> /dev/pts/0
lrwx------ 1 user user 64 Dec 15 10:30 1 -> /dev/pts/0
lrwx------ 1 user user 64 Dec 15 10:30 2 -> /dev/pts/0

# Process environment
$ cat /proc/1234/environ | tr '\0' '\n' | head -5
```

```
PATH=/usr/local/bin:/usr/bin:/bin
HOME=/home/user
USER=user
SHELL=/bin/bash
TERM=xterm-256color

# Process command line
$ cat /proc/1234/cmdline
firefox--no-remote--profile
```

Resource Monitoring Scripts

```
#!/bin/bash
# system-monitor.sh - Comprehensive system monitoring

echo "=== System Performance Monitor ==="
echo "Date: $(date)"
echo

# CPU usage
echo "CPU Usage:"
top -bn1 | grep "Cpu(s)" | awk '{print "User: " $2 ", System: " $4 ", Idle: " $8}'
echo

# Memory usage
echo "Memory Usage:"
free -h | grep -E "Mem|Swap"
echo

# Load average
echo "Load Average:"
uptime | awk -F'load average:' '{print $2}'
echo

# Top 5 CPU consumers
echo "Top 5 CPU Consumers:"
ps aux --sort=-%cpu | head -6 | awk '{print $11 " (" $3 "%)"}'  | tail -5
echo

# Top 5 Memory consumers
echo "Top 5 Memory Consumers:"
ps aux --sort=-%mem | head -6 | awk '{print $11 " (" $4 "%)"}'  | tail -5
echo

# Disk usage
echo "Disk Usage:"
df -h | grep -E "^/dev"
echo

echo "=== Monitor Complete ==="
```

Performance Optimization Tips

```
# Find resource-intensive processes
$ ps aux --sort=-%cpu | head -10    # Top CPU users
$ ps aux --sort=-%mem | head -10    # Top memory users

# Find processes with many open files
$ lsof | awk '{print $2}' | sort | uniq -c | sort -nr | head -10

# Monitor process creation
$ sudo sysctl kernel.fork_rate
$ watch -n 1 'ps aux | wc -l'

# Check for zombie processes
$ ps aux | grep -E "<defunct>|Z"

# Monitor system calls
$ strace -p 1234          # Trace system calls for PID 1234
$ strace -c command       # Count system calls

# Profile CPU usage
$ perf top                # Real-time CPU profiling
$ perf record command     # Record performance data
$ perf report             # Analyze recorded data
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between `kill` and `killall`?**

   ▶ Answer

   `kill` terminates a specific process by PID, while `killall` terminates all processes with a specific name.

2. **How do you start a process that survives logout?**

   ▶ Answer

   ```
   nohup command &
   # or
   command &
   disown
   ```

3. **What does a load average of 2.0 mean on a 4-core system?**

   ▶ Answer

The system is 50% utilized. Load average represents the number of processes waiting for CPU time. 2.0 on a 4-core system means 2 cores are fully utilized on average.

4. **How do you change the priority of a running process?**

▶ Answer

```
sudo renice 10 PID    # Lower priority
sudo renice -5 PID    # Higher priority
```

Hands-On Challenges

Challenge 1: Process Monitoring Dashboard

```
# Create a script that displays:
# - Top 5 CPU-consuming processes
# - Top 5 memory-consuming processes
# - Current load average
# - Available memory
# - Number of running processes
```

Challenge 2: Service Management

```
# Create a custom systemd service for a simple application
# Configure it to:
# - Start automatically at boot
# - Restart on failure
# - Log to a specific file
# - Run as a non-root user
```

Challenge 3: Performance Troubleshooting

```
# Simulate a high CPU load and:
# - Identify the problematic process
# - Reduce its priority
# - Monitor the impact
# - Create a script to automatically detect and handle such situations
```

🚀 Next Steps

Excellent! You've mastered Linux process management. You can now:

- Monitor system processes and resource usage
- Control process execution and priority

- Manage background jobs effectively
- Work with systemd services
- Troubleshoot performance issues
- Optimize system resource utilization

Ready for networking fundamentals? Continue to 05-networking-basics.md to learn about network concepts and protocols.

> **Pro Tip**: Use `htop` for interactive process management, `systemctl` for service control, and always check system load before making changes. Remember that killing processes with `-9` should be a last resort - try graceful termination first! ⚙️

🌐 Networking Basics: Core Concepts

> **Understand IP addresses, subnets, protocols, and how devices communicate**

📖 What You'll Learn

Networking is the foundation of modern computing. Whether you're troubleshooting connectivity issues, setting up servers, or securing systems, understanding networking fundamentals is essential. This chapter covers:

- IP addressing and subnetting (IPv4 and IPv6)
- MAC addresses and how they work
- Network protocols and the OSI model
- CIDR notation and subnet calculations
- Private vs public IP addresses
- How devices find and communicate with each other

🌍 Why This Matters

Real-world applications:

- **Web Development**: Understanding how browsers connect to servers
- **System Administration**: Configuring network interfaces and routing
- **DevOps**: Setting up cloud infrastructure and container networking
- **Cybersecurity**: Analyzing network traffic and implementing security
- **Troubleshooting**: Diagnosing connectivity and performance issues

🏘️ Network Fundamentals

What is a Network?

A network is a collection of devices that can communicate with each other. Think of it like a postal system:

```
[Your Computer] ↔ [Router] ↔ [Internet] ↔ [Web Server]
       ↓                ↓          ↓           ↓
```

```
    Your House    Post Office   Highway    Destination
```

Network Components

Component	Purpose	Example
Host	End device that sends/receives data	Computer, phone, server
Switch	Connects devices in local network	Office network switch
Router	Connects different networks	Home router, ISP router
Gateway	Entry/exit point between networks	Default gateway
Firewall	Controls network traffic	Hardware/software firewall

📍 IP Addresses: The Internet's Postal System

IPv4 Addresses

IPv4 addresses are 32-bit numbers written as four octets (0-255):

```
# Example IPv4 address
192.168.1.100
# ^^^.^^^.^.^^^
#  |   |  | |
#  |   |  | +-- Host part (100)
#  |   |  +---- Network part
#  |   +------- Network part
#  +----------- Network part (192.168.1)
```

IPv4 Address Classes (Historical)

Class	Range	Default Subnet	Typical Use
A	1.0.0.0 - 126.255.255.255	/8	Large organizations
B	128.0.0.0 - 191.255.255.255	/16	Medium organizations
C	192.0.0.0 - 223.255.255.255	/24	Small networks
D	224.0.0.0 - 239.255.255.255	-	Multicast
E	240.0.0.0 - 255.255.255.255	-	Reserved

Private vs Public IP Addresses

Private IP Ranges (RFC 1918):

```
# These are NOT routed on the internet
10.0.0.0/8         # 10.0.0.0 - 10.255.255.255
172.16.0.0/12      # 172.16.0.0 - 172.31.255.255
192.168.0.0/16     # 192.168.0.0 - 192.168.255.255

# Special addresses
127.0.0.0/8        # Loopback (localhost)
169.254.0.0/16     # Link-local (APIPA)
```

Public IP Addresses:

- Globally unique and routable on the internet
- Assigned by Internet Service Providers (ISPs)
- Examples: 8.8.8.8 (Google DNS), 1.1.1.1 (Cloudflare DNS)

Checking Your IP Addresses

```
# Check private IP address
$ ip addr show
# or
$ ifconfig

# Check public IP address
$ curl ifconfig.me
203.0.113.45

# or
$ curl ipinfo.io/ip
203.0.113.45

# Detailed public IP info
$ curl ipinfo.io
{
  "ip": "203.0.113.45",
  "city": "New York",
  "region": "New York",
  "country": "US",
  "org": "AS12345 Example ISP"
}
```

🔢 Subnetting and CIDR Notation

Understanding Subnet Masks

Subnet masks determine which part of an IP address is the network and which part is the host:

```
# IP Address:    192.168.1.100
# Subnet Mask:   255.255.255.0
# Network:       192.168.1.0
# Host:          100
# Broadcast:     192.168.1.255
```

CIDR Notation

CIDR (Classless Inter-Domain Routing) uses a slash followed by the number of network bits:

```
# Common CIDR notations
192.168.1.0/24    # 255.255.255.0   - 254 hosts
192.168.1.0/25    # 255.255.255.128 - 126 hosts
192.168.1.0/26    # 255.255.255.192 - 62 hosts
192.168.1.0/27    # 255.255.255.224 - 30 hosts
192.168.1.0/28    # 255.255.255.240 - 14 hosts
```

Subnet Calculation Examples

Example 1: /24 Network

```
Network: 192.168.1.0/24
Subnet Mask: 255.255.255.0
Network Address: 192.168.1.0
First Host: 192.168.1.1
Last Host: 192.168.1.254
Broadcast: 192.168.1.255
Total Hosts: 254
```

Example 2: /26 Network

```
Network: 192.168.1.0/26
Subnet Mask: 255.255.255.192
Network Address: 192.168.1.0
First Host: 192.168.1.1
Last Host: 192.168.1.62
Broadcast: 192.168.1.63
Total Hosts: 62
```

Subnet Calculation Tools

```
# Using ipcalc (install if needed)
$ sudo apt install ipcalc
$ ipcalc 192.168.1.0/24
Address:   192.168.1.0          11000000.10101000.00000001. 00000000
Netmask:   255.255.255.0 = 24   11111111.11111111.11111111. 00000000
```

```
Wildcard:  0.0.0.255            00000000.00000000.00000000. 11111111
=>
Network:   192.168.1.0/24       11000000.10101000.00000001. 00000000
HostMin:   192.168.1.1          11000000.10101000.00000001. 00000001
HostMax:   192.168.1.254        11000000.10101000.00000001. 11111110
Broadcast: 192.168.1.255        11000000.10101000.00000001. 11111111
Hosts/Net: 254                   Class C, Private Internet

# Using sipcalc
$ sudo apt install sipcalc
$ sipcalc 192.168.1.0/24
```

🆔 MAC Addresses: Hardware Identifiers

What is a MAC Address?

MAC (Media Access Control) addresses are unique 48-bit identifiers assigned to network interfaces:

```
# MAC address format
00:1B:44:11:3A:B7
# ^^:^^:^^:^^:^^:^^
# |     |     |
# |     |     +-- Device-specific (NIC)
# |     +-------- Manufacturer-specific
# +-------------- Organizationally Unique Identifier (OUI)
```

Viewing MAC Addresses

```
# Linux - show all interfaces
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
    link/ether 00:1b:44:11:3a:b7 brd ff:ff:ff:ff:ff:ff

# Show specific interface
$ cat /sys/class/net/eth0/address
00:1b:44:11:3a:b7

# Using ifconfig
$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:1b:44:11:3a:b7  txqueuelen 1000  (Ethernet)
```

MAC Address vs IP Address

Feature	MAC Address	IP Address
Scope	Local network only	Global (with routing)
Layer	Data Link (Layer 2)	Network (Layer 3)
Changes	Fixed to hardware	Can change
Format	6 hex octets	4 decimal octets (IPv4)
Purpose	Local delivery	End-to-end delivery

🌐 IPv6: The Future of Internet Addressing

IPv6 Address Format

IPv6 addresses are 128-bit numbers written in hexadecimal:

```
# Full IPv6 address
2001:0db8:85a3:0000:0000:8a2e:0370:7334

# Compressed format (remove leading zeros)
2001:db8:85a3:0:0:8a2e:370:7334

# Further compressed (:: replaces consecutive zeros)
2001:db8:85a3::8a2e:370:7334
```

IPv6 Address Types

```
# Loopback
::1                    # Equivalent to 127.0.0.1

# Link-local (auto-configured)
fe80::/10              # fe80:0000:0000:0000::/64

# Unique local (private)
fc00::/7               # Similar to RFC 1918

# Global unicast (public)
2000::/3               # Routable on internet

# Multicast
ff00::/8               # Group communication
```

Viewing IPv6 Addresses

```
# Show IPv6 addresses
$ ip -6 addr show
```

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 state UNKNOWN
    inet6 ::1/128 scope host
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 state UP
    inet6 2001:db8:85a3::8a2e:370:7334/64 scope global
    inet6 fe80::21b:44ff:fe11:3ab7/64 scope link

# Check IPv6 connectivity
$ ping6 google.com
$ ping6 2001:4860:4860::8888  # Google's IPv6 DNS
```

🚦 Network Protocols and the OSI Model

The OSI Model

Layer	Name	Purpose	Examples
7	Application	User interface	HTTP, FTP, SSH, DNS
6	Presentation	Data formatting	SSL/TLS, compression
5	Session	Connection management	NetBIOS, RPC
4	Transport	End-to-end delivery	TCP, UDP
3	Network	Routing	IP, ICMP, OSPF
2	Data Link	Local delivery	Ethernet, WiFi
1	Physical	Electrical signals	Cables, radio waves

TCP vs UDP

TCP (Transmission Control Protocol):

```
# Characteristics:
# ✅ Reliable (guaranteed delivery)
# ✅ Connection-oriented
# ✅ Error checking and correction
# ✅ Flow control
# ❌ Higher overhead

# Common TCP ports:
22   # SSH
23   # Telnet
25   # SMTP
53   # DNS (also UDP)
80   # HTTP
110  # POP3
143  # IMAP
443  # HTTPS
993  # IMAPS
995  # POP3S
```

UDP (User Datagram Protocol):

```
# Characteristics:
# ✅ Fast (low overhead)
# ✅ Connectionless
# ❌ No guaranteed delivery
# ❌ No error correction
# ❌ No flow control

# Common UDP ports:
53   # DNS
67   # DHCP Server
68   # DHCP Client
69   # TFTP
123  # NTP
161  # SNMP
514  # Syslog
```

Common Network Protocols

Application Layer Protocols:

```
# HTTP/HTTPS - Web traffic
$ curl -I https://google.com
HTTP/2 200
date: Fri, 15 Dec 2023 10:30:00 GMT
server: gws

# DNS - Name resolution
$ nslookup google.com
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.191.14

# SSH - Secure remote access
$ ssh user@192.168.1.100

# FTP - File transfer
$ ftp ftp.example.com
```

Network Layer Protocols:

```
# ICMP - Internet Control Message Protocol
$ ping google.com
```

```
PING google.com (142.250.191.14) 56(84) bytes of data.
64 bytes from lga25s62-in-f14.1e100.net (142.250.191.14): icmp_seq=1 ttl=117
time=12.3 ms

# ARP - Address Resolution Protocol
$ arp -a
? (192.168.1.1) at 00:1a:2b:3c:4d:5e [ether] on eth0
? (192.168.1.100) at 00:1b:44:11:3a:b7 [ether] on eth0
```

🏠 How Home Networks Work

Typical Home Network Setup

```
Internet (ISP)
      |
   [Modem]  ← Converts ISP signal to Ethernet
      |
   [Router] ← NAT, DHCP, Firewall, WiFi
      |
   [Switch] ← Additional wired ports (optional)
   /  |  \
 [PC] [Laptop] [Phone]
```

Network Address Translation (NAT)

NAT allows multiple devices to share one public IP address:

```
# Without NAT (not possible - not enough public IPs)
Phone:  203.0.113.45
Laptop: 203.0.113.46  ← Would need separate public IPs
PC:     203.0.113.47

# With NAT (how it actually works)
Public IP:  203.0.113.45
           ↓ (NAT Router)
Phone:     192.168.1.10:5000 → 203.0.113.45:5000
Laptop:    192.168.1.11:5001 → 203.0.113.45:5001
PC:        192.168.1.12:5002 → 203.0.113.45:5002
```

DHCP (Dynamic Host Configuration Protocol)

DHCP automatically assigns IP addresses to devices:

```
# DHCP process:
1. Device: "I need an IP address" (DHCP Discover)
2. Router: "Here are available options" (DHCP Offer)
3. Device: "I'll take 192.168.1.100" (DHCP Request)
```

```
4. Router: "Confirmed, here's your config" (DHCP Acknowledge)

# DHCP provides:
# - IP address (192.168.1.100)
# - Subnet mask (255.255.255.0)
# - Default gateway (192.168.1.1)
# - DNS servers (8.8.8.8, 8.8.4.4)
# - Lease time (24 hours)
```

Viewing Network Configuration

```
# Show IP configuration
$ ip addr show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:1b:44:11:3a:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 86395sec preferred_lft 86395sec

# Show routing table
$ ip route show
default via 192.168.1.1 dev eth0 proto dhcp metric 100
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.100 metric 100

# Show DNS configuration
$ cat /etc/resolv.conf
nameserver 8.8.8.8
nameserver 8.8.4.4
search local.domain
```

🎯 Practical Scenario: Network Troubleshooting

Let's diagnose a connectivity issue step by step:

```
# Problem: "I can't access google.com"

# Step 1: Check local network interface
$ ip addr show
# Look for: IP address assigned, interface UP

# Step 2: Check local connectivity
$ ping 127.0.0.1
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.045 ms
# ✅ Local interface works

# Step 3: Check default gateway
$ ip route | grep default
default via 192.168.1.1 dev eth0
```

```
$ ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) 56(84) bytes of data.
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=1.23 ms
# ✅ Can reach router

# Step 4: Check DNS resolution
$ nslookup google.com
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.191.14
# ✅ DNS works

# Step 5: Check internet connectivity
$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=12.3 ms
# ✅ Internet connectivity works

# Step 6: Check specific service
$ curl -I https://google.com
HTTP/2 200
# ✅ Web service works

# Conclusion: Network is working properly
# Issue might be browser-specific or application-level
```

⚠ Common Networking Mistakes

1. Subnet Misconfiguration

```
# ❌ Wrong: Device and gateway in different subnets
Device:  192.168.1.100/25  (subnet: 192.168.1.0-127)
Gateway: 192.168.1.200     (subnet: 192.168.1.128-255)

# ✅ Correct: Same subnet
Device:  192.168.1.100/24
Gateway: 192.168.1.1
```

2. IP Address Conflicts

```
# ❌ Two devices with same IP
Device A: 192.168.1.100
Device B: 192.168.1.100  ← Conflict!
```

```
# Check for conflicts
$ arping 192.168.1.100
```

3. DNS Issues

```
# ❌ Wrong DNS servers
nameserver 192.168.1.999  ← Invalid IP

# ✅ Use reliable DNS
nameserver 8.8.8.8        ← Google
nameserver 1.1.1.1        ← Cloudflare
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between 192.168.1.0/24 and 192.168.1.0/25?**

   ▶ Answer
   - /24 has 254 usable hosts (192.168.1.1-254)
   - /25 has 126 usable hosts (192.168.1.1-126)
   - /25 splits the /24 network into two smaller subnets

2. **Why can't you ping a device on the internet using its MAC address?**

   ▶ Answer

   MAC addresses only work on the local network segment. Routers strip and replace MAC addresses as packets travel between networks. Only IP addresses are used for end-to-end communication.

3. **What happens when you type "google.com" in your browser?**

   ▶ Answer
   1. DNS lookup to resolve google.com to IP address
   2. TCP connection to port 80/443
   3. HTTP/HTTPS request sent
   4. Server responds with web page
   5. Browser renders the content

4. **What's the broadcast address for 10.0.0.0/8?**

   ▶ Answer

   10.255.255.255

Hands-On Challenges

Challenge 1: Subnet Planning

```
# You have 192.168.1.0/24 and need 4 subnets with ~60 hosts each
# Calculate the subnet addresses, ranges, and broadcast addresses
```

Challenge 2: Network Discovery

```
# Find all devices on your local network
# Identify their IP addresses, MAC addresses, and hostnames
```

Challenge 3: Protocol Analysis

```
# Use netstat to identify:
# - All listening TCP services
# - All active connections
# - Which processes are using the network
```

🚀 Next Steps

Fantastic! You now understand networking fundamentals. You can:

- Calculate subnets and understand IP addressing
- Distinguish between different types of addresses and protocols
- Understand how devices communicate on networks
- Troubleshoot basic connectivity issues

Ready to get hands-on with tools? Continue to 09-basic-network-tools.md to master essential networking commands and diagnostic tools.

> **Pro Tip**: Networking concepts build on each other. Make sure you understand IP addressing and subnetting before moving to advanced topics. Practice subnet calculations until they become second nature - this knowledge is fundamental for network administration and troubleshooting! 🌐

🛠️ System Administration: Managing Linux Systems

> **Master essential system administration tasks for Linux servers and workstations**

📖 What You'll Learn

System administration is the backbone of maintaining reliable Linux systems. This chapter covers essential administrative tasks that every Linux professional should master:

- User and group management
- Package management across distributions
- System monitoring and logging

- Disk and filesystem management
- Network configuration and management
- System backup and recovery
- Security hardening basics
- Automation with cron and scripts

🌍 Why This Matters

Critical applications:

- **Server Management**: Maintain production servers and services
- **Security**: Implement proper access controls and monitoring
- **Performance**: Optimize system resources and troubleshoot issues
- **Reliability**: Ensure system uptime and data protection
- **Compliance**: Meet organizational and regulatory requirements

🧑‍🤝‍🧑 User and Group Management

User Account Management

```
# Create new user
$ sudo useradd -m -s /bin/bash john
$ sudo useradd -m -s /bin/bash -G sudo,developers john  # With groups

# Set user password
$ sudo passwd john
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully

# Create user with specific UID and home directory
$ sudo useradd -m -u 1500 -d /home/custom -s /bin/bash jane

# Modify existing user
$ sudo usermod -aG sudo john        # Add to sudo group
$ sudo usermod -s /bin/zsh john     # Change shell
$ sudo usermod -d /new/home john    # Change home directory
$ sudo usermod -l newname john      # Change username

# Lock/unlock user account
$ sudo usermod -L john              # Lock account
$ sudo usermod -U john              # Unlock account
$ sudo passwd -l john               # Lock password
$ sudo passwd -u john               # Unlock password

# Delete user
$ sudo userdel john                 # Delete user (keep home)
$ sudo userdel -r john              # Delete user and home directory

# View user information
$ id john
```

```
uid=1001(john) gid=1001(john) groups=1001(john),27(sudo),1002(developers)

$ finger john
Login: john                             Name: John Doe
Directory: /home/john                   Shell: /bin/bash
Last login Mon Dec 15 10:30 (UTC) on pts/0
No mail.
No Plan.

# List all users
$ cat /etc/passwd | cut -d: -f1
$ getent passwd | cut -d: -f1
```

Group Management

```
# Create new group
$ sudo groupadd developers
$ sudo groupadd -g 2000 admins     # With specific GID

# Add user to group
$ sudo usermod -aG developers john
$ sudo gpasswd -a john developers  # Alternative method

# Remove user from group
$ sudo gpasswd -d john developers

# Change user's primary group
$ sudo usermod -g developers john

# Delete group
$ sudo groupdel developers

# List all groups
$ cat /etc/group | cut -d: -f1
$ getent group | cut -d: -f1

# Show groups for user
$ groups john
john : john sudo developers

# Show group members
$ getent group sudo
sudo:x:27:john,jane,admin
```

Advanced User Management

```
# Set password policies
$ sudo nano /etc/login.defs
# PASS_MAX_DAYS   90
# PASS_MIN_DAYS   1
# PASS_WARN_AGE   7
# PASS_MIN_LEN    8

# Force password change on next login
$ sudo chage -d 0 john

# Set password expiration
$ sudo chage -M 90 john             # Max 90 days
$ sudo chage -m 1 john              # Min 1 day between changes
$ sudo chage -W 7 john              # Warn 7 days before expiry

# View password aging information
$ sudo chage -l john
Last password change                                    : Dec 15, 2023
Password expires                                        : Mar 14, 2024
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change         : 1
Maximum number of days between password change         : 90
Number of days of warning before password expires      : 7

# Create system user (no login)
$ sudo useradd -r -s /bin/false -d /var/lib/myapp myapp

# Bulk user creation
$ sudo newusers users.txt
# users.txt format:
# username:password:uid:gid:comment:home:shell
```

📦 Package Management

Debian/Ubuntu (APT)

```
# Update package lists
$ sudo apt update

# Upgrade packages
$ sudo apt upgrade
$ sudo apt full-upgrade            # More comprehensive upgrade

# Install packages
$ sudo apt install nginx
$ sudo apt install nginx mysql-server php  # Multiple packages

# Remove packages
$ sudo apt remove nginx
$ sudo apt purge nginx             # Remove with config files
$ sudo apt autoremove              # Remove unused dependencies

# Search packages
```

```
$ apt search nginx
$ apt-cache search web server

# Show package information
$ apt show nginx
$ apt-cache show nginx

# List installed packages
$ apt list --installed
$ dpkg -l

# Check if package is installed
$ dpkg -l | grep nginx
$ apt list --installed | grep nginx

# Download package without installing
$ apt download nginx

# Install local .deb package
$ sudo dpkg -i package.deb
$ sudo apt install -f             # Fix dependencies

# Hold package (prevent updates)
$ sudo apt-mark hold nginx
$ sudo apt-mark unhold nginx

# Clean package cache
$ sudo apt clean
$ sudo apt autoclean
```

Red Hat/CentOS (YUM/DNF)

```
# Update package lists
$ sudo yum update                  # CentOS 7
$ sudo dnf update                  # CentOS 8+

# Install packages
$ sudo yum install nginx
$ sudo dnf install nginx mysql-server

# Remove packages
$ sudo yum remove nginx
$ sudo dnf remove nginx

# Search packages
$ yum search nginx
$ dnf search nginx

# Show package information
$ yum info nginx
$ dnf info nginx
```

```
# List installed packages
$ yum list installed
$ dnf list installed
$ rpm -qa

# Install local .rpm package
$ sudo rpm -ivh package.rpm
$ sudo yum localinstall package.rpm
$ sudo dnf localinstall package.rpm

# Enable/disable repositories
$ sudo yum-config-manager --enable epel
$ sudo dnf config-manager --enable epel

# Clean package cache
$ sudo yum clean all
$ sudo dnf clean all
```

Package Management Best Practices

```
# Always update before installing
$ sudo apt update && sudo apt install package

# Check for security updates
$ sudo apt list --upgradable
$ sudo unattended-upgrade --dry-run  # Ubuntu

# Backup package list
$ dpkg --get-selections > package-list.txt
$ sudo dpkg --set-selections < package-list.txt
$ sudo apt-get dselect-upgrade

# Find which package provides a file
$ dpkg -S /usr/bin/nginx
$ yum whatprovides /usr/bin/nginx
$ dnf whatprovides /usr/bin/nginx

# List files in package
$ dpkg -L nginx
$ rpm -ql nginx

# Verify package integrity
$ debsums nginx                     # Debian/Ubuntu
$ rpm -V nginx                      # Red Hat/CentOS
```

📊 System Monitoring and Logging

System Logs

```
# View system logs with journalctl (systemd)
$ sudo journalctl
$ sudo journalctl -f               # Follow logs
$ sudo journalctl -u nginx         # Service-specific logs
$ sudo journalctl --since today
$ sudo journalctl --since "2023-12-15 10:00:00"
$ sudo journalctl --until "2023-12-15 12:00:00"
$ sudo journalctl -p err           # Error level and above
$ sudo journalctl -n 50            # Last 50 entries

# Traditional log files
$ sudo tail -f /var/log/syslog     # System log
$ sudo tail -f /var/log/auth.log   # Authentication log
$ sudo tail -f /var/log/kern.log   # Kernel log
$ sudo tail -f /var/log/mail.log   # Mail log
$ sudo tail -f /var/log/apache2/access.log  # Web server logs

# Log rotation
$ sudo logrotate -d /etc/logrotate.conf  # Dry run
$ sudo logrotate -f /etc/logrotate.conf  # Force rotation

# Configure log rotation
$ sudo nano /etc/logrotate.d/myapp
/var/log/myapp/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 644 myapp myapp
    postrotate
        systemctl reload myapp
    endscript
}
```

System Resource Monitoring

```
# Disk usage monitoring
$ df -h                            # Filesystem usage
$ du -sh /var/log/*                # Directory sizes
$ du -h --max-depth=1 /            # Top-level directory sizes

# Find large files
$ find / -type f -size +100M 2>/dev/null | head -10
$ find /var/log -type f -size +10M -exec ls -lh {} \;

# Monitor disk I/O
$ iostat -x 1                      # Extended I/O statistics
$ iotop                            # Top-like I/O monitor
```

```
# Network monitoring
$ netstat -tuln                    # Listening ports
$ ss -tuln                         # Modern alternative
$ iftop                            # Network bandwidth usage
$ nethogs                          # Per-process network usage

# System performance
$ vmstat 1                         # Virtual memory statistics
$ sar -u 1 10                      # CPU usage
$ sar -r 1 10                      # Memory usage
$ sar -d 1 10                      # Disk activity
```

Automated Monitoring Scripts

```
#!/bin/bash
# system-health-check.sh - Automated system health monitoring

LOGFILE="/var/log/system-health.log"
DATE=$(date '+%Y-%m-%d %H:%M:%S')

echo "[$DATE] Starting system health check" >> $LOGFILE

# Check disk usage
DISK_USAGE=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')
if [ $DISK_USAGE -gt 90 ]; then
    echo "[$DATE] WARNING: Root filesystem is ${DISK_USAGE}% full" >> $LOGFILE
    # Send alert email
    echo "Root filesystem is ${DISK_USAGE}% full" | mail -s "Disk Space Alert"
admin@example.com
fi

# Check memory usage
MEM_USAGE=$(free | awk 'NR==2{printf "%.0f", $3*100/$2}')
if [ $MEM_USAGE -gt 90 ]; then
    echo "[$DATE] WARNING: Memory usage is ${MEM_USAGE}%" >> $LOGFILE
fi

# Check load average
LOAD_AVG=$(uptime | awk -F'load average:' '{print $2}' | awk '{print $1}' | sed
's/,//')
LOAD_THRESHOLD="2.0"
if (( $(echo "$LOAD_AVG > $LOAD_THRESHOLD" | bc -l) )); then
    echo "[$DATE] WARNING: High load average: $LOAD_AVG" >> $LOGFILE
fi

# Check failed services
FAILED_SERVICES=$(systemctl --failed --no-legend | wc -l)
if [ $FAILED_SERVICES -gt 0 ]; then
    echo "[$DATE] WARNING: $FAILED_SERVICES failed services detected" >> $LOGFILE
    systemctl --failed --no-legend >> $LOGFILE
```

```
fi

echo "[$DATE] System health check completed" >> $LOGFILE
```

💾 Disk and Filesystem Management

Disk Partitioning

```
# List block devices
$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda      8:0    0   20G  0 disk
├─sda1   8:1    0    1G  0 part /boot
├─sda2   8:2    0    2G  0 part [SWAP]
└─sda3   8:3    0   17G  0 part /
sdb      8:16   0   10G  0 disk

# Partition disk with fdisk
$ sudo fdisk /dev/sdb
Command (m for help): n     # New partition
Command (m for help): p     # Print partition table
Command (m for help): w     # Write changes

# Partition disk with parted
$ sudo parted /dev/sdb
(parted) mklabel gpt
(parted) mkpart primary ext4 0% 100%
(parted) print
(parted) quit

# Create filesystem
$ sudo mkfs.ext4 /dev/sdb1
$ sudo mkfs.xfs /dev/sdb1
$ sudo mkfs.btrfs /dev/sdb1

# Check filesystem
$ sudo fsck /dev/sdb1
$ sudo fsck.ext4 /dev/sdb1
$ sudo e2fsck -f /dev/sdb1
```

Mount Management

```
# Mount filesystem
$ sudo mkdir /mnt/data
$ sudo mount /dev/sdb1 /mnt/data

# Mount with specific options
$ sudo mount -o rw,noexec,nosuid /dev/sdb1 /mnt/data
```

```
# Unmount filesystem
$ sudo umount /mnt/data
$ sudo umount /dev/sdb1

# Force unmount (if busy)
$ sudo fuser -km /mnt/data
$ sudo umount -f /mnt/data

# Permanent mounts in /etc/fstab
$ sudo nano /etc/fstab
# Add line:
/dev/sdb1 /mnt/data ext4 defaults,noatime 0 2

# Test fstab entries
$ sudo mount -a

# View mounted filesystems
$ mount | grep ^/dev
$ df -h
$ findmnt
```

LVM (Logical Volume Management)

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

# Create volume group
$ sudo vgcreate data_vg /dev/sdb1
$ sudo vgdisplay

# Create logical volume
$ sudo lvcreate -L 5G -n data_lv data_vg
$ sudo lvdisplay

# Create filesystem on LV
$ sudo mkfs.ext4 /dev/data_vg/data_lv

# Mount logical volume
$ sudo mkdir /mnt/lvm_data
$ sudo mount /dev/data_vg/data_lv /mnt/lvm_data

# Extend logical volume
$ sudo lvextend -L +2G /dev/data_vg/data_lv
$ sudo resize2fs /dev/data_vg/data_lv  # For ext4
$ sudo xfs_growfs /mnt/lvm_data        # For XFS

# Add physical volume to VG
$ sudo pvcreate /dev/sdc1
$ sudo vgextend data_vg /dev/sdc1
```

🔒 Security Hardening Basics

SSH Security

```
# Configure SSH security
$ sudo nano /etc/ssh/sshd_config

# Recommended settings:
Port 2222                          # Change default port
PermitRootLogin no                 # Disable root login
PasswordAuthentication no          # Use key-based auth only
PubkeyAuthentication yes
MaxAuthTries 3
ClientAliveInterval 300
ClientAliveCountMax 2
AllowUsers john jane               # Limit allowed users
DenyUsers guest                    # Deny specific users

# Restart SSH service
$ sudo systemctl restart sshd

# Generate SSH key pair
$ ssh-keygen -t rsa -b 4096 -C "user@example.com"
$ ssh-keygen -t ed25519 -C "user@example.com"  # More secure

# Copy public key to server
$ ssh-copy-id user@server
$ ssh-copy-id -i ~/.ssh/id_ed25519.pub user@server
```

Firewall Configuration

```
# UFW (Uncomplicated Firewall)
$ sudo ufw enable
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
$ sudo ufw allow ssh
$ sudo ufw allow 80/tcp
$ sudo ufw allow 443/tcp
$ sudo ufw allow from 192.168.1.0/24
$ sudo ufw status verbose

# iptables (more advanced)
$ sudo iptables -L
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
$ sudo iptables -A INPUT -j DROP
$ sudo iptables-save > /etc/iptables/rules.v4
```

System Security Auditing

```
# Check for rootkits
$ sudo apt install rkhunter chkrootkit
$ sudo rkhunter --check
$ sudo chkrootkit

# File integrity monitoring
$ sudo apt install aide
$ sudo aideinit
$ sudo aide --check

# Check for SUID/SGID files
$ find / -type f \( -perm -4000 -o -perm -2000 \) -exec ls -l {} \; 2>/dev/null

# Check for world-writable files
$ find / -type f -perm -002 -exec ls -l {} \; 2>/dev/null

# Monitor login attempts
$ sudo tail -f /var/log/auth.log | grep "Failed password"
$ sudo lastb                          # Failed login attempts
$ sudo last                           # Successful logins
```

⏰ Automation with Cron

Cron Job Management

```
# Edit user crontab
$ crontab -e

# List user crontab
$ crontab -l

# Remove user crontab
$ crontab -r

# Edit system-wide crontab
$ sudo nano /etc/crontab

# Cron format:
# minute hour day month weekday command
# 0-59   0-23 1-31 1-12  0-7 (0 and 7 are Sunday)

# Examples:
0 2 * * *     /usr/local/bin/backup.sh           # Daily at 2 AM
*/15 * * * *  /usr/local/bin/check-health.sh      # Every 15 minutes
0 0 * * 0     /usr/local/bin/weekly-cleanup.sh    # Weekly on Sunday
0 0 1 * *     /usr/local/bin/monthly-report.sh    # Monthly on 1st
@reboot       /usr/local/bin/startup.sh           # At boot
@daily        /usr/local/bin/daily-tasks.sh       # Daily
@weekly       /usr/local/bin/weekly-tasks.sh      # Weekly
```

Cron Directories

```
# System cron directories
/etc/cron.hourly/     # Scripts run hourly
/etc/cron.daily/      # Scripts run daily
/etc/cron.weekly/     # Scripts run weekly
/etc/cron.monthly/    # Scripts run monthly

# Place executable scripts in these directories
$ sudo cp backup.sh /etc/cron.daily/
$ sudo chmod +x /etc/cron.daily/backup.sh

# Check cron logs
$ sudo grep CRON /var/log/syslog
$ sudo journalctl -u cron
```

Backup Automation Example

```
#!/bin/bash
# /usr/local/bin/backup.sh - Automated backup script

BACKUP_DIR="/backup"
SOURCE_DIRS="/home /etc /var/www"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_FILE="$BACKUP_DIR/system_backup_$DATE.tar.gz"
LOG_FILE="/var/log/backup.log"

# Create backup directory if it doesn't exist
mkdir -p $BACKUP_DIR

# Log start
echo "[$(date)] Starting backup" >> $LOG_FILE

# Create compressed backup
tar -czf $BACKUP_FILE $SOURCE_DIRS 2>>$LOG_FILE

if [ $? -eq 0 ]; then
    echo "[$(date)] Backup completed successfully: $BACKUP_FILE" >> $LOG_FILE

    # Remove backups older than 7 days
    find $BACKUP_DIR -name "system_backup_*.tar.gz" -mtime +7 -delete

    # Send success notification
    echo "Backup completed: $BACKUP_FILE" | mail -s "Backup Success"
admin@example.com
else
    echo "[$(date)] Backup failed" >> $LOG_FILE
    echo "Backup failed. Check $LOG_FILE for details" | mail -s "Backup Failed"
admin@example.com
fi
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between `apt remove` and `apt purge`?**

   ▶ Answer

   `apt remove` uninstalls the package but keeps configuration files, while `apt purge` removes both the package and its configuration files.

2. **How do you make a filesystem mount automatically at boot?**

   ▶ Answer

   Add an entry to `/etc/fstab` with the appropriate mount options and run `sudo mount -a` to test.

3. **What does the cron expression `0 */6 * * *` mean?**

   ▶ Answer

   Run at minute 0 of every 6th hour (00:00, 06:00, 12:00, 18:00) every day.

4. **How do you check which package provides a specific file?**

   ▶ Answer

   ```
   dpkg -S /path/to/file        # Debian/Ubuntu
   rpm -qf /path/to/file        # Red Hat/CentOS
   ```

Hands-On Challenges

Challenge 1: User Management System

```
# Create a script that:
# - Adds users from a CSV file
# - Sets up proper home directories
# - Assigns users to appropriate groups
# - Sets password policies
# - Generates a report of created users
```

Challenge 2: System Monitoring Dashboard

```
# Create a monitoring system that:
# - Checks disk usage, memory, and CPU
# - Monitors critical services
# - Sends alerts when thresholds are exceeded
```

```
# - Logs all activities
# - Runs automatically via cron
```

Challenge 3: Automated Backup Solution

```
# Design a backup system that:
# - Backs up multiple directories
# - Implements rotation (keep last 30 days)
# - Compresses and encrypts backups
# - Verifies backup integrity
# - Sends status notifications
```

🚀 Next Steps

Excellent! You've mastered essential Linux system administration. You can now:

- Manage users and groups effectively
- Handle package management across distributions
- Monitor system resources and logs
- Configure disks and filesystems
- Implement basic security measures
- Automate tasks with cron
- Troubleshoot common system issues

Ready for advanced networking? Continue to 07-ssh-remote-access.md to master secure remote access and administration.

> **Pro Tip**: Always test system changes in a non-production environment first. Keep regular backups, monitor system logs, and document your configurations. Automation is powerful, but always include error handling and logging in your scripts! 🛠️

🔐 SSH & Remote Access: Secure System Administration

> **Master secure remote access, file transfers, and advanced SSH techniques**

📖 What You'll Learn

SSH (Secure Shell) is the cornerstone of secure remote system administration. This chapter covers everything from basic connections to advanced SSH techniques:

- SSH fundamentals and security principles
- Key-based authentication setup
- SSH configuration and hardening
- Secure file transfers (SCP, SFTP, rsync)

- SSH tunneling and port forwarding
- SSH agent and key management
- Remote command execution
- Troubleshooting SSH connections

🌍 Why This Matters

Critical applications:

- **Remote Administration**: Manage servers from anywhere securely
- **DevOps**: Deploy applications and manage infrastructure
- **Security**: Replace insecure protocols like Telnet and FTP
- **Automation**: Execute remote commands in scripts
- **File Management**: Transfer files securely between systems

🔑 SSH Fundamentals

Basic SSH Connection

```
# Basic connection
$ ssh username@hostname
$ ssh user@192.168.1.100
$ ssh user@example.com

# Connect with specific port
$ ssh -p 2222 user@hostname

# Connect with verbose output (debugging)
$ ssh -v user@hostname
$ ssh -vv user@hostname     # More verbose
$ ssh -vvv user@hostname    # Maximum verbosity

# Connect and execute command
$ ssh user@hostname 'ls -la'
$ ssh user@hostname 'df -h && free -h'

# Connect with X11 forwarding (GUI apps)
$ ssh -X user@hostname
$ ssh -Y user@hostname      # Trusted X11 forwarding

# Example connection output:
$ ssh john@192.168.1.100
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
ECDSA key fingerprint is SHA256:abc123def456...
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.100' (ECDSA) to the list of known hosts.
john@192.168.1.100's password:
Last login: Mon Dec 15 10:30:15 2023 from 192.168.1.50
john@server:~$
```

SSH Key Generation and Management

```
# Generate SSH key pair (RSA)
$ ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa
Your public key has been saved in /home/user/.ssh/id_rsa.pub

# Generate Ed25519 key (more secure, recommended)
$ ssh-keygen -t ed25519 -C "your_email@example.com"

# Generate key with custom filename
$ ssh-keygen -t ed25519 -f ~/.ssh/server_key -C "server access key"

# List SSH keys
$ ls -la ~/.ssh/
total 16
drwx------ 2 user user 4096 Dec 15 10:30 .
drwxr-xr-x 5 user user 4096 Dec 15 10:29 ..
-rw------- 1 user user  411 Dec 15 10:30 id_ed25519
-rw-r--r-- 1 user user   99 Dec 15 10:30 id_ed25519.pub
-rw-r--r-- 1 user user  222 Dec 15 10:25 known_hosts

# View public key
$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGq... your_email@example.com

# Copy public key to server
$ ssh-copy-id user@hostname
$ ssh-copy-id -i ~/.ssh/id_ed25519.pub user@hostname

# Manual key installation
$ cat ~/.ssh/id_ed25519.pub | ssh user@hostname 'mkdir -p ~/.ssh && cat >>
~/.ssh/authorized_keys'

# Set proper permissions
$ chmod 700 ~/.ssh
$ chmod 600 ~/.ssh/id_ed25519
$ chmod 644 ~/.ssh/id_ed25519.pub
$ chmod 600 ~/.ssh/authorized_keys
```

SSH Configuration

```
# User SSH config file
$ nano ~/.ssh/config

# Example configuration:
```

```
Host server1
    HostName 192.168.1.100
    User john
    Port 2222
    IdentityFile ~/.ssh/server1_key
    IdentitiesOnly yes

Host *.example.com
    User admin
    Port 22
    IdentityFile ~/.ssh/company_key
    ForwardAgent yes

Host bastion
    HostName bastion.example.com
    User jumpuser
    Port 22
    IdentityFile ~/.ssh/bastion_key

Host internal-server
    HostName 10.0.1.100
    User admin
    ProxyJump bastion
    IdentityFile ~/.ssh/internal_key

# Global SSH client config
$ sudo nano /etc/ssh/ssh_config

# Common client settings:
Host *
    ServerAliveInterval 60
    ServerAliveCountMax 3
    TCPKeepAlive yes
    Compression yes
    ControlMaster auto
    ControlPath ~/.ssh/master-%r@%h:%p
    ControlPersist 10m

# Connect using config
$ ssh server1                # Uses config settings
$ ssh internal-server        # Automatically uses bastion as jump host
```

🛡 SSH Server Configuration and Hardening

SSH Server Configuration

```
# Edit SSH server config
$ sudo nano /etc/ssh/sshd_config

# Security hardening settings:
Port 2222                          # Change default port
```

```
Protocol 2                          # Use SSH protocol 2 only
PermitRootLogin no                  # Disable root login
PasswordAuthentication no           # Disable password auth
PubkeyAuthentication yes            # Enable key-based auth
AuthenticationMethods publickey     # Require public key auth
PermitEmptyPasswords no             # No empty passwords
MaxAuthTries 3                      # Limit auth attempts
MaxSessions 2                       # Limit concurrent sessions
MaxStartups 2                       # Limit connection attempts

# User and group restrictions
AllowUsers john jane admin         # Only allow specific users
DenyUsers guest nobody              # Deny specific users
AllowGroups ssh-users admin        # Only allow specific groups
DenyGroups guests                   # Deny specific groups

# Network settings
ListenAddress 0.0.0.0              # Listen on all interfaces
ListenAddress 192.168.1.100        # Listen on specific IP
AddressFamily inet                  # IPv4 only (or inet6 for IPv6)
TCPKeepAlive yes
ClientAliveInterval 300             # Send keepalive every 5 minutes
ClientAliveCountMax 2               # Disconnect after 2 failed keepalives

# Logging
LogLevel VERBOSE                    # Detailed logging
SyslogFacility AUTH                # Log to auth facility

# Disable dangerous features
X11Forwarding no                    # Disable X11 forwarding
AllowTcpForwarding no              # Disable TCP forwarding
GatewayPorts no                    # Disable gateway ports
PermitTunnel no                    # Disable tunneling

# Banner and MOTD
Banner /etc/ssh/banner             # Display banner before login
PrintMotd yes                      # Show message of the day

# Restart SSH service after changes
$ sudo systemctl restart sshd
$ sudo systemctl status sshd

# Test configuration
$ sudo sshd -t                     # Test config syntax
$ sudo sshd -T                     # Show effective configuration
```

SSH Security Best Practices

```
# Create SSH banner
$ sudo nano /etc/ssh/banner
***************************************************************************
```

```
                    AUTHORIZED ACCESS ONLY
***************************************************************************
This system is for authorized users only. All activities are monitored
and logged. Unauthorized access is prohibited and will be prosecuted.
***************************************************************************

# Monitor SSH connections
$ sudo tail -f /var/log/auth.log | grep sshd
$ sudo journalctl -u sshd -f

# Check failed login attempts
$ sudo grep "Failed password" /var/log/auth.log
$ sudo lastb                          # Show failed logins

# Check successful logins
$ sudo last                           # Show successful logins
$ w                                   # Show current users
$ who                                 # Show logged-in users

# Install and configure fail2ban
$ sudo apt install fail2ban
$ sudo nano /etc/fail2ban/jail.local

[sshd]
enabled = true
port = 2222
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600
findtime = 600

$ sudo systemctl restart fail2ban
$ sudo fail2ban-client status sshd
```

🗀 Secure File Transfers

SCP (Secure Copy)

```
# Copy file to remote server
$ scp file.txt user@hostname:/path/to/destination/
$ scp -P 2222 file.txt user@hostname:/home/user/  # Custom port

# Copy file from remote server
$ scp user@hostname:/path/to/file.txt ./
$ scp user@hostname:/path/to/file.txt /local/path/

# Copy directory recursively
$ scp -r /local/directory user@hostname:/remote/path/
$ scp -r user@hostname:/remote/directory ./
```

```
# Copy with compression
$ scp -C large_file.tar.gz user@hostname:/tmp/

# Copy preserving permissions and timestamps
$ scp -p file.txt user@hostname:/path/

# Copy multiple files
$ scp file1.txt file2.txt user@hostname:/path/
$ scp *.txt user@hostname:/path/

# Copy with progress and verbose output
$ scp -v file.txt user@hostname:/path/

# Copy through jump host
$ scp -o ProxyJump=bastion file.txt user@target:/path/
```

SFTP (SSH File Transfer Protocol)

```
# Connect to SFTP server
$ sftp user@hostname
$ sftp -P 2222 user@hostname      # Custom port

sftp> help                        # Show available commands
sftp> pwd                         # Show remote working directory
sftp> lpwd                        # Show local working directory
sftp> ls                          # List remote files
sftp> lls                         # List local files
sftp> cd /path/to/directory       # Change remote directory
sftp> lcd /local/path             # Change local directory

# File operations
sftp> get remote_file.txt         # Download file
sftp> get -r remote_directory     # Download directory recursively
sftp> put local_file.txt          # Upload file
sftp> put -r local_directory      # Upload directory recursively
sftp> mget *.txt                  # Download multiple files
sftp> mput *.txt                  # Upload multiple files

# File management
sftp> mkdir new_directory         # Create remote directory
sftp> rmdir directory             # Remove remote directory
sftp> rm file.txt                 # Delete remote file
sftp> rename old.txt new.txt      # Rename remote file
sftp> chmod 644 file.txt          # Change remote file permissions

sftp> quit                        # Exit SFTP

# Batch SFTP operations
$ echo -e "cd /remote/path\nput file.txt\nquit" | sftp user@hostname

# SFTP with script file
```

```
$ nano sftp_commands.txt
cd /remote/path
put *.txt
get *.log
quit

$ sftp -b sftp_commands.txt user@hostname
```

Rsync over SSH

```
# Basic rsync over SSH
$ rsync -avz /local/path/ user@hostname:/remote/path/
$ rsync -avz user@hostname:/remote/path/ /local/path/

# Rsync options explained:
# -a: archive mode (preserves permissions, timestamps, etc.)
# -v: verbose output
# -z: compress data during transfer
# -h: human-readable output
# -P: show progress and keep partial files
# --delete: delete files in destination that don't exist in source

# Sync with progress and delete
$ rsync -avzP --delete /local/path/ user@hostname:/remote/path/

# Exclude files and directories
$ rsync -avz --exclude='*.log' --exclude='tmp/' /local/path/
user@hostname:/remote/path/

# Dry run (test without making changes)
$ rsync -avzn /local/path/ user@hostname:/remote/path/

# Rsync with custom SSH port
$ rsync -avz -e 'ssh -p 2222' /local/path/ user@hostname:/remote/path/

# Rsync with bandwidth limit
$ rsync -avz --bwlimit=1000 /local/path/ user@hostname:/remote/path/

# Backup script using rsync
#!/bin/bash
SOURCE="/home/user/documents"
DEST="backup@server:/backups/$(hostname)"
LOGFILE="/var/log/backup.log"

echo "[$(date)] Starting backup" >> $LOGFILE
rsync -avz --delete --log-file=$LOGFILE $SOURCE $DEST
if [ $? -eq 0 ]; then
    echo "[$(date)] Backup completed successfully" >> $LOGFILE
else
    echo "[$(date)] Backup failed" >> $LOGFILE
fi
```

🌐 SSH Tunneling and Port Forwarding

Local Port Forwarding

```
# Forward local port to remote service
$ ssh -L 8080:localhost:80 user@hostname
# Access remote web server at http://localhost:8080

# Forward to different remote host
$ ssh -L 3306:database.internal:3306 user@gateway
# Access internal database at localhost:3306

# Multiple port forwards
$ ssh -L 8080:web.internal:80 -L 3306:db.internal:3306 user@gateway

# Background tunnel
$ ssh -f -N -L 8080:localhost:80 user@hostname
# -f: run in background
# -N: don't execute remote command

# Example: Access internal web application
$ ssh -L 8080:intranet.company.com:80 user@gateway.company.com
# Now browse to http://localhost:8080
```

Remote Port Forwarding

```
# Forward remote port to local service
$ ssh -R 8080:localhost:80 user@hostname
# Remote users can access your local web server via hostname:8080

# Forward to different local host
$ ssh -R 3306:database.local:3306 user@remote

# Example: Share local development server
$ ssh -R 8080:localhost:3000 user@public-server
# Others can access your dev server at public-server:8080

# Persistent remote tunnel
$ ssh -f -N -R 8080:localhost:80 user@hostname
```

Dynamic Port Forwarding (SOCKS Proxy)

```
# Create SOCKS proxy
$ ssh -D 1080 user@hostname
# Configure browser to use localhost:1080 as SOCKS proxy
# Background SOCKS proxy
$ ssh -f -N -D 1080 user@hostname

# Use with curl
$ curl --socks5 localhost:1080 http://internal.website.com

# Configure Firefox for SOCKS proxy:
# Preferences > Network Settings > Manual proxy configuration
# SOCKS Host: localhost, Port: 1080, SOCKS v5
```

SSH Jump Hosts (ProxyJump)

```
# Connect through jump host
$ ssh -J jumphost user@target
$ ssh -J user1@jump1,user2@jump2 user@target  # Multiple jumps

# Using ProxyCommand (older method)
$ ssh -o ProxyCommand="ssh -W %h:%p user@jumphost" user@target

# Configure in ~/.ssh/config
Host target-server
    HostName 10.0.1.100
    User admin
    ProxyJump jumphost

Host jumphost
    HostName jump.example.com
    User jumpuser
    Port 22

# Now simply:
$ ssh target-server
```

🔧 SSH Agent and Key Management

SSH Agent

```
# Start SSH agent
$ eval $(ssh-agent)
Agent pid 12345

# Add keys to agent
$ ssh-add ~/.ssh/id_ed25519
Enter passphrase for /home/user/.ssh/id_ed25519:
Identity added: /home/user/.ssh/id_ed25519 (user@hostname)

# Add all keys
$ ssh-add
```

```
# List loaded keys
$ ssh-add -l
256 SHA256:abc123... user@hostname (ED25519)

# Remove key from agent
$ ssh-add -d ~/.ssh/id_ed25519

# Remove all keys
$ ssh-add -D

# Kill SSH agent
$ ssh-agent -k

# Auto-start SSH agent in ~/.bashrc
if [ -z "$SSH_AUTH_SOCK" ]; then
    eval $(ssh-agent -s)
    ssh-add ~/.ssh/id_ed25519
fi
```

SSH Agent Forwarding

```
# Enable agent forwarding
$ ssh -A user@hostname

# Configure in ~/.ssh/config
Host *
    ForwardAgent yes

# Test agent forwarding
$ ssh server1
user@server1:~$ ssh server2  # Uses forwarded agent

# Security note: Only use agent forwarding with trusted hosts
```

Key Management Best Practices

```
# Use different keys for different purposes
$ ssh-keygen -t ed25519 -f ~/.ssh/work_key -C "work access"
$ ssh-keygen -t ed25519 -f ~/.ssh/personal_key -C "personal servers"
$ ssh-keygen -t ed25519 -f ~/.ssh/github_key -C "github access"

# Configure specific keys in ~/.ssh/config
Host work-server
    HostName work.example.com
    User admin
    IdentityFile ~/.ssh/work_key
    IdentitiesOnly yes

Host github.com
```

```
    User git
    IdentityFile ~/.ssh/github_key
    IdentitiesOnly yes

# Rotate keys regularly
$ ssh-keygen -t ed25519 -f ~/.ssh/new_key
$ ssh-copy-id -i ~/.ssh/new_key.pub user@hostname
# Test new key, then remove old key from server

# Backup keys securely
$ tar -czf ssh_keys_backup.tar.gz ~/.ssh/
$ gpg -c ssh_keys_backup.tar.gz  # Encrypt backup
```

🔍 Troubleshooting SSH

Common SSH Issues

```
# Connection refused
$ ssh -v user@hostname
# Check if SSH service is running on target
$ sudo systemctl status sshd
$ sudo netstat -tlnp | grep :22

# Permission denied (publickey)
# Check key permissions
$ ls -la ~/.ssh/
$ chmod 700 ~/.ssh
$ chmod 600 ~/.ssh/id_ed25519
$ chmod 644 ~/.ssh/id_ed25519.pub

# Check server-side authorized_keys
$ ssh user@hostname
$ ls -la ~/.ssh/authorized_keys
$ chmod 600 ~/.ssh/authorized_keys

# Host key verification failed
$ ssh-keygen -R hostname        # Remove old host key
$ ssh-keyscan hostname >> ~/.ssh/known_hosts  # Add new key

# Connection timeout
# Check firewall rules
$ sudo ufw status
$ sudo iptables -L

# Check network connectivity
$ ping hostname
$ telnet hostname 22
$ nmap -p 22 hostname

# Too many authentication failures
$ ssh -o IdentitiesOnly=yes -i ~/.ssh/specific_key user@hostname
```

```
# Debug SSH connection
$ ssh -vvv user@hostname 2>&1 | tee ssh_debug.log
```

SSH Server Troubleshooting

```
# Check SSH server configuration
$ sudo sshd -t                      # Test config syntax
$ sudo sshd -T                      # Show effective config

# Check SSH server logs
$ sudo journalctl -u sshd -f
$ sudo tail -f /var/log/auth.log | grep sshd

# Check listening ports
$ sudo netstat -tlnp | grep sshd
$ sudo ss -tlnp | grep sshd

# Restart SSH service
$ sudo systemctl restart sshd
$ sudo systemctl status sshd

# Check SSH process
$ ps aux | grep sshd

# Test SSH from localhost
$ ssh localhost
$ ssh -p 22 127.0.0.1
```

Network Diagnostics

```
# Test SSH port connectivity
$ nc -zv hostname 22
$ telnet hostname 22
$ nmap -p 22 hostname

# Trace route to SSH server
$ traceroute hostname
$ mtr hostname

# Check DNS resolution
$ nslookup hostname
$ dig hostname

# Test with different SSH client options
$ ssh -o ConnectTimeout=10 user@hostname
$ ssh -o ServerAliveInterval=60 user@hostname
$ ssh -4 user@hostname              # Force IPv4
$ ssh -6 user@hostname              # Force IPv6
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between SSH local and remote port forwarding?**

   ▶ Answer

   Local forwarding (`-L`) forwards a local port to a remote service, while remote forwarding (`-R`) forwards a remote port to a local service.

2. **How do you create an SSH tunnel that runs in the background?**

   ▶ Answer

   Use `ssh -f -N -L localport:remotehost:remoteport user@hostname` where `-f` runs in background and `-N` doesn't execute commands.

3. **What's the most secure SSH key type to use?**

   ▶ Answer

   Ed25519 (`ssh-keygen -t ed25519`) is currently the most secure and efficient option.

4. **How do you disable password authentication and only allow key-based auth?**

   ▶ Answer

   Set `PasswordAuthentication no` and `PubkeyAuthentication yes` in `/etc/ssh/sshd_config`, then restart sshd.

Hands-On Challenges

Challenge 1: SSH Hardening

```
# Secure an SSH server by:
# - Changing the default port
# - Disabling root login
# - Setting up key-based authentication only
# - Configuring fail2ban
# - Setting up proper logging and monitoring
```

Challenge 2: SSH Tunnel Setup

```
# Create a setup where:
# - You access an internal web application through SSH tunnel
# - Set up a SOCKS proxy for browsing internal networks
# - Configure persistent tunnels that auto-reconnect
# - Use jump hosts to reach internal servers
```

Challenge 3: Automated File Sync

```
# Create a system that:
# - Syncs files between multiple servers using rsync over SSH
# - Implements proper error handling and logging
# - Runs automatically via cron
# - Sends notifications on success/failure
# - Handles network interruptions gracefully
```

🚀 Next Steps

Excellent! You've mastered SSH and secure remote access. You can now:

- Set up secure SSH connections with key-based authentication
- Configure and harden SSH servers
- Transfer files securely using SCP, SFTP, and rsync
- Create SSH tunnels for secure access to internal services
- Troubleshoot SSH connection issues
- Manage SSH keys and agents effectively

Ready for web services? Continue to 08-web-services.md to learn about setting up and managing web servers and services.

> **Pro Tip**: Always use key-based authentication, keep your SSH keys secure, and regularly rotate them. Use SSH agent forwarding carefully and only with trusted hosts. Monitor SSH logs for security threats and implement proper access controls! 🔐

🌐 Web Services: HTTP, HTTPS, and Server Management

> **Master web server setup, SSL/TLS configuration, and HTTP protocol fundamentals**

📖 What You'll Learn

Web services are the backbone of modern internet infrastructure. This chapter covers everything from basic HTTP concepts to advanced web server management:

- HTTP/HTTPS protocol fundamentals
- Web server installation and configuration (Apache, Nginx)
- SSL/TLS certificate management
- Virtual hosts and domain configuration
- Load balancing and reverse proxying
- Web server security and hardening
- Performance optimization

- Troubleshooting web services

🌍 Why This Matters

Critical applications:

- **Web Development**: Host websites and web applications
- **API Services**: Serve REST APIs and microservices
- **Content Delivery**: Serve static and dynamic content efficiently
- **Security**: Implement HTTPS and secure web communications
- **DevOps**: Deploy and manage web infrastructure

🔗 HTTP/HTTPS Protocol Fundamentals

Understanding HTTP

```
# Basic HTTP request with curl
$ curl -v http://example.com
* Trying 93.184.216.34:80...
* Connected to example.com (93.184.216.34) port 80 (#0)
> GET / HTTP/1.1
> Host: example.com
> User-Agent: curl/7.68.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=UTF-8
< Content-Length: 1256
< Server: ECS (dcb/7F83)
<
<!doctype html>
<html>
...

# HTTP methods demonstration
$ curl -X GET http://api.example.com/users
$ curl -X POST -H "Content-Type: application/json" -d '{"name":"John"}'
http://api.example.com/users
$ curl -X PUT -H "Content-Type: application/json" -d '{"name":"Jane"}'
http://api.example.com/users/1
$ curl -X DELETE http://api.example.com/users/1

# HTTP headers
$ curl -H "Authorization: Bearer token123" http://api.example.com/protected
$ curl -H "Accept: application/json" http://api.example.com/data
$ curl -H "User-Agent: MyApp/1.0" http://example.com

# View response headers only
$ curl -I http://example.com
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 1256
```

```
Server: ECS (dcb/7F83)
Date: Mon, 15 Dec 2023 10:30:00 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
ETag: "3147526947+gzip"
Expires: Mon, 22 Dec 2023 10:30:00 GMT
Cache-Control: max-age=604800

# HTTP status codes
$ curl -w "%{http_code}" -s -o /dev/null http://example.com
200

$ curl -w "%{http_code}" -s -o /dev/null http://example.com/nonexistent
404
```

HTTPS and SSL/TLS

```
# HTTPS request
$ curl -v https://example.com
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384

# Check SSL certificate
$ openssl s_client -connect example.com:443 -servername example.com
$ openssl s_client -connect example.com:443 -showcerts

# Check certificate expiration
$ echo | openssl s_client -connect example.com:443 2>/dev/null | openssl x509 -
noout -dates
notBefore=Oct 22 12:00:00 2023 GMT
notAfter=Oct 22 12:00:00 2024 GMT

# Test SSL configuration
$ curl --tlsv1.2 https://example.com
$ curl --tlsv1.3 https://example.com

# Ignore SSL certificate errors (for testing)
$ curl -k https://self-signed.example.com
$ curl --insecure https://self-signed.example.com
```

🔧 Apache Web Server

Apache Installation and Basic Configuration

```
# Install Apache
$ sudo apt update
$ sudo apt install apache2

# Start and enable Apache
$ sudo systemctl start apache2
$ sudo systemctl enable apache2
$ sudo systemctl status apache2

# Check Apache version
$ apache2 -v
Server version: Apache/2.4.41 (Ubuntu)
Server built:   2023-10-10T19:35:51

# Test Apache installation
$ curl http://localhost
$ curl http://$(hostname -I | awk '{print $1}')

# Apache configuration files
/etc/apache2/
├── apache2.conf          # Main configuration
├── sites-available/      # Available sites
├── sites-enabled/        # Enabled sites
├── mods-available/       # Available modules
├── mods-enabled/         # Enabled modules
├── conf-available/       # Available configurations
└── conf-enabled/         # Enabled configurations

# Enable/disable sites
$ sudo a2ensite default-ssl
$ sudo a2dissite 000-default
$ sudo systemctl reload apache2

# Enable/disable modules
$ sudo a2enmod ssl
$ sudo a2enmod rewrite
$ sudo a2dismod autoindex
$ sudo systemctl reload apache2

# Check enabled modules
$ apache2ctl -M
```

Apache Virtual Hosts

```
# Create virtual host configuration
$ sudo nano /etc/apache2/sites-available/example.com.conf

<VirtualHost *:80>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com
    ErrorLog ${APACHE_LOG_DIR}/example.com_error.log
    CustomLog ${APACHE_LOG_DIR}/example.com_access.log combined

    <Directory /var/www/example.com>
        Options -Indexes +FollowSymLinks
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>

# HTTPS virtual host
$ sudo nano /etc/apache2/sites-available/example.com-ssl.conf

<VirtualHost *:443>
    ServerName example.com
    ServerAlias www.example.com
    DocumentRoot /var/www/example.com

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/example.com.crt
    SSLCertificateKeyFile /etc/ssl/private/example.com.key
    SSLCertificateChainFile /etc/ssl/certs/example.com-chain.crt

    # Security headers
    Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains"
    Header always set X-Content-Type-Options nosniff
    Header always set X-Frame-Options DENY
    Header always set X-XSS-Protection "1; mode=block"

    ErrorLog ${APACHE_LOG_DIR}/example.com_ssl_error.log
    CustomLog ${APACHE_LOG_DIR}/example.com_ssl_access.log combined
</VirtualHost>

# Create document root
$ sudo mkdir -p /var/www/example.com
$ sudo chown -R www-data:www-data /var/www/example.com
$ sudo chmod -R 755 /var/www/example.com

# Create test page
$ sudo nano /var/www/example.com/index.html
<!DOCTYPE html>
<html>
<head>
    <title>Welcome to example.com</title>
</head>
<body>
    <h1>Hello from example.com!</h1>
    <p>This is a test page.</p>
</body>
</html>

# Enable site
```

```
$ sudo a2ensite example.com
$ sudo systemctl reload apache2
```

Apache Security and Performance

```
# Security configuration
$ sudo nano /etc/apache2/conf-available/security.conf

# Hide Apache version
ServerTokens Prod
ServerSignature Off

# Disable server-info and server-status
<Location "/server-info">
    Require all denied
</Location>

<Location "/server-status">
    Require all denied
</Location>

# Prevent access to .htaccess files
<FilesMatch "^\.ht">
    Require all denied
</FilesMatch>

# Enable security configuration
$ sudo a2enconf security
$ sudo systemctl reload apache2

# Performance tuning
$ sudo nano /etc/apache2/mods-available/mpm_prefork.conf

<IfModule mpm_prefork_module>
    StartServers             4
    MinSpareServers          20
    MaxSpareServers          40
    MaxRequestWorkers        200
    MaxConnectionsPerChild   4500
</IfModule>

# Enable compression
$ sudo a2enmod deflate
$ sudo nano /etc/apache2/mods-available/deflate.conf

<IfModule mod_deflate.c>
    AddOutputFilterByType DEFLATE text/plain
    AddOutputFilterByType DEFLATE text/html
    AddOutputFilterByType DEFLATE text/xml
    AddOutputFilterByType DEFLATE text/css
    AddOutputFilterByType DEFLATE application/xml
    AddOutputFilterByType DEFLATE application/xhtml+xml
    AddOutputFilterByType DEFLATE application/rss+xml
    AddOutputFilterByType DEFLATE application/javascript
    AddOutputFilterByType DEFLATE application/x-javascript
</IfModule>

# Enable caching
$ sudo a2enmod expires
$ sudo a2enmod headers
$ sudo nano /etc/apache2/mods-available/expires.conf

<IfModule mod_expires.c>
    ExpiresActive On
    ExpiresByType text/css "access plus 1 month"
    ExpiresByType application/javascript "access plus 1 month"
    ExpiresByType image/png "access plus 1 year"
    ExpiresByType image/jpg "access plus 1 year"
    ExpiresByType image/jpeg "access plus 1 year"
    ExpiresByType image/gif "access plus 1 year"
    ExpiresByType image/ico "access plus 1 year"
</IfModule>
```

Nginx Web Server

Nginx Installation and Basic Configuration

```
# Install Nginx
$ sudo apt update
$ sudo apt install nginx

# Start and enable Nginx
$ sudo systemctl start nginx
$ sudo systemctl enable nginx
$ sudo systemctl status nginx

# Check Nginx version
$ nginx -v
nginx version: nginx/1.18.0 (Ubuntu)

# Test Nginx configuration
$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful

# Reload Nginx configuration
$ sudo systemctl reload nginx
$ sudo nginx -s reload

# Nginx configuration structure
/etc/nginx/
├── nginx.conf            # Main configuration
├── sites-available/      # Available sites
├── sites-enabled/        # Enabled sites (symlinks)
├── conf.d/              # Additional configurations
└── snippets/            # Configuration snippets
```

Nginx Server Blocks (Virtual Hosts)

```
# Create server block configuration
$ sudo nano /etc/nginx/sites-available/example.com

server {
    listen 80;
    listen [::]:80;
    server_name example.com www.example.com;
    root /var/www/example.com;
    index index.html index.htm index.nginx-debian.html;

    location / {
        try_files $uri $uri/ =404;
    }

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Content-Security-Policy "default-src 'self' http: https: data:
blob: 'unsafe-inline'" always;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_proxied expired no-cache no-store private must-revalidate auth;
    gzip_types text/plain text/css text/xml text/javascript application/x-
javascript application/xml+rss;

    # Logging
    access_log /var/log/nginx/example.com.access.log;
    error_log /var/log/nginx/example.com.error.log;
}

# HTTPS server block
$ sudo nano /etc/nginx/sites-available/example.com-ssl

server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name example.com www.example.com;
    root /var/www/example.com;
    index index.html index.htm;
    # SSL configuration
    ssl_certificate /etc/ssl/certs/example.com.crt;
    ssl_certificate_key /etc/ssl/private/example.com.key;
    ssl_session_timeout 1d;
    ssl_session_cache shared:MozTLS:10m;
    ssl_session_tickets off;

    # Modern configuration
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-
ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384;
    ssl_prefer_server_ciphers off;

    # HSTS
    add_header Strict-Transport-Security "max-age=63072000" always;

    location / {
        try_files $uri $uri/ =404;
    }

    access_log /var/log/nginx/example.com-ssl.access.log;
    error_log /var/log/nginx/example.com-ssl.error.log;
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    listen [::]:80;
    server_name example.com www.example.com;
    return 301 https://$server_name$request_uri;
}

# Enable site
$ sudo ln -s /etc/nginx/sites-available/example.com /etc/nginx/sites-enabled/
$ sudo nginx -t
$ sudo systemctl reload nginx
```

Nginx as Reverse Proxy

```
# Reverse proxy configuration
$ sudo nano /etc/nginx/sites-available/app.example.com

upstream backend {
    server 127.0.0.1:3000;
    server 127.0.0.1:3001;
    server 127.0.0.1:3002;
}

server {
    listen 80;
    server_name app.example.com;

    location / {
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;

        # Timeouts
        proxy_connect_timeout 60s;
        proxy_send_timeout 60s;
        proxy_read_timeout 60s;
    }

    # Static files
    location /static/ {
        alias /var/www/app/static/;
        expires 1y;
        add_header Cache-Control "public, immutable";
    }

    # API rate limiting
    location /api/ {
        limit_req zone=api burst=10 nodelay;
        proxy_pass http://backend;
        # ... other proxy settings
    }
}

# Rate limiting configuration
$ sudo nano /etc/nginx/nginx.conf

http {
    # Rate limiting
    limit_req_zone $binary_remote_addr zone=api:10m rate=10r/s;
    limit_req_zone $binary_remote_addr zone=login:10m rate=1r/s;

    # Connection limiting
    limit_conn_zone $binary_remote_addr zone=conn_limit_per_ip:10m;
    limit_conn conn_limit_per_ip 20;

    # ... rest of configuration
}
```

🔒 SSL/TLS Certificate Management

Let's Encrypt with Certbot

```
# Install Certbot
$ sudo apt install certbot python3-certbot-apache  # For Apache
$ sudo apt install certbot python3-certbot-nginx   # For Nginx

# Obtain certificate (Apache)
$ sudo certbot --apache -d example.com -d www.example.com

# Obtain certificate (Nginx)
$ sudo certbot --nginx -d example.com -d www.example.com

# Manual certificate generation
$ sudo certbot certonly --standalone -d example.com -d www.example.com
$ sudo certbot certonly --webroot -w /var/www/example.com -d example.com

# List certificates
$ sudo certbot certificates
Found the following certs:
  Certificate Name: example.com
    Serial Number: 3a2b1c4d5e6f7890abcdef1234567890
    Key Type: RSA
    Domains: example.com www.example.com
    Expiry Date: 2024-03-15 10:30:00+00:00 (VALID: 89 days)
    Certificate Path: /etc/letsencrypt/live/example.com/fullchain.pem
    Private Key Path: /etc/letsencrypt/live/example.com/privkey.pem

# Renew certificates
$ sudo certbot renew
$ sudo certbot renew --dry-run  # Test renewal

# Auto-renewal with cron
$ sudo crontab -e
0 12 * * * /usr/bin/certbot renew --quiet

# Revoke certificate
$ sudo certbot revoke --cert-path /etc/letsencrypt/live/example.com/cert.pem
```

Self-Signed Certificates

```
# Generate private key
$ sudo openssl genrsa -out /etc/ssl/private/example.com.key 2048

# Generate certificate signing request
$ sudo openssl req -new -key /etc/ssl/private/example.com.key -out
/tmp/example.com.csr
Country Name (2 letter code) [AU]: US
State or Province Name (full name) [Some-State]: California
Locality Name (eg, city) []: San Francisco
Organization Name (eg, company) [Internet Widgits Pty Ltd]: Example Corp
Organizational Unit Name (eg, section) []: IT Department
Common Name (e.g. server FQDN or YOUR name) []: example.com
```

```
Email Address []: admin@example.com

# Generate self-signed certificate
$ sudo openssl x509 -req -days 365 -in /tmp/example.com.csr -signkey
/etc/ssl/private/example.com.key -out /etc/ssl/certs/example.com.crt

# Generate certificate and key in one command
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/example.com.key -out /etc/ssl/certs/example.com.crt

# Set proper permissions
$ sudo chmod 600 /etc/ssl/private/example.com.key
$ sudo chmod 644 /etc/ssl/certs/example.com.crt

# Verify certificate
$ openssl x509 -in /etc/ssl/certs/example.com.crt -text -noout
```

📊 Web Server Monitoring and Troubleshooting

Apache Monitoring

```
# Enable server status module
$ sudo a2enmod status
$ sudo nano /etc/apache2/mods-available/status.conf

<Location "/server-status">
    SetHandler server-status
    Require local
    Require ip 192.168.1
</Location>

<Location "/server-info">
    SetHandler server-info
    Require local
    Require ip 192.168.1
</Location>

# View server status
$ curl http://localhost/server-status
$ curl http://localhost/server-status?auto  # Machine readable

# Monitor Apache logs
$ sudo tail -f /var/log/apache2/access.log
$ sudo tail -f /var/log/apache2/error.log

# Analyze Apache logs
$ sudo grep "404" /var/log/apache2/access.log | head -10
$ sudo awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr |
head -10

# Check Apache processes
```

```
$ ps aux | grep apache2
$ sudo apache2ctl status
```

Nginx Monitoring

```
# Enable stub status module
$ sudo nano /etc/nginx/sites-available/default

server {
    listen 80;
    server_name localhost;

    location /nginx_status {
        stub_status on;
        access_log off;
        allow 127.0.0.1;
        allow 192.168.1.0/24;
        deny all;
    }
}

# View Nginx status
$ curl http://localhost/nginx_status
Active connections: 2
server accepts handled requests
 1000 1000 2000
Reading: 0 Writing: 1 Waiting: 1

# Monitor Nginx logs
$ sudo tail -f /var/log/nginx/access.log
$ sudo tail -f /var/log/nginx/error.log

# Analyze Nginx logs
$ sudo awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr |
head -10
$ sudo grep "error" /var/log/nginx/error.log | tail -10

# Check Nginx processes
$ ps aux | grep nginx
$ sudo nginx -T  # Show complete configuration
```

Performance Testing

```
# Apache Bench (ab)
$ ab -n 1000 -c 10 http://example.com/
This is ApacheBench, Version 2.3
Copyright 1996 Adam Twiss, Zeus Technology Ltd
...
Server Software:        nginx/1.18.0
Server Hostname:        example.com
Server Port:            80

Document Path:          /
Document Length:        612 bytes

Concurrency Level:      10
Time taken for tests:   0.123 seconds
Complete requests:      1000
Failed requests:        0
Total transferred:      853000 bytes
HTML transferred:       612000 bytes
Requests per second:    8130.08 [#/sec] (mean)
Time per request:       1.230 [ms] (mean)
Time per request:       0.123 [ms] (mean, across all concurrent requests)
Transfer rate:          6767.89 [Kbytes/sec] received

# wrk (modern alternative)
$ sudo apt install wrk
$ wrk -t12 -c400 -d30s http://example.com/
Running 30s test @ http://example.com/
  12 threads and 400 connections
  Thread Stats   Avg      Stdev     Max   +/- Stdev
    Latency    15.50ms   10.23ms 200.00ms   89.12%
    Req/Sec     2.15k   500.23     3.50k    68.75%
  774000 requests in 30.00s, 1.23GB read
Requests/sec:  25800.00
Transfer/sec:   42.00MB

# curl timing
$ curl -w "@curl-format.txt" -o /dev/null -s http://example.com/
# curl-format.txt:
     time_namelookup:  %{time_namelookup}\n
        time_connect:  %{time_connect}\n
     time_appconnect:  %{time_appconnect}\n
    time_pretransfer:  %{time_pretransfer}\n
       time_redirect:  %{time_redirect}\n
  time_starttransfer:  %{time_starttransfer}\n
                     ----------\n
          time_total:  %{time_total}\n
```

Common Troubleshooting

```
# Check if web server is running
$ sudo systemctl status apache2
$ sudo systemctl status nginx

# Check listening ports
$ sudo netstat -tlnp | grep :80
$ sudo ss -tlnp | grep :80
```

```
# Test configuration
$ sudo apache2ctl configtest
$ sudo nginx -t

# Check disk space
$ df -h /var/log
$ df -h /var/www

# Check file permissions
$ ls -la /var/www/example.com/
$ sudo find /var/www/example.com -type f -exec chmod 644 {} \;
$ sudo find /var/www/example.com -type d -exec chmod 755 {} \;

# Check DNS resolution
$ nslookup example.com
$ dig example.com

# Test SSL certificate
$ openssl s_client -connect example.com:443 -servername example.com
$ curl -I https://example.com

# Check firewall
$ sudo ufw status
$ sudo iptables -L

# Monitor real-time connections
$ sudo netstat -an | grep :80 | wc -l
$ watch 'sudo netstat -an | grep :80 | wc -l'
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between Apache and Nginx architecture?**

   ▶ Answer

   Apache uses a process/thread-based model where each request is handled by a separate process or thread, while Nginx uses an event-driven, asynchronous architecture that can handle many connections with fewer resources.

2. **How do you redirect HTTP to HTTPS in Nginx?**

   ▶ Answer

   ```
   server {
       listen 80;
       server_name example.com;
       return 301 https://$server_name$request_uri;
   }
   ```

3. **What does the HTTP status code 502 mean?**

   ▶ Answer

   502 Bad Gateway means the server acting as a gateway or proxy received an invalid response from the upstream server.

4. **How do you check SSL certificate expiration from command line?**

   ▶ Answer

   ```
   echo | openssl s_client -connect example.com:443 2>/dev/null | openssl x509
   -noout -dates
   ```

Hands-On Challenges

Challenge 1: Complete Web Server Setup

```
# Set up a web server that:
# - Serves multiple domains with virtual hosts
# - Implements SSL/TLS with Let's Encrypt
# - Includes security headers and hardening
# - Has proper logging and monitoring
# - Implements caching and compression
```

Challenge 2: Load Balancer Configuration

```
# Create a load balancer setup that:
# - Distributes traffic across multiple backend servers
# - Implements health checks
# - Handles SSL termination
# - Includes rate limiting
# - Provides failover capabilities
```

Challenge 3: Web Performance Optimization

```
# Optimize a web server for:
# - Maximum concurrent connections
# - Fastest response times
# - Efficient resource usage
# - Proper caching strategies
# - Security without performance impact
```

🚀 Next Steps

Excellent! You've mastered web services and HTTP/HTTPS. You can now:

- Set up and configure Apache and Nginx web servers
- Implement SSL/TLS certificates and HTTPS
- Create virtual hosts for multiple domains
- Configure reverse proxies and load balancing
- Monitor and troubleshoot web services
- Optimize web server performance
- Implement security best practices

Ready for advanced networking tools? Continue to 12-security-firewalls.md to learn about network security, firewalls, and intrusion detection.

> **Pro Tip**: Always use HTTPS in production, keep your web server software updated, monitor logs regularly, and implement proper security headers. Performance and security go hand in hand - optimize for both! 🌐

🛠 Basic Network Tools: Essential Commands

> **Master ping, traceroute, nslookup, dig, netstat, and ip commands for network diagnostics**

📖 What You'll Learn

Network troubleshooting requires the right tools. This chapter covers the essential command-line utilities that every network administrator, developer, and system administrator must know:

- Connectivity testing with `ping` and `traceroute`
- DNS troubleshooting with `nslookup` and `dig`
- Network interface management with `ip` and `ifconfig`
- Connection monitoring with `netstat` and `ss`
- Basic network scanning and testing
- Real-world troubleshooting workflows

🌍 Why This Matters

Real-world applications:

- **Troubleshooting**: Diagnose network connectivity issues quickly
- **Monitoring**: Check network performance and identify bottlenecks
- **Security**: Identify open ports and suspicious connections
- **Administration**: Configure network interfaces and routing
- **Development**: Test API endpoints and service connectivity

🏓 Connectivity Testing with `ping`

Basic `ping` Usage

`ping` sends ICMP Echo Request packets to test connectivity:

```
# Basic ping
$ ping google.com
PING google.com (142.250.191.14) 56(84) bytes of data.
64 bytes from lga25s62-in-f14.1e100.net (142.250.191.14): icmp_seq=1 ttl=117
time=12.3 ms
64 bytes from lga25s62-in-f14.1e100.net (142.250.191.14): icmp_seq=2 ttl=117
time=11.8 ms
64 bytes from lga25s62-in-f14.1e100.net (142.250.191.14): icmp_seq=3 ttl=117
time=12.1 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss
time 2003ms
rtt min/avg/max/mdev = 11.8/12.1/12.3/0.2 ms
```

Understanding `ping` Output

```
# Analyzing ping results:
# 64 bytes: packet size
# icmp_seq: sequence number
# ttl: Time To Live (hops remaining)
# time: round-trip time in milliseconds

# Packet loss indicates network issues:
# 0% = Perfect connectivity
# 1-5% = Minor issues
# >10% = Significant problems
# 100% = No connectivity
```

Advanced `ping` Options

```
# Ping specific number of times
$ ping -c 4 google.com

# Ping with larger packet size
$ ping -s 1000 google.com

# Ping with specific interval (default is 1 second)
$ ping -i 0.5 google.com  # Every 0.5 seconds

# Ping IPv6
$ ping6 google.com
$ ping6 2001:4860:4860::8888

# Flood ping (be careful!)
$ sudo ping -f google.com

# Ping with timestamp
```

```
$ ping -D google.com
[1702636200.123456] 64 bytes from google.com: icmp_seq=1 ttl=117 time=12.3 ms

# Set TTL value
$ ping -t 10 google.com

# Ping broadcast address (local network discovery)
$ ping -b 192.168.1.255
```

Ping Troubleshooting Scenarios

```
# Scenario 1: No response
$ ping 192.168.1.999
PING 192.168.1.999 (192.168.1.999) 56(84) bytes of data.
^C
--- 192.168.1.999 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss
# Possible causes: Wrong IP, firewall, device down

# Scenario 2: High latency
$ ping google.com
64 bytes from google.com: icmp_seq=1 ttl=117 time=500.3 ms
# Possible causes: Network congestion, poor connection

# Scenario 3: Intermittent loss
64 bytes from google.com: icmp_seq=1 ttl=117 time=12.3 ms
Request timeout for icmp_seq 2
64 bytes from google.com: icmp_seq=3 ttl=117 time=11.8 ms
# Possible causes: Unstable connection, interference
```

🗺 Path Tracing with `traceroute`

Basic `traceroute` Usage

`traceroute` shows the path packets take to reach a destination:

```
# Trace route to destination
$ traceroute google.com
traceroute to google.com (142.250.191.14), 30 hops max, 60 byte packets
 1  192.168.1.1 (192.168.1.1)  1.234 ms  1.123 ms  1.045 ms
 2  10.0.0.1 (10.0.0.1)  5.678 ms  5.432 ms  5.234 ms
 3  203.0.113.1 (203.0.113.1)  12.345 ms  12.123 ms  12.456 ms
 4  * * *
 5  142.250.191.14 (142.250.191.14)  15.678 ms  15.432 ms  15.234 ms
```

Understanding `traceroute` Output

```
# Each line represents a "hop" (router) in the path:
# 1. Hop number
# 2. Hostname/IP address
# 3. Three round-trip times (3 packets sent)

# Special symbols:
# * = No response (timeout)
# !H = Host unreachable
# !N = Network unreachable
# !P = Protocol unreachable
```

Advanced `traceroute` Options

```
# Use ICMP instead of UDP
$ traceroute -I google.com

# Use TCP (useful when UDP is blocked)
$ sudo traceroute -T -p 80 google.com

# IPv6 traceroute
$ traceroute6 google.com

# Set maximum hops
$ traceroute -m 15 google.com

# Set packet size
$ traceroute -s 1000 google.com

# Don't resolve hostnames (faster)
$ traceroute -n google.com
1  192.168.1.1  1.234 ms  1.123 ms  1.045 ms
2  10.0.0.1  5.678 ms  5.432 ms  5.234 ms
```

Traceroute Analysis

```
# Identifying network issues:

# High latency at specific hop:
 3  slow-router.isp.com  150.345 ms  149.123 ms  151.456 ms
 4  fast-router.isp.com  25.678 ms  25.432 ms  25.234 ms
# Problem: Hop 3 is slow (bottleneck)

# Timeouts in middle of path:
 5  * * *
 6  destination.com  15.678 ms  15.432 ms  15.234 ms
# Problem: Hop 5 doesn't respond but traffic gets through

# Complete failure:
 3  router.isp.com  12.345 ms  12.123 ms  12.456 ms
 4  * * *
 5  * * *
# Problem: Traffic blocked after hop 3
```

🔍 DNS Troubleshooting

Using `nslookup`

```
# Basic DNS lookup
$ nslookup google.com
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.191.14

# Reverse DNS lookup
$ nslookup 8.8.8.8
Server:     8.8.8.8
Address:    8.8.8.8#53

Non-authoritative answer:
8.8.8.8.in-addr.arpa    name = dns.google.

# Query specific DNS server
$ nslookup google.com 1.1.1.1
Server:     1.1.1.1
Address:    1.1.1.1#53

# Interactive mode
$ nslookup
> google.com
> set type=MX
> google.com
> exit
```

Advanced `nslookup` Queries

```
# Query different record types
$ nslookup -type=MX google.com
google.com  mail exchanger = 10 smtp.google.com.

$ nslookup -type=NS google.com
google.com  nameserver = ns1.google.com.
google.com  nameserver = ns2.google.com.

$ nslookup -type=TXT google.com
```

```
google.com  text = "v=spf1 include:_spf.google.com ~all"

$ nslookup -type=AAAA google.com
google.com  has AAAA address 2607:f8b0:4004:c1b::65
```

Using `dig` (More Powerful)

```
# Basic dig query
$ dig google.com

; <<>> DiG 9.16.1-Ubuntu <<>> google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12345
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; QUESTION SECTION:
;google.com.            IN  A

;; ANSWER SECTION:
google.com.     300 IN  A   142.250.191.14

;; Query time: 12 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Fri Dec 15 10:30:00 UTC 2023
;; MSG SIZE  rcvd: 55

# Short answer only
$ dig +short google.com
142.250.191.14

# Query specific record type
$ dig MX google.com
$ dig NS google.com
$ dig TXT google.com
$ dig AAAA google.com

# Reverse DNS lookup
$ dig -x 8.8.8.8
$ dig +short -x 8.8.8.8
dns.google.
```

Advanced `dig` Usage

```
# Query specific DNS server
$ dig @1.1.1.1 google.com

# Trace DNS resolution path
$ dig +trace google.com
```

```
# Show all record types
$ dig ANY google.com

# Query with TCP (instead of UDP)
$ dig +tcp google.com

# Batch queries from file
$ echo -e "google.com\nfacebook.com\ntwitter.com" | dig -f -

# Check DNSSEC validation
$ dig +dnssec google.com
```

DNS Troubleshooting Scenarios

```
# Scenario 1: Domain doesn't resolve
$ dig nonexistent.example.com
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN
# NXDOMAIN = Domain doesn't exist

# Scenario 2: DNS server timeout
$ dig @192.168.1.999 google.com
;; connection timed out; no servers could be reached

# Scenario 3: Different results from different servers
$ dig @8.8.8.8 example.com
$ dig @1.1.1.1 example.com
# Compare results for consistency
```

🔧 Network Interface Management

Using `ip` Command (Modern)

```
# Show all network interfaces
$ ip addr show
# or
$ ip a

1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 00:1b:44:11:3a:b7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic eth0
       valid_lft 86395sec preferred_lft 86395sec
```

```
# Show specific interface
$ ip addr show eth0

# Show only IPv4 addresses
$ ip -4 addr show

# Show only IPv6 addresses
$ ip -6 addr show
```

Interface Configuration with `ip`

```
# Bring interface up/down
$ sudo ip link set eth0 up
$ sudo ip link set eth0 down

# Add IP address to interface
$ sudo ip addr add 192.168.1.200/24 dev eth0

# Remove IP address from interface
$ sudo ip addr del 192.168.1.200/24 dev eth0

# Change MAC address
$ sudo ip link set dev eth0 address 00:1b:44:11:3a:b8

# Set MTU
$ sudo ip link set dev eth0 mtu 1400
```

Routing with `ip`

```
# Show routing table
$ ip route show
# or
$ ip r

default via 192.168.1.1 dev eth0 proto dhcp metric 100
192.168.1.0/24 dev eth0 proto kernel scope link src 192.168.1.100 metric 100

# Add static route
$ sudo ip route add 10.0.0.0/8 via 192.168.1.1

# Delete route
$ sudo ip route del 10.0.0.0/8

# Add default gateway
$ sudo ip route add default via 192.168.1.1

# Show route to specific destination
$ ip route get 8.8.8.8
8.8.8.8 via 192.168.1.1 dev eth0 src 192.168.1.100 uid 1000
```

Using `ifconfig` (Legacy but Common)

```
# Show all interfaces
$ ifconfig

# Show specific interface
$ ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
        ether 00:1b:44:11:3a:b7  txqueuelen 1000  (Ethernet)
        RX packets 12345  bytes 1234567 (1.2 MB)
        TX packets 6789   bytes 789012 (789.0 KB)

# Configure interface
$ sudo ifconfig eth0 192.168.1.200 netmask 255.255.255.0
$ sudo ifconfig eth0 up
$ sudo ifconfig eth0 down
```

📊 Connection Monitoring

Using `netstat`

```
# Show all connections
$ netstat -a

# Show listening ports
$ netstat -l
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN

# Show TCP connections
$ netstat -t

# Show UDP connections
$ netstat -u

# Show with process information
$ netstat -p

# Show numerical addresses (don't resolve)
$ netstat -n

# Common combinations
$ netstat -tulpn  # TCP, UDP, Listening, Process, Numerical
$ netstat -an     # All connections, Numerical
```

Advanced `netstat` Usage

```
# Show routing table
$ netstat -r
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
default         192.168.1.1     0.0.0.0         UG        0 0          0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 eth0

# Show interface statistics
$ netstat -i
Kernel Interface table
Iface      MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0      1500    12345      0      0 0          6789      0      0      0 BMRU
lo       65536      100      0      0 0           100      0      0      0 LRU

# Show network statistics
$ netstat -s
```

Using `ss` (Modern Alternative)

```
# Show all sockets
$ ss -a

# Show listening sockets
$ ss -l

# Show TCP sockets
$ ss -t

# Show UDP sockets
$ ss -u

# Show with process information
$ ss -p

# Common combination
$ ss -tulpn

# Show sockets for specific port
$ ss -tulpn | grep :80
$ ss -tulpn | grep :22

# Show established connections
$ ss -o state established
```

Practical Connection Analysis

```
# Find what's using port 80
$ ss -tulpn | grep :80
tcp   LISTEN 0      128          0.0.0.0:80        0.0.0.0:*    users:
(("nginx",pid=1234,fd=6))

# Find all connections to specific IP
$ ss -an | grep 192.168.1.100

# Count connections by state
$ ss -an | awk '{print $2}' | sort | uniq -c
     10 ESTAB
      5 LISTEN
      2 TIME-WAIT

# Monitor connections in real-time
$ watch -n 1 'ss -tulpn'
```

🎯 Practical Troubleshooting Workflow

Complete Network Diagnostic

```
#!/bin/bash
# network-check.sh - Comprehensive network diagnostic

echo "=== Network Diagnostic Report ==="
echo "Date: $(date)"
echo

# 1. Check network interfaces
echo "1. Network Interfaces:"
ip addr show | grep -E '^[0-9]+:|inet '
echo

# 2. Check routing
echo "2. Default Gateway:"
ip route | grep default
echo

# 3. Check DNS
echo "3. DNS Servers:"
cat /etc/resolv.conf | grep nameserver
echo

# 4. Test local connectivity
echo "4. Local Connectivity:"
ping -c 1 127.0.0.1 > /dev/null && echo "✅ Loopback OK" || echo "❌ Loopback
FAIL"
GATEWAY=$(ip route | grep default | awk '{print $3}')
if [ ! -z "$GATEWAY" ]; then
    ping -c 1 $GATEWAY > /dev/null && echo "✅ Gateway OK" || echo "❌ Gateway
```

```
FAIL"
fi
echo

# 5. Test DNS resolution
echo "5. DNS Resolution:"
nslookup google.com > /dev/null && echo "✅ DNS OK" || echo "❌ DNS FAIL"
echo

# 6. Test internet connectivity
echo "6. Internet Connectivity:"
ping -c 1 8.8.8.8 > /dev/null && echo "✅ Internet OK" || echo "❌ Internet FAIL"
echo

# 7. Show listening services
echo "7. Listening Services:"
ss -tulpn | grep LISTEN | head -10
echo

echo "=== End Report ==="
```

Web Service Testing

```
# Test HTTP connectivity
$ curl -I http://example.com
HTTP/1.1 200 OK
Date: Fri, 15 Dec 2023 10:30:00 GMT
Server: Apache/2.4.41

# Test HTTPS with timing
$ curl -w "@curl-format.txt" -o /dev/null -s https://google.com

# Create curl-format.txt:
$ cat > curl-format.txt << EOF
     time_namelookup:  %{time_namelookup}\n
        time_connect:  %{time_connect}\n
     time_appconnect:  %{time_appconnect}\n
    time_pretransfer:  %{time_pretransfer}\n
       time_redirect:  %{time_redirect}\n
  time_starttransfer:  %{time_starttransfer}\n
                     ----------\n
          time_total:  %{time_total}\n
EOF

# Test specific port connectivity
$ telnet google.com 80
$ nc -zv google.com 80
Connection to google.com 80 port [tcp/http] succeeded!
```

⚠ Common Tool Limitations and Alternatives

When Tools Don't Work

```
# ICMP might be blocked
$ ping google.com
# No response doesn't always mean the host is down

# Try TCP ping instead
$ nc -zv google.com 80
$ telnet google.com 80

# Some routers don't respond to traceroute
$ traceroute google.com
# Use different protocols
$ traceroute -I google.com  # ICMP
$ traceroute -T google.com  # TCP

# DNS might be cached
$ dig google.com
# Clear DNS cache
$ sudo systemctl flush-dns  # systemd-resolved
$ sudo service nscd restart  # nscd
```

Tool Alternatives

Traditional	Modern	Purpose
`ifconfig`	`ip`	Interface management
`route`	`ip route`	Routing table
`netstat`	`ss`	Socket statistics
`arp`	`ip neigh`	ARP table
`iwconfig`	`iw`	Wireless configuration

🧠 Knowledge Check

Quick Quiz

1. **What does a TTL of 1 in a ping response indicate?**

   ▶ Answer

   The packet has only 1 hop remaining before it's discarded. This usually means the responding device is very close (possibly the next router).

2. **How do you test if port 443 is open on google.com without using a web browser?**

   ▶ Answer

```
nc -zv google.com 443
# or
telnet google.com 443
# or
curl -I https://google.com
```

3. **What's the difference between `dig +short` and regular `dig`?**

   ▶ Answer

   `dig +short` returns only the answer (IP address), while regular `dig` returns the full DNS response including headers, query details, and timing information.

4. **How do you find which process is listening on port 80?**

   ▶ Answer

   ```
   ss -tulpn | grep :80
   # or
   netstat -tulpn | grep :80
   # or
   lsof -i :80
   ```

Hands-On Challenges

Challenge 1: Network Path Analysis

```
# Trace the route to 3 different websites
# Compare the paths and identify common routers
# Measure and compare latencies
```

Challenge 2: DNS Investigation

```
# Find all DNS record types for your domain
# Compare results from different DNS servers
# Identify the authoritative name servers
```

Challenge 3: Service Discovery

```
# Find all listening services on your system
# Identify which processes are using the network
# Create a summary report of open ports
```

🚀 Next Steps

Excellent! You've mastered the essential network diagnostic tools. You can now:

- Test connectivity and diagnose network issues
- Troubleshoot DNS problems effectively
- Monitor network connections and services
- Analyze network paths and performance

Ready for advanced techniques? Continue to 10-advanced-network-tools.md to learn about `nmap`, `tcpdump`, `wireshark`, and advanced network analysis.

> **Pro Tip**: Create aliases for commonly used command combinations like `alias netcheck='ss -tulpn'` and `alias myip='curl -s ifconfig.me'`. Build a personal toolkit of network diagnostic scripts that you can run quickly during troubleshooting sessions! 🔧

🔬 Advanced Network Tools: Professional Diagnostics

> **Master nmap, tcpdump, wireshark, iptables, and advanced network analysis techniques**

📖 What You'll Learn

Advanced network tools provide deep insights into network behavior, security, and performance. This chapter covers professional-grade utilities used by network administrators, security professionals, and system engineers:

- Network scanning and discovery with `nmap`
- Packet capture and analysis with `tcpdump` and `wireshark`
- Firewall management with `iptables` and `ufw`
- Network performance testing with `iperf3` and `mtr`
- Advanced troubleshooting techniques
- Security scanning and vulnerability assessment

🌍 Why This Matters

Professional applications:

- **Security**: Identify open ports, services, and vulnerabilities
- **Performance**: Analyze network bottlenecks and optimize traffic
- **Troubleshooting**: Deep packet inspection for complex issues
- **Monitoring**: Real-time network traffic analysis
- **Compliance**: Network security auditing and documentation

🗺 Network Discovery with `nmap`

Basic `nmap` Usage

`nmap` (Network Mapper) is the most powerful network discovery and security auditing tool:

```
# Basic host discovery
$ nmap 192.168.1.1
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for router.local (192.168.1.1)
Host is up (0.001s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
53/tcp   open  domain
80/tcp   open  http
443/tcp  open  https

# Scan multiple hosts
$ nmap 192.168.1.1-10
$ nmap 192.168.1.0/24
$ nmap google.com facebook.com

# Quick scan (top 100 ports)
$ nmap -F 192.168.1.1

# Scan all 65535 ports
$ nmap -p- 192.168.1.1

# Scan specific ports
$ nmap -p 22,80,443 192.168.1.1
$ nmap -p 1-1000 192.168.1.1
```

Advanced nmap Scanning

```
# TCP SYN scan (default, fast and stealthy)
$ nmap -sS 192.168.1.1

# TCP connect scan (when SYN scan not possible)
$ nmap -sT 192.168.1.1

# UDP scan (slower but important)
$ nmap -sU 192.168.1.1

# Service version detection
$ nmap -sV 192.168.1.1
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
80/tcp open  http    Apache httpd 2.4.41

# Operating system detection
$ nmap -O 192.168.1.1
Running: Linux 4.X|5.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5
OS details: Linux 4.15 - 5.6
```

```
# Aggressive scan (OS, version, script, traceroute)
$ nmap -A 192.168.1.1

# Script scanning
$ nmap --script vuln 192.168.1.1
$ nmap --script http-enum 192.168.1.1
$ nmap --script ssl-cert 192.168.1.1
```

Network Discovery Techniques

```
# Ping scan (discover live hosts)
$ nmap -sn 192.168.1.0/24
Nmap scan report for 192.168.1.1
Host is up (0.001s latency).
Nmap scan report for 192.168.1.100
Host is up (0.002s latency).
Nmap scan report for 192.168.1.150
Host is up (0.003s latency).

# ARP scan (local network only)
$ nmap -PR 192.168.1.0/24

# List scan (just list targets, no scanning)
$ nmap -sL 192.168.1.0/24

# Top ports scan
$ nmap --top-ports 1000 192.168.1.1

# Fast scan with service detection
$ nmap -sV -T4 -F 192.168.1.1

# Stealth scan (avoid detection)
$ nmap -sS -T2 -f 192.168.1.1
```

Practical `nmap` Examples

```
# Web server analysis
$ nmap -p 80,443 --script http-title,http-headers google.com

# SSH server analysis
$ nmap -p 22 --script ssh-hostkey,ssh-auth-methods 192.168.1.1

# SMB/NetBIOS analysis
$ nmap -p 139,445 --script smb-os-discovery 192.168.1.1

# Database server scan
$ nmap -p 3306,5432,1433 --script mysql-info,pgsql-info 192.168.1.1

# Vulnerability scanning
```

```
$ nmap --script vuln --script-args=unsafe=1 192.168.1.1

# Save results to file
$ nmap -oN scan_results.txt 192.168.1.0/24
$ nmap -oX scan_results.xml 192.168.1.0/24
```

📦 Packet Capture with `tcpdump`

Basic `tcpdump` Usage

`tcpdump` captures and analyzes network packets in real-time:

```
# Capture all traffic on interface
$ sudo tcpdump -i eth0
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
10:30:00.123456 IP 192.168.1.100.54321 > 8.8.8.8.53: UDP, length 32
10:30:00.125678 IP 8.8.8.8.53 > 192.168.1.100.54321: UDP, length 48

# Capture specific number of packets
$ sudo tcpdump -i eth0 -c 10

# Capture to file
$ sudo tcpdump -i eth0 -w capture.pcap

# Read from file
$ tcpdump -r capture.pcap

# More verbose output
$ sudo tcpdump -i eth0 -v
$ sudo tcpdump -i eth0 -vv
$ sudo tcpdump -i eth0 -vvv
```

Advanced `tcpdump` Filtering

```
# Filter by host
$ sudo tcpdump -i eth0 host 192.168.1.100
$ sudo tcpdump -i eth0 src 192.168.1.100
$ sudo tcpdump -i eth0 dst 192.168.1.100

# Filter by port
$ sudo tcpdump -i eth0 port 80
$ sudo tcpdump -i eth0 src port 80
$ sudo tcpdump -i eth0 dst port 80

# Filter by protocol
$ sudo tcpdump -i eth0 tcp
$ sudo tcpdump -i eth0 udp
$ sudo tcpdump -i eth0 icmp
```

```
# Complex filters
$ sudo tcpdump -i eth0 'tcp port 80 and host 192.168.1.100'
$ sudo tcpdump -i eth0 'udp port 53 or tcp port 80'
$ sudo tcpdump -i eth0 'not port 22'

# HTTP traffic analysis
$ sudo tcpdump -i eth0 -A 'tcp port 80'
$ sudo tcpdump -i eth0 -s 0 -A 'tcp port 80 and (tcp[tcpflags] & tcp-push != 0)'
```

Practical `tcpdump` Examples

```
# Monitor DNS queries
$ sudo tcpdump -i eth0 -n 'udp port 53'
10:30:00.123 IP 192.168.1.100.54321 > 8.8.8.8.53: 12345+ A? google.com. (28)
10:30:00.125 IP 8.8.8.8.53 > 192.168.1.100.54321: 12345 1/0/0 A 142.250.191.14
(44)

# Monitor HTTP requests
$ sudo tcpdump -i eth0 -A -s 1500 'tcp port 80 and (tcp[tcpflags] & tcp-push !=
0)'

# Monitor SSH connections
$ sudo tcpdump -i eth0 'tcp port 22'

# Monitor specific subnet
$ sudo tcpdump -i eth0 'net 192.168.1.0/24'

# Monitor large packets (potential issues)
$ sudo tcpdump -i eth0 'greater 1000'

# Monitor SYN packets (connection attempts)
$ sudo tcpdump -i eth0 'tcp[tcpflags] & tcp-syn != 0'

# Save and rotate capture files
$ sudo tcpdump -i eth0 -w capture-%Y%m%d-%H%M%S.pcap -G 3600 -C 100
```

Analyzing Captured Traffic

```
# Basic statistics
$ tcpdump -r capture.pcap | wc -l  # Count packets

# Protocol distribution
$ tcpdump -r capture.pcap -n | awk '{print $3}' | cut -d'.' -f1-4 | sort | uniq -c
| sort -nr

# Top talkers
$ tcpdump -r capture.pcap -n | awk '{print $3}' | cut -d'.' -f1-4 | sort | uniq -c
| sort -nr | head -10
```

```
# Extract HTTP requests
$ tcpdump -r capture.pcap -A | grep -E 'GET|POST|PUT|DELETE'

# Find specific patterns
$ tcpdump -r capture.pcap -A | grep -i 'password\|login\|auth'
```

🔍 Advanced Analysis with `wireshark`

Command-line Wireshark (`tshark`)

```
# Basic capture
$ tshark -i eth0

# Capture with display filter
$ tshark -i eth0 -f "tcp port 80"

# Capture to file
$ tshark -i eth0 -w capture.pcapng

# Read and analyze file
$ tshark -r capture.pcapng

# Protocol statistics
$ tshark -r capture.pcapng -q -z io,phs

# Conversation statistics
$ tshark -r capture.pcapng -q -z conv,tcp
$ tshark -r capture.pcapng -q -z conv,udp

# HTTP statistics
$ tshark -r capture.pcapng -q -z http,stat
$ tshark -r capture.pcapng -q -z http,tree
```

Advanced `tshark` Analysis

```
# Extract specific fields
$ tshark -r capture.pcapng -T fields -e ip.src -e ip.dst -e tcp.port

# Filter and extract HTTP data
$ tshark -r capture.pcapng -Y "http.request" -T fields -e http.host -e
http.request.uri

# DNS analysis
$ tshark -r capture.pcapng -Y "dns" -T fields -e dns.qry.name -e dns.resp.addr

# SSL/TLS analysis
$ tshark -r capture.pcapng -Y "ssl.handshake.type == 1" -T fields -e ip.src -e
ssl.handshake.extensions_server_name
```

```
# Export objects
$ tshark -r capture.pcapng --export-objects http,/tmp/http_objects/

# Follow TCP streams
$ tshark -r capture.pcapng -q -z follow,tcp,ascii,0
```

Firewall Management

Using `iptables`

```
# View current rules
$ sudo iptables -L
$ sudo iptables -L -n -v  # Numerical, verbose

Chain INPUT (policy ACCEPT)
target     prot opt source               destination
ACCEPT     all  --  anywhere             anywhere             ctstate
RELATED,ESTABLISHED
ACCEPT     all  --  anywhere             anywhere
INPUT_direct  all  --  anywhere             anywhere

# View specific chain
$ sudo iptables -L INPUT
$ sudo iptables -L OUTPUT
$ sudo iptables -L FORWARD

# View with line numbers
$ sudo iptables -L --line-numbers
```

Basic `iptables` Rules

```
# Allow SSH (port 22)
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow HTTP and HTTPS
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow from specific IP
$ sudo iptables -A INPUT -s 192.168.1.100 -j ACCEPT

# Block specific IP
$ sudo iptables -A INPUT -s 192.168.1.200 -j DROP

# Allow loopback
$ sudo iptables -A INPUT -i lo -j ACCEPT

# Allow established connections
```

```
$ sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

# Set default policies
$ sudo iptables -P INPUT DROP
$ sudo iptables -P FORWARD DROP
$ sudo iptables -P OUTPUT ACCEPT
```

Advanced `iptables` Usage

```
# Rate limiting (prevent DoS)
$ sudo iptables -A INPUT -p tcp --dport 22 -m limit --limit 3/min --limit-burst 3
-j ACCEPT

# Port forwarding
$ sudo iptables -t nat -A PREROUTING -p tcp --dport 8080 -j REDIRECT --to-port 80

# NAT (masquerading)
$ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE

# Log dropped packets
$ sudo iptables -A INPUT -j LOG --log-prefix "DROPPED: "
$ sudo iptables -A INPUT -j DROP

# Delete specific rule
$ sudo iptables -D INPUT 3  # Delete rule number 3
$ sudo iptables -D INPUT -p tcp --dport 80 -j ACCEPT  # Delete specific rule

# Flush all rules
$ sudo iptables -F
$ sudo iptables -t nat -F

# Save and restore rules
$ sudo iptables-save > /etc/iptables/rules.v4
$ sudo iptables-restore < /etc/iptables/rules.v4
```

Using `ufw` (Uncomplicated Firewall)

```
# Enable/disable firewall
$ sudo ufw enable
$ sudo ufw disable

# Check status
$ sudo ufw status
$ sudo ufw status verbose
$ sudo ufw status numbered

# Basic rules
$ sudo ufw allow 22
$ sudo ufw allow ssh
$ sudo ufw allow 80/tcp
$ sudo ufw allow 443/tcp

# Allow from specific IP
$ sudo ufw allow from 192.168.1.100
$ sudo ufw allow from 192.168.1.0/24

# Allow to specific port from specific IP
$ sudo ufw allow from 192.168.1.100 to any port 22

# Deny rules
$ sudo ufw deny 23
$ sudo ufw deny from 192.168.1.200

# Delete rules
$ sudo ufw delete allow 80
$ sudo ufw delete 3  # Delete rule number 3

# Reset firewall
$ sudo ufw --force reset
```

📊 Network Performance Testing

Using `iperf3`

```
# Server mode (on target machine)
$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------

# Client mode (test bandwidth)
$ iperf3 -c 192.168.1.100
Connecting to host 192.168.1.100, port 5201
[  5] local 192.168.1.200 port 54321 connected to 192.168.1.100 port 5201
[ ID] Interval           Transfer     Bitrate         Retr  Cwnd
[  5]   0.00-1.00   sec  112 MBytes   941 Mbits/sec    0    408 KBytes
[  5]   1.00-2.00   sec  112 MBytes   941 Mbits/sec    0    408 KBytes
...
[  5]   0.00-10.00  sec  1.10 GBytes   941 Mbits/sec    0             sender
[  5]   0.00-10.04  sec  1.10 GBytes   938 Mbits/sec                  receiver

# UDP test
$ iperf3 -c 192.168.1.100 -u

# Reverse test (server sends to client)
$ iperf3 -c 192.168.1.100 -R

# Bidirectional test
$ iperf3 -c 192.168.1.100 --bidir
```

```
# Custom duration and parallel streams
$ iperf3 -c 192.168.1.100 -t 60 -P 4

# Test specific bandwidth
$ iperf3 -c 192.168.1.100 -u -b 100M
```

Using `mtr` (My Traceroute)

```
# Real-time traceroute with statistics
$ mtr google.com
                             My traceroute  [v0.93]
host (192.168.1.100)                   2023-12-15T10:30:00+0000
Keys:  Help   Display mode   Restart statistics   Order of fields   quit
                                        Packets               Pings
 Host                                 Loss%   Snt   Last   Avg  Best  Wrst
StDev
 1. 192.168.1.1                       0.0%    10    1.2    1.3   1.1   1.8   0.2
 2. 10.0.0.1                          0.0%    10    5.4    5.6   5.2   6.1   0.3
 3. 203.0.113.1                       0.0%    10   12.3   12.5  12.1  13.2   0.4
 4. google.com                        0.0%    10   15.6   15.8  15.4  16.5   0.3

# Report mode (non-interactive)
$ mtr --report --report-cycles 10 google.com

# Show IP addresses only
$ mtr -n google.com

# Show both hostnames and IPs
$ mtr -b google.com

# UDP mode
$ mtr -u google.com

# TCP mode
$ mtr -T google.com
```

Advanced Troubleshooting Techniques

Network Latency Analysis

```bash
#!/bin/bash
# latency-test.sh - Comprehensive latency analysis

TARGET="$1"
if [ -z "$TARGET" ]; then
    echo "Usage: $0 <target>"
    exit 1
fi
```

```
echo "=== Latency Analysis for $TARGET ==="
echo

# Basic ping test
echo "1. Basic Connectivity:"
ping -c 4 "$TARGET" | tail -1
echo

# Detailed ping statistics
echo "2. Detailed Ping Statistics:"
ping -c 20 "$TARGET" | grep -E 'min/avg/max'
echo

# MTR analysis
echo "3. Path Analysis (MTR):"
mtr --report --report-cycles 5 "$TARGET"
echo

# Traceroute comparison
echo "4. Traceroute Analysis:"
traceroute "$TARGET" 2>/dev/null | tail -5
echo

echo "=== Analysis Complete ==="
```

Bandwidth Testing Script

```
#!/bin/bash
# bandwidth-test.sh - Multi-target bandwidth testing

TEST_SERVERS=(
    "iperf.scottlinux.com"
    "ping.online.net"
    "bouygues.iperf.fr"
)

echo "=== Bandwidth Testing ==="
echo

for server in "${TEST_SERVERS[@]}"; do
    echo "Testing against $server:"
    timeout 15 iperf3 -c "$server" -t 10 2>/dev/null | grep -E 'sender|receiver'
|| echo "Failed to connect"
    echo
    sleep 2
done

echo "=== Testing Complete ==="
```

Security Scanning Script

```bash
#!/bin/bash
# security-scan.sh - Basic security assessment

TARGET="$1"
if [ -z "$TARGET" ]; then
    echo "Usage: $0 <target>"
    exit 1
fi

echo "=== Security Scan for $TARGET ==="
echo

# Host discovery
echo "1. Host Discovery:"
nmap -sn "$TARGET" 2>/dev/null | grep -E 'Nmap scan report|Host is up'
echo

# Port scan
echo "2. Open Ports:"
nmap -F "$TARGET" 2>/dev/null | grep -E 'open|filtered'
echo

# Service detection
echo "3. Service Detection:"
nmap -sV --top-ports 100 "$TARGET" 2>/dev/null | grep -E 'open.*tcp'
echo

# Basic vulnerability check
echo "4. Basic Vulnerability Check:"
nmap --script vuln --top-ports 100 "$TARGET" 2>/dev/null | grep -E
'VULNERABLE|CVE'
echo

echo "=== Scan Complete ==="
echo "Note: This is a basic scan. Use professional tools for comprehensive
security assessment."
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between `nmap -sS` and `nmap -sT`?**

   ▶ Answer

   `-sS` performs a SYN scan (half-open scan) which is faster and more stealthy, while `-sT` performs a full TCP connect scan which completes the three-way handshake.

2. **How do you capture only HTTP traffic with tcpdump?**

   ▶ Answer

```
sudo tcpdump -i eth0 'tcp port 80'
# or for both HTTP and HTTPS
sudo tcpdump -i eth0 'tcp port 80 or tcp port 443'
```

3. **What iptables command blocks all traffic from IP 192.168.1.100?**

   ▶ Answer

```
sudo iptables -A INPUT -s 192.168.1.100 -j DROP
```

4. **How do you test bidirectional bandwidth with iperf3?**

   ▶ Answer

```
iperf3 -c server_ip --bidir
```

Hands-On Challenges

Challenge 1: Network Mapping

```
# Map your local network
# Identify all active hosts and their open ports
# Create a network diagram with services
```

Challenge 2: Traffic Analysis

```
# Capture network traffic for 5 minutes
# Analyze the most active connections
# Identify the protocols being used
```

Challenge 3: Security Assessment

```
# Perform a security scan of a test system
# Identify potential vulnerabilities
# Create a basic security report
```

🚀 Next Steps

Excellent! You've mastered advanced network tools and techniques. You can now:

- Perform comprehensive network discovery and mapping
- Capture and analyze network traffic
- Manage firewalls and security policies
- Test network performance and diagnose issues
- Conduct basic security assessments

Ready for real-world applications? Continue to 11-dns-deep-dive.md to master DNS configuration, troubleshooting, and security.

> **Pro Tip**: Always get proper authorization before scanning networks or systems you don't own. Create a lab environment for practicing these tools safely. Consider setting up virtual machines or containers to practice advanced techniques without affecting production systems! 🔒

🌐 DNS Deep Dive: Domain Name System Mastery

> **Master DNS configuration, troubleshooting, security, and advanced DNS concepts**

📖 What You'll Learn

DNS is the backbone of the internet, translating human-readable domain names into IP addresses. This chapter provides comprehensive coverage of DNS from basic concepts to advanced administration:

- DNS hierarchy and record types
- DNS server configuration and management
- Advanced DNS troubleshooting techniques
- DNS security (DNSSEC, DoH, DoT)
- DNS performance optimization
- Common DNS attacks and mitigation
- Setting up your own DNS server

🌍 Why This Matters

Critical applications:

- **Web Services**: Every website depends on DNS resolution
- **Email**: MX records route email to correct servers
- **Security**: DNS filtering and threat protection
- **Performance**: DNS optimization affects user experience
- **Infrastructure**: Service discovery and load balancing
- **Troubleshooting**: DNS issues cause 80% of connectivity problems

🏗️ DNS Hierarchy and Architecture

Understanding DNS Structure

```
# DNS hierarchy visualization
#                    . (root)
#                   / | \
#                 com org net
#                /   |   \
#           google  |    cloudflare
#           /   \   |    /
#         www   mail |  1.1.1.1
#              /    |
#           gmail   |
#                wikipedia

# Full Qualified Domain Name (FQDN) breakdown:
# www.google.com.
# |   |      |  |
# |   |      |  +-- Root (implicit)
# |   |      +-- Top Level Domain (TLD)
# |   +-- Second Level Domain (SLD)
# +-- Subdomain/Host
```

DNS Record Types

```
# A Record - IPv4 address
$ dig A google.com
google.com.     300 IN  A   142.250.191.14

# AAAA Record - IPv6 address
$ dig AAAA google.com
google.com.     300 IN  AAAA    2607:f8b0:4004:c1b::65

# CNAME Record - Canonical name (alias)
$ dig CNAME www.github.com
www.github.com.     3600    IN  CNAME   github.com.

# MX Record - Mail exchange
$ dig MX google.com
google.com.     600 IN  MX  10 smtp.google.com.
google.com.     600 IN  MX  20 smtp2.google.com.
google.com.     600 IN  MX  30 smtp3.google.com.

# NS Record - Name server
$ dig NS google.com
google.com.     172800  IN  NS  ns1.google.com.
google.com.     172800  IN  NS  ns2.google.com.

# TXT Record - Text information
$ dig TXT google.com
google.com.     300 IN  TXT "v=spf1 include:_spf.google.com ~all"
google.com.     300 IN  TXT "google-site-verification=..."

# PTR Record - Reverse DNS
$ dig -x 8.8.8.8
8.8.8.8.in-addr.arpa.   86400   IN  PTR dns.google.
```

```
# SOA Record - Start of Authority
$ dig SOA google.com
google.com.     60   IN   SOA ns1.google.com. dns-admin.google.com. 2023121501 7200
3600 1209600 3600

# SRV Record - Service location
$ dig SRV _sip._tcp.example.com
_sip._tcp.example.com.  300 IN  SRV 10 5 5060 sip.example.com.
```

DNS Resolution Process

```
# Step-by-step DNS resolution for www.google.com

# 1. Check local cache
$ systemd-resolve --status | grep -A 20 "DNS Servers"

# 2. Query recursive resolver
$ dig +trace www.google.com

; <<>> DiG 9.16.1-Ubuntu <<>> +trace www.google.com
;; global options: +cmd
.           518400  IN  NS  a.root-servers.net.
.           518400  IN  NS  b.root-servers.net.
;; Received 228 bytes from 8.8.8.8#53(8.8.8.8) in 12 ms

com.            172800  IN  NS  a.gtld-servers.net.
com.            172800  IN  NS  b.gtld-servers.net.
;; Received 1173 bytes from 198.41.0.4#53(a.root-servers.net) in 89 ms

google.com.     172800  IN  NS  ns1.google.com.
google.com.     172800  IN  NS  ns2.google.com.
;; Received 839 bytes from 192.5.6.30#53(a.gtld-servers.net) in 45 ms

www.google.com.     300 IN  A   142.250.191.147
;; Received 62 bytes from 216.239.32.10#53(ns1.google.com) in 23 ms

# 3. Understanding the trace:
# Root servers (.) -> TLD servers (.com) -> Authoritative servers (google.com)
```

🔧 DNS Server Configuration

Setting up BIND9 DNS Server

```
# Install BIND9
$ sudo apt update
$ sudo apt install bind9 bind9utils bind9-doc

# Main configuration file
```

```
$ sudo nano /etc/bind/named.conf.local

# Add zone configuration
zone "example.local" {
    type master;
    file "/etc/bind/db.example.local";
};

zone "1.168.192.in-addr.arpa" {
    type master;
    file "/etc/bind/db.192.168.1";
};

# Create forward zone file
$ sudo nano /etc/bind/db.example.local

;
; BIND data file for example.local
;
$TTL    604800
@       IN      SOA     ns1.example.local. admin.example.local. (
                              2023121501         ; Serial
                                 604800         ; Refresh
                                  86400         ; Retry
                                2419200         ; Expire
                                 604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.example.local.
@       IN      A       192.168.1.10
ns1     IN      A       192.168.1.10
www     IN      A       192.168.1.20
mail    IN      A       192.168.1.30
ftp     IN      A       192.168.1.40
@       IN      MX  10  mail.example.local.

# Create reverse zone file
$ sudo nano /etc/bind/db.192.168.1

;
; BIND reverse data file for 192.168.1.x
;
$TTL    604800
@       IN      SOA     ns1.example.local. admin.example.local. (
                              2023121501         ; Serial
                                 604800         ; Refresh
                                  86400         ; Retry
                                2419200         ; Expire
                                 604800 )       ; Negative Cache TTL
;
@       IN      NS      ns1.example.local.
10      IN      PTR     ns1.example.local.
20      IN      PTR     www.example.local.
30      IN      PTR     mail.example.local.
40      IN      PTR     ftp.example.local.
```

```
# Check configuration
$ sudo named-checkconf
$ sudo named-checkzone example.local /etc/bind/db.example.local
$ sudo named-checkzone 1.168.192.in-addr.arpa /etc/bind/db.192.168.1

# Restart BIND9
$ sudo systemctl restart bind9
$ sudo systemctl enable bind9

# Test the DNS server
$ dig @localhost example.local
$ dig @localhost www.example.local
$ dig @localhost -x 192.168.1.20
```

DNS Forwarders and Caching

```
# Configure DNS forwarders
$ sudo nano /etc/bind/named.conf.options

options {
        directory "/var/cache/bind";

        // Forwarders
        forwarders {
                8.8.8.8;
                1.1.1.1;
        };

        // Forward only (don't do recursive queries)
        forward only;

        // Allow queries from local network
        allow-query { localhost; 192.168.1.0/24; };

        // Allow recursion for local network
        allow-recursion { localhost; 192.168.1.0/24; };

        // DNS security
        dnssec-validation auto;

        // Listen on specific interfaces
        listen-on { 127.0.0.1; 192.168.1.10; };
        listen-on-v6 { ::1; };
};

# View DNS cache
$ sudo rndc dumpdb -cache
$ sudo cat /var/cache/bind/named_dump.db | grep google.com

# Clear DNS cache
```

```
$ sudo rndc flush

# Reload configuration
$ sudo rndc reload

# View DNS statistics
$ sudo rndc stats
$ sudo cat /var/cache/bind/named.stats
```

DNS Load Balancing

```
# Round-robin DNS (multiple A records)
$ sudo nano /etc/bind/db.example.local

www     IN      A       192.168.1.20
www     IN      A       192.168.1.21
www     IN      A       192.168.1.22

# Test round-robin
$ for i in {1..6}; do dig +short www.example.local; done
192.168.1.20
192.168.1.21
192.168.1.22
192.168.1.20
192.168.1.21
192.168.1.22

# Weighted records (using SRV)
$ sudo nano /etc/bind/db.example.local

_http._tcp.www  IN  SRV  10  60  80  server1.example.local.
_http._tcp.www  IN  SRV  10  30  80  server2.example.local.
_http._tcp.www  IN  SRV  10  10  80  server3.example.local.

# Geographic DNS (views)
$ sudo nano /etc/bind/named.conf.local

acl "us-clients" {
    192.168.1.0/24;
};

acl "eu-clients" {
    10.0.0.0/8;
};

view "us" {
    match-clients { "us-clients"; };
    zone "example.com" {
        type master;
        file "/etc/bind/db.example.com.us";
    };
};

view "eu" {
    match-clients { "eu-clients"; };
    zone "example.com" {
        type master;
        file "/etc/bind/db.example.com.eu";
    };
};
```

🔍 Advanced DNS Troubleshooting

Comprehensive DNS Diagnostics

```bash
#!/bin/bash
# dns-troubleshoot.sh - Complete DNS diagnostic tool

DOMAIN="$1"
if [ -z "$DOMAIN" ]; then
    echo "Usage: $0 <domain>"
    exit 1
fi

echo "=== DNS Troubleshooting for $DOMAIN ==="
echo

# 1. Basic resolution test
echo "1. Basic Resolution:"
dig +short "$DOMAIN" || echo "❌ Resolution failed"
echo

# 2. Authoritative servers
echo "2. Authoritative Name Servers:"
dig +short NS "$DOMAIN"
echo

# 3. SOA record analysis
echo "3. SOA Record:"
dig +short SOA "$DOMAIN"
echo

# 4. TTL analysis
echo "4. TTL Values:"
dig "$DOMAIN" | grep -E "^$DOMAIN.*IN.*A" | awk '{print "A record TTL: " $2}'
echo

# 5. Test multiple DNS servers
echo "5. Resolution from Different Servers:"
for server in 8.8.8.8 1.1.1.1 208.67.222.222; do
    echo -n "$server: "
    dig @"$server" +short "$DOMAIN" | head -1 || echo "Failed"
```

```
done
echo

# 6. Reverse DNS
echo "6. Reverse DNS:"
IP=$(dig +short "$DOMAIN" | head -1)
if [ ! -z "$IP" ]; then
    dig +short -x "$IP" || echo "No reverse DNS"
else
    echo "No IP to reverse lookup"
fi
echo

# 7. DNS propagation check
echo "7. DNS Propagation (Root Servers):"
for root in a.root-servers.net b.root-servers.net c.root-servers.net; do
    echo -n "$root: "
    timeout 5 dig @"$root" "$DOMAIN" +short 2>/dev/null | head -1 || echo
"Timeout/No response"
done
echo

echo "=== Troubleshooting Complete ==="
```

DNS Performance Analysis

```
# DNS response time testing
$ dig +stats google.com | grep "Query time"
;; Query time: 12 msec

# Batch DNS performance test
#!/bin/bash
# dns-performance.sh

DOMAINS=("google.com" "facebook.com" "amazon.com" "microsoft.com" "apple.com")
SERVERS=("8.8.8.8" "1.1.1.1" "208.67.222.222" "9.9.9.9")

echo "DNS Performance Test"
echo "==================="
printf "%-15s" "Server"
for domain in "${DOMAINS[@]}"; do
    printf "%-15s" "$domain"
done
echo

for server in "${SERVERS[@]}"; do
    printf "%-15s" "$server"
    for domain in "${DOMAINS[@]}"; do
        time=$(dig @"$server" "$domain" | grep "Query time" | awk '{print $4}')
        printf "%-15s" "${time}ms"
    done
    echo
done

# DNS cache analysis
$ systemd-resolve --statistics
DNSSEC supported by current servers: no
Transactions
Current Transactions: 0
Total Transactions: 12345

Cache
Current Cache Size: 150
Cache Hits: 8901
Cache Misses: 3444

# Clear system DNS cache
$ sudo systemd-resolve --flush-caches
$ sudo systemctl restart systemd-resolved
```

DNS Debugging Tools

```
# Verbose dig output
$ dig +trace +additional +answer google.com

# Show all record types
$ dig ANY google.com

# Query specific DNS server with debugging
$ dig @8.8.8.8 +debug google.com

# DNS over HTTPS testing
$ curl -H "accept: application/dns-json" "https://cloudflare-dns.com/dns-query?
name=google.com&type=A"

# DNS over TLS testing
$ kdig @1.1.1.1 +tls google.com

# Monitor DNS queries in real-time
$ sudo tcpdump -i any -n port 53

# DNS query logging with named
$ sudo nano /etc/bind/named.conf.local

logging {
    channel query_log {
        file "/var/log/bind/query.log" versions 3 size 5m;
        severity info;
        print-category yes;
        print-severity yes;
        print-time yes;
    };
    category queries { query_log; };
};

# View DNS query logs
$ sudo tail -f /var/log/bind/query.log
```

🔒 DNS Security

DNSSEC Implementation

```
# Generate DNSSEC keys
$ sudo dnssec-keygen -a RSASHA256 -b 2048 -n ZONE example.local
$ sudo dnssec-keygen -a RSASHA256 -b 2048 -n ZONE -f KSK example.local

# Sign the zone
$ sudo dnssec-signzone -o example.local -k Kexample.local.+008+12345.key
db.example.local Kexample.local.+008+54321.key

# Update BIND configuration for DNSSEC
$ sudo nano /etc/bind/named.conf.local

zone "example.local" {
    type master;
    file "/etc/bind/db.example.local.signed";
    key-directory "/etc/bind";
    auto-dnssec maintain;
    inline-signing yes;
};

# Verify DNSSEC
$ dig +dnssec example.local
$ dig +dnssec +multiline example.local

# Check DNSSEC validation
$ dig @8.8.8.8 +dnssec google.com | grep -E "ad|RRSIG"
```

DNS Filtering and Security

```
# Configure DNS filtering with Pi-hole
$ curl -sSL https://install.pi-hole.net | bash

# Manual DNS blacklist configuration
$ sudo nano /etc/bind/named.conf.local

# Blackhole zone for malicious domains
zone "malicious.com" {
    type master;
    file "/etc/bind/db.blackhole";
};
```

```
# Create blackhole zone file
$ sudo nano /etc/bind/db.blackhole

$TTL 86400
@   IN  SOA localhost. root.localhost. (
        2023121501
        3600
        1800
        604800
        86400 )
@   IN  NS  localhost.
@   IN  A   127.0.0.1
*   IN  A   127.0.0.1

# DNS over HTTPS (DoH) configuration
$ sudo nano /etc/systemd/resolved.conf

[Resolve]
DNS=1.1.1.1#cloudflare-dns.com 8.8.8.8#dns.google
DNSOverTLS=yes
DNSSEC=yes

# Test DoH
$ systemd-resolve --status | grep -A 5 "DNS Servers"

# DNS monitoring for security
$ sudo tcpdump -i any -n 'port 53 and host not 127.0.0.1' | grep -E
'suspicious|malware|phishing'
```

DNS Attack Detection

```
# DNS amplification attack detection
$ sudo netstat -nu | awk '/^udp/ && $4 ~ /:53$/ {print $2}' | sort -n

# Monitor DNS query patterns
#!/bin/bash
# dns-monitor.sh - Detect suspicious DNS activity

LOGFILE="/var/log/bind/query.log"
THRESHOLD=100

echo "DNS Security Monitor"
echo "==================="

# Check for high query volume from single IP
echo "High Volume Queries (last hour):"
tail -n 10000 "$LOGFILE" | grep "$(date '+%d-%b-%Y %H')" | \
    awk '{print $7}' | sort | uniq -c | sort -nr | head -10

# Check for suspicious domains
```

```
echo "\nSuspicious Domain Queries:"
tail -n 10000 "$LOGFILE" | grep -E '(bit\.ly|tinyurl|suspicious)' | \
    awk '{print $9}' | sort | uniq -c | sort -nr

# Check for DNS tunneling indicators
echo "\nPotential DNS Tunneling:"
tail -n 10000 "$LOGFILE" | grep -E 'TXT|NULL' | \
    awk 'length($9) > 50 {print $7, $9}' | head -10

echo "\n=== Monitor Complete ==="
```

🚀 DNS Performance Optimization

Caching Strategies

```
# Optimize BIND cache settings
$ sudo nano /etc/bind/named.conf.options

options {
    // Increase cache size
    max-cache-size 512M;

    // Optimize cache TTL
    max-cache-ttl 86400;        // 1 day
    max-ncache-ttl 3600;        // 1 hour for negative cache

    // Prefetch expiring records
    prefetch 2 9;

    // Optimize recursion
    recursive-clients 10000;
    tcp-clients 1000;

    // Rate limiting
    rate-limit {
        responses-per-second 20;
        window 5;
    };
};

# Monitor cache performance
$ sudo rndc stats
$ grep -E "cache|queries" /var/cache/bind/named.stats

# DNS cache warming script
#!/bin/bash
# dns-warm.sh - Warm DNS cache with popular domains

POPULAR_DOMAINS=(
    "google.com" "youtube.com" "facebook.com" "amazon.com"
    "wikipedia.org" "twitter.com" "instagram.com" "linkedin.com"
    "netflix.com" "microsoft.com" "apple.com" "github.com"
)

echo "Warming DNS cache..."
for domain in "${POPULAR_DOMAINS[@]}"; do
    dig "$domain" > /dev/null 2>&1
    echo "Cached: $domain"
done
echo "Cache warming complete!"
```

DNS Load Testing

```
# Install dnsperf for load testing
$ sudo apt install dnsperf

# Create query file
$ cat > queries.txt << EOF
google.com A
facebook.com A
amazon.com A
microsoft.com A
apple.com A
EOF

# Run DNS performance test
$ dnsperf -s 8.8.8.8 -d queries.txt -l 30

DNS Performance Testing Tool
Version 2.3.2

[Status] Command line: dnsperf -s 8.8.8.8 -d queries.txt -l 30
[Status] Sending queries (to 8.8.8.8:53)
[Status] Started at: Mon Dec 15 10:30:00 2023
[Status] Stopping after 30.000000 seconds
[Timeout] Query timed out: msg id 1234
[Status] Testing complete (time limit)

Statistics:

  Queries sent:         1500
  Queries completed:    1485 (99.00%)
  Queries lost:         15 (1.00%)

  Response codes:       NOERROR 1485 (100.00%)
  Average packet size:  request 28, response 44
  Run time (s):         30.000000
  Queries per second:   49.500000

  Average Latency (s):  0.012345 (min 0.001234, max 0.098765)
  Latency StdDev (s):   0.005678
```

```
# Concurrent DNS testing
$ for i in {1..10}; do
    (dnsperf -s 8.8.8.8 -d queries.txt -l 10 &)
done
wait
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between recursive and iterative DNS queries?**

   ▶ Answer

   Recursive: DNS server does all the work, querying other servers until it gets the final answer. Iterative: DNS server returns referrals, and the client must query each server in the chain.

2. **What does a TTL of 300 seconds mean in a DNS record?**

   ▶ Answer

   The record can be cached for 300 seconds (5 minutes) before it should be refreshed from the authoritative server.

3. **How do you check if DNSSEC is working for a domain?**

   ▶ Answer

   ```
   dig +dnssec domain.com
   # Look for RRSIG records and the 'ad' (authenticated data) flag
   ```

4. **What's the purpose of an MX record's priority number?**

   ▶ Answer

   Lower numbers have higher priority. Mail servers try the lowest priority first, using higher priority servers as backups.

Hands-On Challenges

Challenge 1: DNS Server Setup

```
# Set up a complete DNS server for a test domain
# Include forward and reverse zones
# Test all record types
```

Challenge 2: DNS Security Implementation

```
# Implement DNSSEC for your test domain
# Set up DNS filtering
# Monitor for suspicious activity
```

Challenge 3: DNS Performance Optimization

```
# Optimize DNS server performance
# Implement caching strategies
# Conduct load testing
```

🚀 Next Steps

Excellent! You've mastered DNS from basic concepts to advanced administration. You can now:

- Configure and manage DNS servers
- Troubleshoot complex DNS issues
- Implement DNS security measures
- Optimize DNS performance
- Detect and prevent DNS attacks

Ready for web technologies? Continue to 12-web-servers-nginx.md to learn web server configuration and management.

> **Pro Tip**: DNS changes can take time to propagate globally (up to 48 hours). Always test DNS changes thoroughly in a lab environment first. Keep DNS records simple and well-documented - complex DNS setups are harder to troubleshoot! 🌐

🛡️ Security & Firewalls: Network Protection and Intrusion Detection

> **Master network security, firewall configuration, and intrusion detection systems**

📖 What You'll Learn

Network security is critical for protecting systems and data. This chapter covers comprehensive security measures from basic firewalls to advanced intrusion detection:

- Firewall fundamentals and types
- iptables and UFW configuration
- Network security scanning and assessment
- Intrusion detection and prevention systems
- Log analysis and security monitoring
- VPN setup and configuration
- Security hardening best practices

- Incident response and forensics basics

🌍 Why This Matters

Critical applications:

- **System Protection**: Defend against network attacks and intrusions
- **Compliance**: Meet security standards and regulations
- **Data Security**: Protect sensitive information in transit
- **Network Monitoring**: Detect and respond to security threats
- **Access Control**: Manage who can access what resources

🔥 Firewall Fundamentals

Understanding Firewalls

```
# Check current firewall status
$ sudo ufw status
Status: inactive

$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

# Check if firewall is running
$ sudo systemctl status ufw
$ sudo systemctl status iptables

# View network connections
$ sudo netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN

# Modern alternative with ss
$ sudo ss -tuln
Netid  State   Recv-Q  Send-Q    Local Address:Port    Peer Address:Port
tcp    LISTEN  0       128             0.0.0.0:22           0.0.0.0:*
tcp    LISTEN  0       80            127.0.0.1:3306         0.0.0.0:*
tcp    LISTEN  0       128                [::]:22              [::]:*
tcp    LISTEN  0       128                [::]:80              [::]:*
```

UFW (Uncomplicated Firewall)

```
# Enable UFW
$ sudo ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup

# Check status
$ sudo ufw status verbose
Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), disabled (routed)
New profiles: skip

# Basic rules
$ sudo ufw default deny incoming
$ sudo ufw default allow outgoing
$ sudo ufw default deny forward

# Allow specific services
$ sudo ufw allow ssh
$ sudo ufw allow 22/tcp
$ sudo ufw allow 80/tcp
$ sudo ufw allow 443/tcp
$ sudo ufw allow 53/udp

# Allow from specific IP
$ sudo ufw allow from 192.168.1.100
$ sudo ufw allow from 192.168.1.0/24
$ sudo ufw allow from 192.168.1.100 to any port 22

# Allow specific protocols
$ sudo ufw allow out 53/udp
$ sudo ufw allow in on eth0 to any port 80

# Deny rules
$ sudo ufw deny 23/tcp
$ sudo ufw deny from 10.0.0.0/8

# Delete rules
$ sudo ufw status numbered
Status: active

     To                         Action      From
     --                         ------      ----
[ 1] 22/tcp                     ALLOW IN    Anywhere
[ 2] 80/tcp                     ALLOW IN    Anywhere
[ 3] 443/tcp                    ALLOW IN    Anywhere

$ sudo ufw delete 2
$ sudo ufw delete allow 80/tcp
```

```
# Advanced rules
$ sudo ufw limit ssh                    # Rate limiting
$ sudo ufw allow from 192.168.1.0/24 to any app OpenSSH

# Application profiles
$ sudo ufw app list
Available applications:
  Apache
  Apache Full
  Apache Secure
  OpenSSH
  Postfix
  Postfix SMTPS
  Postfix Submission

$ sudo ufw allow 'Apache Full'
$ sudo ufw app info 'Apache Full'
Profile: Apache Full
Title: Web Server (HTTP,HTTPS)
Description: Apache v2 is the next generation of the omnipresent Apache web
server.

Ports:
  80,443/tcp

# Logging
$ sudo ufw logging on
$ sudo ufw logging medium
$ sudo tail -f /var/log/ufw.log

# Reset UFW
$ sudo ufw --force reset
```

iptables Advanced Configuration

```
# View current rules
$ sudo iptables -L -n -v
Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

# Save current rules
$ sudo iptables-save > /tmp/iptables-backup.rules

# Basic iptables rules
# Allow loopback
$ sudo iptables -A INPUT -i lo -j ACCEPT
$ sudo iptables -A OUTPUT -o lo -j ACCEPT

# Allow established connections
$ sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
```

```
# Allow SSH
$ sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow HTTP and HTTPS
$ sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow DNS
$ sudo iptables -A INPUT -p udp --dport 53 -j ACCEPT
$ sudo iptables -A INPUT -p tcp --dport 53 -j ACCEPT

# Allow from specific network
$ sudo iptables -A INPUT -s 192.168.1.0/24 -j ACCEPT

# Rate limiting SSH
$ sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --
set
$ sudo iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --
update --seconds 60 --hitcount 4 -j DROP

# Drop invalid packets
$ sudo iptables -A INPUT -m conntrack --ctstate INVALID -j DROP

# Log dropped packets
$ sudo iptables -A INPUT -j LOG --log-prefix "iptables-dropped: "

# Default policies
$ sudo iptables -P INPUT DROP
$ sudo iptables -P FORWARD DROP
$ sudo iptables -P OUTPUT ACCEPT

# Save rules (Ubuntu/Debian)
$ sudo apt install iptables-persistent
$ sudo netfilter-persistent save
$ sudo netfilter-persistent reload

# Advanced iptables script
#!/bin/bash
# /etc/iptables/firewall.sh

# Flush existing rules
iptables -F
iptables -X
iptables -t nat -F
iptables -t nat -X
iptables -t mangle -F
iptables -t mangle -X

# Default policies
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -P OUTPUT ACCEPT

# Allow loopback
```

```
iptables -A INPUT -i lo -j ACCEPT
iptables -A OUTPUT -o lo -j ACCEPT

# Allow established connections
iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT

# Allow SSH with rate limiting
iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --set --
name SSH
iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -m recent --update
--seconds 60 --hitcount 4 --name SSH -j DROP
iptables -A INPUT -p tcp --dport 22 -j ACCEPT

# Allow web services
iptables -A INPUT -p tcp --dport 80 -j ACCEPT
iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow DNS
iptables -A INPUT -p udp --dport 53 -j ACCEPT
iptables -A INPUT -p tcp --dport 53 -j ACCEPT

# Allow ICMP (ping)
iptables -A INPUT -p icmp --icmp-type echo-request -j ACCEPT

# Log and drop everything else
iptables -A INPUT -j LOG --log-prefix "iptables-dropped: " --log-level 4
iptables -A INPUT -j DROP

echo "Firewall rules applied"
```

🔍 Network Security Scanning

Nmap Security Scanning

```
# Basic host discovery
$ nmap -sn 192.168.1.0/24
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for 192.168.1.1
Host is up (0.001s latency).
Nmap scan report for 192.168.1.100
Host is up (0.002s latency).
Nmap scan report for 192.168.1.150
Host is up (0.001s latency).
Nmap done: 256 IP addresses (3 hosts up) scanned in 2.5 seconds

# Port scanning
$ nmap -sS 192.168.1.100          # SYN scan
$ nmap -sT 192.168.1.100          # TCP connect scan
$ nmap -sU 192.168.1.100          # UDP scan
$ nmap -sA 192.168.1.100          # ACK scan
```

```
# Comprehensive scan
$ nmap -A 192.168.1.100
Starting Nmap 7.80 ( https://nmap.org )
Nmap scan report for server.local (192.168.1.100)
Host is up (0.001s latency).
Not shown: 996 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.5
80/tcp   open  http    Apache httpd 2.4.41
443/tcp  open  https   Apache httpd 2.4.41
3306/tcp open  mysql   MySQL 8.0.28-0ubuntu0.20.04.3

# Vulnerability scanning
$ nmap --script vuln 192.168.1.100
$ nmap --script ssl-enum-ciphers -p 443 192.168.1.100

# OS detection
$ sudo nmap -O 192.168.1.100

# Service version detection
$ nmap -sV 192.168.1.100

# Stealth scanning
$ sudo nmap -sS -T2 192.168.1.100  # Slow and stealthy
$ sudo nmap -f 192.168.1.100       # Fragment packets

# Scan specific ports
$ nmap -p 22,80,443 192.168.1.100
$ nmap -p 1-1000 192.168.1.100
$ nmap -p- 192.168.1.100           # All ports

# Output formats
$ nmap -oN scan_results.txt 192.168.1.100
$ nmap -oX scan_results.xml 192.168.1.100
$ nmap -oG scan_results.gnmap 192.168.1.100
$ nmap -oA scan_results 192.168.1.100  # All formats
```

Security Assessment Tools

```
# Nikto web vulnerability scanner
$ sudo apt install nikto
$ nikto -h http://192.168.1.100
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.100
+ Target Hostname:    192.168.1.100
+ Target Port:        80
+ Start Time:         2023-12-15 10:30:00 (GMT0)
---------------------------------------------------------------------------
+ Server: Apache/2.4.41 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined.
+ The X-Content-Type-Options header is not set.

# OpenVAS vulnerability scanner
$ sudo apt install openvas
$ sudo gvm-setup
$ sudo gvm-start
# Access web interface at https://localhost:9392

# Lynis system auditing
$ sudo apt install lynis
$ sudo lynis audit system

# Chkrootkit rootkit detection
$ sudo apt install chkrootkit
$ sudo chkrootkit

# rkhunter rootkit detection
$ sudo apt install rkhunter
$ sudo rkhunter --update
$ sudo rkhunter --check

# ClamAV antivirus
$ sudo apt install clamav clamav-daemon
$ sudo freshclam                    # Update virus definitions
$ sudo clamscan -r /home            # Scan directory
$ sudo clamscan -r --infected /     # Scan for infected files only
```

🚨 Intrusion Detection Systems

Fail2ban Configuration

```
# Install Fail2ban
$ sudo apt install fail2ban

# Check status
$ sudo systemctl status fail2ban
$ sudo fail2ban-client status
Status
|- Number of jail:      1
`- Jail list:   sshd

# Configuration files
/etc/fail2ban/
├── jail.conf         # Default configuration (don't edit)
├── jail.local        # Local overrides
├── filter.d/         # Filter definitions
└── action.d/         # Action definitions

# Create local configuration
$ sudo nano /etc/fail2ban/jail.local
```

```
[DEFAULT]
# Ban time in seconds (10 minutes)
bantime = 600

# Find time window (10 minutes)
findtime = 600

# Number of failures before ban
maxretry = 3

# Ignore local IPs
ignoreip = 127.0.0.1/8 192.168.1.0/24

# Email notifications
destemail = admin@example.com
sender = fail2ban@example.com
mta = sendmail
action = %(action_mwl)s

[sshd]
enabled = true
port = ssh
filter = sshd
logpath = /var/log/auth.log
maxretry = 3
bantime = 3600

[apache-auth]
enabled = true
port = http,https
filter = apache-auth
logpath = /var/log/apache2/error.log
maxretry = 3

[apache-badbots]
enabled = true
port = http,https
filter = apache-badbots
logpath = /var/log/apache2/access.log
maxretry = 2

[nginx-http-auth]
enabled = true
filter = nginx-http-auth
logpath = /var/log/nginx/error.log
maxretry = 3

# Restart Fail2ban
$ sudo systemctl restart fail2ban

# Check jail status
$ sudo fail2ban-client status sshd
Status for the jail: sshd
```

```
|- Filter
|  |- Currently failed: 0
|  |- Total failed:     5
|  `- File list:        /var/log/auth.log
`- Actions
   |- Currently banned: 1
   |- Total banned:     2
   `- Banned IP list:   192.168.1.200

# Unban IP
$ sudo fail2ban-client set sshd unbanip 192.168.1.200

# Ban IP manually
$ sudo fail2ban-client set sshd banip 192.168.1.200

# Custom filter example
$ sudo nano /etc/fail2ban/filter.d/custom-app.conf

[Definition]
failregex = ^.*Failed login attempt from <HOST>.*$
ignoreregex =

# Test filter
$ fail2ban-regex /var/log/custom-app.log /etc/fail2ban/filter.d/custom-app.conf
```

OSSEC Host-based IDS

```
# Install OSSEC
$ wget https://github.com/ossec/ossec-hids/archive/3.7.0.tar.gz
$ tar -xzf 3.7.0.tar.gz
$ cd ossec-hids-3.7.0
$ sudo ./install.sh

# Configuration
$ sudo nano /var/ossec/etc/ossec.conf

<ossec_config>
  <global>
    <email_notification>yes</email_notification>
    <email_to>admin@example.com</email_to>
    <smtp_server>localhost</smtp_server>
    <email_from>ossec@example.com</email_from>
  </global>

  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
    <include>telnetd_rules.xml</include>
    <include>syslog_rules.xml</include>
    <include>arpwatch_rules.xml</include>
    <include>symantec-av_rules.xml</include>
    <include>symantec-ws_rules.xml</include>
    <include>pix_rules.xml</include>
    <include>named_rules.xml</include>
    <include>smbd_rules.xml</include>
    <include>vsftpd_rules.xml</include>
    <include>pure-ftpd_rules.xml</include>
    <include>proftpd_rules.xml</include>
    <include>ms_ftpd_rules.xml</include>
    <include>ftpd_rules.xml</include>
    <include>hordeimp_rules.xml</include>
    <include>roundcube_rules.xml</include>
    <include>wordpress_rules.xml</include>
    <include>cimserver_rules.xml</include>
    <include>vpopmail_rules.xml</include>
    <include>vmpop3d_rules.xml</include>
    <include>courier_rules.xml</include>
    <include>web_rules.xml</include>
    <include>web_appsec_rules.xml</include>
    <include>apache_rules.xml</include>
    <include>nginx_rules.xml</include>
    <include>php_rules.xml</include>
    <include>mysql_rules.xml</include>
    <include>postgresql_rules.xml</include>
    <include>ids_rules.xml</include>
    <include>squid_rules.xml</include>
    <include>firewall_rules.xml</include>
    <include>cisco-ios_rules.xml</include>
    <include>netscreenfw_rules.xml</include>
    <include>sonicwall_rules.xml</include>
    <include>postfix_rules.xml</include>
    <include>sendmail_rules.xml</include>
    <include>imapd_rules.xml</include>
    <include>mailscanner_rules.xml</include>
    <include>dovecot_rules.xml</include>
    <include>ms-exchange_rules.xml</include>
    <include>racoon_rules.xml</include>
    <include>vpn_concentrator_rules.xml</include>
    <include>spamd_rules.xml</include>
    <include>msauth_rules.xml</include>
    <include>mcafee_av_rules.xml</include>
    <include>trend-osce_rules.xml</include>
    <include>ms-se_rules.xml</include>
    <include>zeus_rules.xml</include>
    <include>solaris_bsm_rules.xml</include>
    <include>vmware_rules.xml</include>
    <include>ms_dhcp_rules.xml</include>
    <include>asterisk_rules.xml</include>
    <include>ossec_rules.xml</include>
    <include>attack_rules.xml</include>
    <include>local_rules.xml</include>
  </rules>

  <syscheck>
```

```
    <!-- Frequency that syscheck is executed -- default every 22 hours -->
    <frequency>79200</frequency>

    <!-- Directories to check  (perform all possible verifications) -->
    <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin,/boot</directories>

    <!-- Files/directories to ignore -->
    <ignore>/etc/mtab</ignore>
    <ignore>/etc/hosts.deny</ignore>
    <ignore>/etc/mail/statistics</ignore>
    <ignore>/etc/random-seed</ignore>
    <ignore>/etc/random.seed</ignore>
    <ignore>/etc/adjtime</ignore>
    <ignore>/etc/httpd/logs</ignore>
    <ignore>/etc/utmpx</ignore>
    <ignore>/etc/wtmpx</ignore>
    <ignore>/etc/cups/certs</ignore>
    <ignore>/etc/dumpdates</ignore>
    <ignore>/etc/svc/volatile</ignore>
  </syscheck>

  <rootcheck>
    <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
    <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
    <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>
    <system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</system_audit>
    <system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</system_audit>
    <system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</system_audit>
  </rootcheck>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/messages</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/auth.log</location>
  </localfile>

  <localfile>
    <log_format>syslog</log_format>
    <location>/var/log/syslog</location>
  </localfile>

  <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/access.log</location>
  </localfile>

  <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/error.log</location>
  </localfile>
</ossec_config>

# Start OSSEC
$ sudo /var/ossec/bin/ossec-control start

# Check status
$ sudo /var/ossec/bin/ossec-control status
ossec-monitord is running...
ossec-logcollector is running...
ossec-syscheckd is running...
ossec-analysisd is running...
ossec-maild is running...
ossec-execd is running...

# View alerts
$ sudo tail -f /var/ossec/logs/alerts/alerts.log
```

📊 Log Analysis and Security Monitoring

Centralized Logging with rsyslog

```
# Configure rsyslog server
$ sudo nano /etc/rsyslog.conf

# Uncomment these lines to enable UDP syslog reception
$ModLoad imudp
$UDPServerRun 514
$UDPServerAddress 0.0.0.0

# Uncomment these lines to enable TCP syslog reception
$ModLoad imtcp
$InputTCPServerRun 514

# Template for remote logs
$template RemoteLogs,"/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log"
*.* ?RemoteLogs
& stop

# Restart rsyslog
$ sudo systemctl restart rsyslog

# Configure rsyslog client
$ sudo nano /etc/rsyslog.conf

# Add at the end
*.* @@192.168.1.100:514  # TCP
*.* @192.168.1.100:514   # UDP

# Restart client rsyslog
$ sudo systemctl restart rsyslog
```

```
# Test logging
$ logger "Test message from $(hostname)"
```

Log Analysis Tools

```
# Basic log analysis with grep and awk
$ sudo grep "Failed password" /var/log/auth.log | awk '{print $11}' | sort | uniq
-c | sort -nr
     15 192.168.1.200
      8 10.0.0.50
      3 172.16.1.100

# Find successful SSH logins
$ sudo grep "Accepted password" /var/log/auth.log | awk '{print $1, $2, $3, $9,
$11}'

# Analyze Apache access logs
$ sudo awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr |
head -10
$ sudo awk '{print $7}' /var/log/apache2/access.log | sort | uniq -c | sort -nr |
head -10

# Find 404 errors
$ sudo awk '$9 == 404 {print $7}' /var/log/apache2/access.log | sort | uniq -c |
sort -nr

# GoAccess web log analyzer
$ sudo apt install goaccess
$ sudo goaccess /var/log/apache2/access.log -c
$ sudo goaccess /var/log/apache2/access.log -o /var/www/html/report.html --log-
format=COMBINED

# Logwatch for automated log analysis
$ sudo apt install logwatch
$ sudo nano /etc/logwatch/conf/logwatch.conf

LogDir = /var/log
TmpDir = /var/cache/logwatch
Output = mail
Format = html
Encode = none
MailTo = admin@example.com
MailFrom = logwatch@example.com
Subject = Logwatch Report
Detail = Med
Service = All
Range = yesterday

# Run logwatch manually
$ sudo logwatch --detail Med --mailto admin@example.com --range yesterday
```

```
# Automated logwatch via cron
$ sudo nano /etc/cron.daily/00logwatch
#!/bin/bash
/usr/sbin/logwatch --output mail --mailto admin@example.com --detail high
```

Security Information and Event Management (SIEM)

```
# ELK Stack installation (Elasticsearch, Logstash, Kibana)
# Add Elastic repository
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add
-
$ echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee
/etc/apt/sources.list.d/elastic-7.x.list
$ sudo apt update

# Install Elasticsearch
$ sudo apt install elasticsearch
$ sudo systemctl enable elasticsearch
$ sudo systemctl start elasticsearch

# Test Elasticsearch
$ curl -X GET "localhost:9200/"

# Install Logstash
$ sudo apt install logstash

# Configure Logstash
$ sudo nano /etc/logstash/conf.d/syslog.conf

input {
  file {
    path => "/var/log/syslog"
    type => "syslog"
  }
  file {
    path => "/var/log/auth.log"
    type => "auth"
  }
}

filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:timestamp} %{IPORHOST:host} %
{DATA:program}(?:\[%{POSINT:pid}\])?: %{GREEDYDATA:message}" }
    }
  }
  if [type] == "auth" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:timestamp} %{IPORHOST:host} %
{DATA:program}(?:\[%{POSINT:pid}\])?: %{GREEDYDATA:message}" }
    }
  }
}

output {
  elasticsearch {
    hosts => ["localhost:9200"]
    index => "logstash-%{+YYYY.MM.dd}"
  }
}

# Start Logstash
$ sudo systemctl enable logstash
$ sudo systemctl start logstash

# Install Kibana
$ sudo apt install kibana
$ sudo systemctl enable kibana
$ sudo systemctl start kibana

# Access Kibana at http://localhost:5601
```

🔐 VPN Configuration

OpenVPN Server Setup

```
# Install OpenVPN and Easy-RSA
$ sudo apt install openvpn easy-rsa

# Set up CA directory
$ make-cadir ~/openvpn-ca
$ cd ~/openvpn-ca

# Configure CA variables
$ nano vars

export KEY_COUNTRY="US"
export KEY_PROVINCE="CA"
export KEY_CITY="SanFrancisco"
export KEY_ORG="Example Corp"
export KEY_EMAIL="admin@example.com"
export KEY_OU="IT Department"
export KEY_NAME="server"

# Source variables and build CA
$ source vars
$ ./clean-all
$ ./build-ca

# Generate server certificate
```

```
$ ./build-key-server server

# Generate Diffie-Hellman parameters
$ ./build-dh

# Generate client certificate
$ ./build-key client1

# Copy certificates to OpenVPN directory
$ sudo cp ~/openvpn-ca/keys/{server.crt,server.key,ca.crt,dh2048.pem}
/etc/openvpn/

# Generate TLS auth key
$ sudo openvpn --genkey --secret /etc/openvpn/ta.key

# Configure OpenVPN server
$ sudo nano /etc/openvpn/server.conf

port 1194
proto udp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh2048.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
push "redirect-gateway def1 bypass-dhcp"
push "dhcp-option DNS 8.8.8.8"
push "dhcp-option DNS 8.8.4.4"
keepalive 10 120
tls-auth ta.key 0
cipher AES-256-CBC
user nobody
group nogroup
persist-key
persist-tun
status openvpn-status.log
verb 3
explicit-exit-notify 1

# Enable IP forwarding
$ sudo nano /etc/sysctl.conf
# Uncomment:
net.ipv4.ip_forward=1

$ sudo sysctl -p

# Configure firewall for VPN
$ sudo ufw allow 1194/udp
$ sudo ufw allow OpenSSH

# Add NAT rules
$ sudo nano /etc/ufw/before.rules
```

```
# Add at the top:
*nat
:POSTROUTING ACCEPT [0:0]
-A POSTROUTING -s 10.8.0.0/8 -o eth0 -j MASQUERADE
COMMIT

# Start OpenVPN
$ sudo systemctl enable openvpn@server
$ sudo systemctl start openvpn@server

# Create client configuration
$ nano ~/client1.ovpn

client
dev tun
proto udp
remote YOUR_SERVER_IP 1194
resolv-retry infinite
nobind
user nobody
group nogroup
persist-key
persist-tun
remote-cert-tls server
cipher AES-256-CBC
verb 3

<ca>
[Insert ca.crt contents]
</ca>

<cert>
[Insert client1.crt contents]
</cert>

<key>
[Insert client1.key contents]
</key>

<tls-auth>
[Insert ta.key contents]
</tls-auth>
key-direction 1
```

WireGuard VPN (Modern Alternative)

```
# Install WireGuard
$ sudo apt install wireguard

# Generate server keys
$ wg genkey | sudo tee /etc/wireguard/private.key
$ sudo chmod 600 /etc/wireguard/private.key
$ sudo cat /etc/wireguard/private.key | wg pubkey | sudo tee
/etc/wireguard/public.key

# Configure WireGuard server
$ sudo nano /etc/wireguard/wg0.conf

[Interface]
PrivateKey = [SERVER_PRIVATE_KEY]
Address = 10.0.0.1/24
ListenPort = 51820
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT;
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j
ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE

[Peer]
PublicKey = [CLIENT_PUBLIC_KEY]
AllowedIPs = 10.0.0.2/32

# Generate client keys
$ wg genkey | tee client_private.key | wg pubkey > client_public.key

# Client configuration
$ nano client.conf

[Interface]
PrivateKey = [CLIENT_PRIVATE_KEY]
Address = 10.0.0.2/24
DNS = 8.8.8.8

[Peer]
PublicKey = [SERVER_PUBLIC_KEY]
Endpoint = YOUR_SERVER_IP:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 25

# Start WireGuard
$ sudo systemctl enable wg-quick@wg0
$ sudo systemctl start wg-quick@wg0

# Check status
$ sudo wg show
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between UFW and iptables?**

   ▶ Answer

UFW (Uncomplicated Firewall) is a user-friendly frontend for iptables, making firewall management easier with simpler commands, while iptables provides more granular control but with more complex syntax.

2. **How does Fail2ban protect against brute force attacks?**

   ▶ Answer

   Fail2ban monitors log files for failed authentication attempts and automatically bans IP addresses that exceed the configured failure threshold for a specified time period.

3. **What's the purpose of a SIEM system?**

   ▶ Answer

   SIEM (Security Information and Event Management) systems collect, analyze, and correlate security events from multiple sources to provide real-time security monitoring and incident response capabilities.

4. **Why is WireGuard considered better than OpenVPN?**

   ▶ Answer

   WireGuard is simpler to configure, has better performance, uses modern cryptography, has a smaller codebase (easier to audit), and provides better battery life on mobile devices.

Hands-On Challenges

Challenge 1: Complete Security Hardening

```
# Implement comprehensive security for a server:
# - Configure advanced firewall rules with iptables
# - Set up Fail2ban with custom filters
# - Install and configure an IDS (OSSEC or Suricata)
# - Implement centralized logging
# - Set up automated security monitoring and alerting
```

Challenge 2: Network Security Assessment

```
# Perform a complete security assessment:
# - Use Nmap for network discovery and port scanning
# - Run vulnerability scans with OpenVAS or Nessus
# - Analyze web applications with Nikto and OWASP ZAP
# - Check for rootkits and malware
# - Generate a comprehensive security report
```

Challenge 3: VPN Infrastructure

```
# Set up a complete VPN infrastructure:
# - Deploy OpenVPN or WireGuard server
# - Configure multiple client access
# - Implement proper certificate management
# - Set up monitoring and logging
# - Create automated client provisioning
```

🚀 Next Steps

Excellent! You've mastered network security and firewalls. You can now:

- Configure and manage firewalls (UFW, iptables)
- Perform security assessments and vulnerability scanning
- Set up intrusion detection and prevention systems
- Implement centralized logging and SIEM solutions
- Configure VPN services for secure remote access
- Monitor and analyze security events
- Respond to security incidents effectively

Ready for performance optimization? Continue to 13-performance-monitoring.md to learn about system performance monitoring, optimization, and troubleshooting.

> **Pro Tip**: Security is a continuous process, not a one-time setup. Regularly update your systems, monitor logs, perform security assessments, and stay informed about new threats. Defense in depth - use multiple layers of security! 🛡

📊 Performance Monitoring: System Optimization and Troubleshooting

> **Master system performance analysis, monitoring tools, and optimization techniques**

📖 What You'll Learn

System performance monitoring is crucial for maintaining efficient and reliable systems. This chapter covers comprehensive performance analysis and optimization:

- System resource monitoring (CPU, Memory, Disk, Network)
- Performance monitoring tools and utilities
- Bottleneck identification and analysis
- System optimization techniques
- Automated monitoring and alerting
- Performance tuning best practices
- Capacity planning and scaling
- Troubleshooting performance issues

🌍 Why This Matters

Critical applications:

- **System Reliability**: Ensure systems run smoothly and efficiently
- **Cost Optimization**: Maximize resource utilization and reduce costs
- **User Experience**: Maintain fast response times and availability
- **Capacity Planning**: Predict and plan for future resource needs
- **Troubleshooting**: Quickly identify and resolve performance issues

🖥 System Resource Monitoring

CPU Monitoring

```
# Real-time CPU usage
$ top
top - 10:30:15 up 5 days,  2:15,  3 users,  load average: 0.45, 0.32, 0.28
Tasks: 234 total,   1 running, 233 sleeping,   0 stopped,   0 zombie
%Cpu(s):  5.2 us,  2.1 sy,  0.0 ni, 92.1 id,  0.6 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7982.1 total,   1234.5 free,   3456.7 used,   3290.9 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   4123.8 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 1234 www-data  20   0  123456  45678  12345 S  15.3   5.7   1:23.45 apache2
 5678 mysql     20   0  987654 234567  56789 S  12.1  29.4  12:34.56 mysqld

# Enhanced top with htop
$ htop
# Interactive process viewer with better visualization

# CPU usage per core
$ nproc                            # Number of CPU cores
4

$ lscpu                            # Detailed CPU information
Architecture:                     x86_64
CPU op-mode(s):                   32-bit, 64-bit
Byte Order:                       Little Endian
CPU(s):                           4
On-line CPU(s) list:              0-3
Thread(s) per core:               2
Core(s) per socket:               2
Socket(s):                        1
NUMA node(s):                     1
Vendor ID:                        GenuineIntel
CPU family:                       6
Model:                            142
Model name:                       Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
Stepping:                         10
CPU MHz:                          1800.000
CPU max MHz:                      3400.0000
CPU min MHz:                      400.0000
BogoMIPS:                         3600.00
```

```
# CPU usage statistics
$ vmstat 1 5                     # Every 1 second, 5 times
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 1  0      0 1234567  89012 3456789    0    0    12    45  123  456  5  2 92  1  0
 0  0      0 1234567  89012 3456789    0    0     0     8  134  467  3  1 96  0  0

# Detailed CPU statistics with sar
$ sar -u 1 5                     # CPU utilization
Linux 5.4.0-74-generic (hostname)      12/15/2023      _x86_64_        (4 CPU)

10:30:16 AM     CPU     %user     %nice   %system   %iowait    %steal     %idle
10:30:17 AM     all      5.25      0.00      2.13      0.63      0.00     91.99
10:30:18 AM     all      4.87      0.00      1.89      0.25      0.00     92.99

# Per-CPU statistics
$ sar -P ALL 1 3                 # All CPUs

# CPU load average
$ uptime
 10:30:15 up 5 days,  2:15,  3 users,  load average: 0.45, 0.32, 0.28

$ cat /proc/loadavg
0.45 0.32 0.28 2/234 12345

# Process CPU usage
$ ps aux --sort=-%cpu | head -10
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
www-data  1234 15.3  5.7 123456 45678 ?        S    09:15   1:23 apache2
mysql     5678 12.1 29.4 987654 234567 ?       Sl   08:30  12:34 mysqld

# CPU usage by process over time
$ pidstat -u 1 5                 # All processes
$ pidstat -u -p 1234 1 5         # Specific process
```

Memory Monitoring

```
# Memory usage overview
$ free -h
              total        used        free      shared  buff/cache   available
Mem:           7.8Gi       3.4Gi       1.2Gi       234Mi       3.2Gi       4.0Gi
Swap:          2.0Gi          0B       2.0Gi

# Detailed memory information
$ cat /proc/meminfo
MemTotal:        8165432 kB
MemFree:         1264532 kB
MemAvailable:    4123456 kB
Buffers:           89012 kB
Cached:          3201234 kB
SwapCached:            0 kB
```

```
Active:          4567890 kB
Inactive:        2345678 kB
Active(anon):    2123456 kB
Inactive(anon):   123456 kB
Active(file):    2444434 kB
Inactive(file): 2222222 kB

# Memory usage by process
$ ps aux --sort=-%mem | head -10
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
mysql     5678 12.1 29.4 987654 234567 ?       Sl   08:30  12:34 mysqld
apache    1234 15.3  5.7 123456 45678 ?        S    09:15   1:23 apache2

# Memory statistics with sar
$ sar -r 1 5                    # Memory utilization
Linux 5.4.0-74-generic (hostname)      12/15/2023      _x86_64_        (4 CPU)

10:30:16 AM kbmemfree kbmemused  %memused kbbuffers  kbcached  kbcommit   %commit
kbactive   kbinact   kbdirty
10:30:17 AM   1264532   6900900     84.51     89012   3201234   4567890     55.93
4567890   2345678      1234

# Memory usage per process with smem
$ sudo apt install smem
$ smem -t
  PID User     Command                         Swap      USS      PSS      RSS
 1234 apache   apache2                            0    12345    23456    45678
 5678 mysql    mysqld                             0   123456   234567   456789
-------------------------------------------------------------------------------
   15 2        2 processes                        0   135801   258023   502467

# Memory mapping of a process
$ pmap -x 1234                  # Process memory map
$ cat /proc/1234/smaps          # Detailed memory mapping

# Check for memory leaks
$ valgrind --tool=memcheck --leak-check=full ./your_program

# Monitor swap usage
$ swapon --show
NAME      TYPE      SIZE USED PRIO
/dev/sda2 partition   2G   0B   -2

$ cat /proc/swaps
Filename                                Type            Size    Used    Priority
/dev/sda2                               partition       2097148 0       -2
```

Disk I/O Monitoring

```
# Disk usage
$ df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        20G   12G  7.2G  63% /
/dev/sda2       100G   45G   50G  48% /home
tmpfs           3.9G     0  3.9G   0% /dev/shm

# Inode usage
$ df -i
Filesystem      Inodes   IUsed   IFree IUse% Mounted on
/dev/sda1      1310720  123456 1187264   10% /
/dev/sda2      6553600  234567 6319033    4% /home

# Directory sizes
$ du -sh /var/log/*
1.2G    /var/log/apache2
456M    /var/log/mysql
123M    /var/log/syslog
89M     /var/log/auth.log

# Find large files
$ find / -type f -size +100M 2>/dev/null | head -10
/var/log/apache2/access.log
/var/lib/mysql/ibdata1
/home/user/large_file.iso

# Real-time disk I/O
$ iostat -x 1 5
Linux 5.4.0-74-generic (hostname)      12/15/2023      _x86_64_        (4 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           5.25    0.00    2.13    0.63    0.00   91.99

Device            r/s     w/s     rkB/s     wkB/s   rrqm/s   wrqm/s  %rrqm  %wrqm
await    r_await    w_await  svctm  %util
sda             12.34     5.67     123.45     67.89     0.12     2.34   0.96  29.23
8.45       6.78       12.34    2.45   4.32

# I/O statistics per process
$ iotop
Total DISK READ :       1.23 M/s | Total DISK WRITE :       456.78 K/s
Actual DISK READ:       1.23 M/s | Actual DISK WRITE:       456.78 K/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN     IO>    COMMAND
 1234 be/4 mysql       123.45 K/s   67.89 K/s  0.00 % 12.34 % mysqld
 5678 be/4 apache       45.67 K/s   23.45 K/s  0.00 %  5.67 % apache2

# Disk I/O per process with pidstat
$ pidstat -d 1 5                 # All processes
$ pidstat -d -p 1234 1 5         # Specific process

# Block device statistics
$ cat /proc/diskstats
   8       0 sda 123456 789 1234567 8901 234567 890 2345678 9012 0 12345 67890
   8       1 sda1 12345 67 123456 789 23456 78 234567 890 0 1234 5678

# Check disk health
```

```
$ sudo smartctl -a /dev/sda
$ sudo smartctl -H /dev/sda       # Health status only
```

Network Monitoring

```
# Network interface statistics
$ ip -s link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT
group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    1234567    8901     0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    1234567    8901     0       0       0       0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether aa:bb:cc:dd:ee:ff brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    12345678901 8901234  0       0       0       123
    TX: bytes  packets  errors  dropped carrier collsns
    9876543210  7654321  0       0       0       0

# Network statistics with sar
$ sar -n DEV 1 5                 # Network device statistics
Linux 5.4.0-74-generic (hostname)      12/15/2023      _x86_64_        (4 CPU)

10:30:16 AM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s
rxmcst/s   %ifutil
10:30:17 AM        lo      1.23      1.23      0.12      0.12      0.00      0.00
0.00      0.00
10:30:17 AM      eth0     45.67     34.56     67.89     45.67      0.00      0.00
0.12      0.05

# Real-time network usage
$ iftop                          # Interactive network usage
$ nethogs                        # Network usage per process
$ nload                          # Network load monitor

# Network connections
$ netstat -tuln                  # Listening ports
$ netstat -tun                   # All connections
$ ss -tuln                       # Modern alternative
$ ss -tun                        # All connections with ss

# Network connection statistics
$ netstat -s
Ip:
    123456 total packets received
    0 forwarded
    0 incoming packets discarded
    123456 incoming packets delivered
    98765 requests sent out
Tcp:
    1234 active connections openings
    567 passive connection openings
    89 failed connection attempts
    12 connection resets received

# Bandwidth monitoring
$ vnstat                         # Network traffic statistics
$ vnstat -d                      # Daily statistics
$ vnstat -m                      # Monthly statistics

# Network latency and packet loss
$ ping -c 10 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 time=12.3 ms
64 bytes from 8.8.8.8: icmp_seq=2 time=11.8 ms
...
--- 8.8.8.8 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9012ms
rtt min/avg/max/mdev = 11.234/12.567/13.890/0.789 ms

# Continuous network monitoring
$ mtr 8.8.8.8                   # My traceroute (combines ping and traceroute)
```

Advanced Monitoring Tools

System Activity Reporter (SAR)

```
# Install sysstat package
$ sudo apt install sysstat

# Enable data collection
$ sudo systemctl enable sysstat
$ sudo systemctl start sysstat

# Configure data collection interval
$ sudo nano /etc/cron.d/sysstat
# Collect data every 10 minutes
*/10 * * * * root /usr/lib/sysstat/debian-sa1 1 1

# Generate daily reports
5 19 * * * root /usr/lib/sysstat/debian-sa2 -A

# View historical data
$ sar -u                         # CPU usage (today)
$ sar -u -f /var/log/sysstat/sa15  # CPU usage (15th of month)
$ sar -r                         # Memory usage
$ sar -d                         # Disk activity
$ sar -n DEV                     # Network statistics
$ sar -q                         # Load average and run queue
$ sar -w                         # Context switches and interrupts

# Generate comprehensive report
$ sar -A                         # All statistics

# Custom time range
$ sar -u -s 09:00:00 -e 17:00:00  # Business hours only

# Export data for analysis
$ sadf -d /var/log/sysstat/sa15 > sar_data.csv
```

Nagios Monitoring

```
# Install Nagios
$ sudo apt update
$ sudo apt install nagios3 nagios-plugins

# Configure Nagios
$ sudo nano /etc/nagios3/nagios.cfg

# Main configuration file
cfg_file=/etc/nagios3/commands.cfg
cfg_dir=/etc/nagios3/conf.d
log_file=/var/log/nagios3/nagios.log
object_cache_file=/var/cache/nagios3/objects.cache
precached_object_file=/var/lib/nagios3/objects.precache
resource_file=/etc/nagios3/resource.cfg
status_file=/var/lib/nagios3/status.dat
status_update_interval=10
nagios_user=nagios
nagios_group=nagios
check_external_commands=1
command_check_interval=-1
command_file=/var/lib/nagios3/rw/nagios.cmd

# Define host
$ sudo nano /etc/nagios3/conf.d/localhost.cfg

define host {
    use                         linux-server
    host_name                   localhost
    alias                       localhost
    address                     127.0.0.1
    max_check_attempts          5
    check_period                24x7
    notification_interval       30
    notification_period         24x7
}

# Define services
```

```
define service {
    use                         generic-service
    host_name                   localhost
    service_description         CPU Load
    check_command               check_load!5.0!4.0!3.0!10.0!6.0!4.0
}

define service {
    use                         generic-service
    host_name                   localhost
    service_description         Disk Space
    check_command               check_all_disks!20%!10%
}

define service {
    use                         generic-service
    host_name                   localhost
    service_description         Memory Usage
    check_command               check_memory!80!90
}

# Custom check commands
$ sudo nano /etc/nagios3/commands.cfg

define command {
    command_name    check_memory
    command_line    /usr/lib/nagios/plugins/check_memory.sh -w $ARG1$ -c $ARG2$
}

# Memory check script
$ sudo nano /usr/lib/nagios/plugins/check_memory.sh

#!/bin/bash
WARNING=$1
CRITICAL=$2

MEMORY_USAGE=$(free | grep Mem | awk '{printf("%.1f", $3/$2 * 100.0)}')

if (( $(echo "$MEMORY_USAGE > $CRITICAL" | bc -l) )); then
    echo "CRITICAL - Memory usage is ${MEMORY_USAGE}%"
    exit 2
elif (( $(echo "$MEMORY_USAGE > $WARNING" | bc -l) )); then
    echo "WARNING - Memory usage is ${MEMORY_USAGE}%"
    exit 1
else
    echo "OK - Memory usage is ${MEMORY_USAGE}%"
    exit 0
fi

$ sudo chmod +x /usr/lib/nagios/plugins/check_memory.sh

# Restart Nagios
$ sudo systemctl restart nagios3
```

```
# Access web interface at http://localhost/nagios3
# Default credentials: nagiosadmin / nagiosadmin
```

Zabbix Monitoring

```
# Install Zabbix repository
$ wget https://repo.zabbix.com/zabbix/5.4/ubuntu/pool/main/z/zabbix-
release/zabbix-release_5.4-1+ubuntu20.04_all.deb
$ sudo dpkg -i zabbix-release_5.4-1+ubuntu20.04_all.deb
$ sudo apt update

# Install Zabbix server, frontend, and agent
$ sudo apt install zabbix-server-mysql zabbix-frontend-php zabbix-apache-conf
zabbix-agent

# Install and configure MySQL
$ sudo apt install mysql-server
$ sudo mysql_secure_installation

# Create Zabbix database
$ sudo mysql -uroot -p
mysql> create database zabbix character set utf8 collate utf8_bin;
mysql> create user zabbix@localhost identified by 'password';
mysql> grant all privileges on zabbix.* to zabbix@localhost;
mysql> quit;

# Import initial schema
$ zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -uzabbix -p
zabbix

# Configure Zabbix server
$ sudo nano /etc/zabbix/zabbix_server.conf

DBPassword=password

# Configure PHP for Zabbix frontend
$ sudo nano /etc/zabbix/apache.conf

php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_size 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
php_value max_input_vars 10000
php_value always_populate_raw_post_data -1
php_value date.timezone Europe/London

# Start Zabbix services
$ sudo systemctl restart zabbix-server zabbix-agent apache2
$ sudo systemctl enable zabbix-server zabbix-agent
```

```
# Access web interface at http://localhost/zabbix
# Default credentials: Admin / zabbix
```

Prometheus and Grafana

```
# Install Prometheus
$ sudo useradd --no-create-home --shell /bin/false prometheus
$ sudo mkdir /etc/prometheus /var/lib/prometheus
$ sudo chown prometheus:prometheus /etc/prometheus /var/lib/prometheus

$ cd /tmp
$ wget
https://github.com/prometheus/prometheus/releases/download/v2.30.3/prometheus-
2.30.3.linux-amd64.tar.gz
$ tar xvf prometheus-2.30.3.linux-amd64.tar.gz
$ sudo cp prometheus-2.30.3.linux-amd64/prometheus /usr/local/bin/
$ sudo cp prometheus-2.30.3.linux-amd64/promtool /usr/local/bin/
$ sudo chown prometheus:prometheus /usr/local/bin/prometheus
/usr/local/bin/promtool

# Configure Prometheus
$ sudo nano /etc/prometheus/prometheus.yml

global:
  scrape_interval: 15s
  evaluation_interval: 15s

rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']

  - job_name: 'node'
    static_configs:
      - targets: ['localhost:9100']

# Create systemd service
$ sudo nano /etc/systemd/system/prometheus.service

[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
```

```
Type=simple
ExecStart=/usr/local/bin/prometheus \
    --config.file /etc/prometheus/prometheus.yml \
    --storage.tsdb.path /var/lib/prometheus/ \
    --web.console.templates=/etc/prometheus/consoles \
    --web.console.libraries=/etc/prometheus/console_libraries

[Install]
WantedBy=multi-user.target

# Install Node Exporter
$ wget
https://github.com/prometheus/node_exporter/releases/download/v1.2.2/node_exporter
-1.2.2.linux-amd64.tar.gz
$ tar xvf node_exporter-1.2.2.linux-amd64.tar.gz
$ sudo cp node_exporter-1.2.2.linux-amd64/node_exporter /usr/local/bin/
$ sudo useradd --no-create-home --shell /bin/false node_exporter
$ sudo chown node_exporter:node_exporter /usr/local/bin/node_exporter

# Create Node Exporter service
$ sudo nano /etc/systemd/system/node_exporter.service

[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

[Install]
WantedBy=multi-user.target

# Start services
$ sudo systemctl daemon-reload
$ sudo systemctl start prometheus node_exporter
$ sudo systemctl enable prometheus node_exporter

# Install Grafana
$ sudo apt-get install -y software-properties-common
$ sudo add-apt-repository "deb https://packages.grafana.com/oss/deb stable main"
$ wget -q -O - https://packages.grafana.com/gpg.key | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install grafana

# Start Grafana
$ sudo systemctl start grafana-server
$ sudo systemctl enable grafana-server

# Access Grafana at http://localhost:3000
# Default credentials: admin / admin
```

⚡ Performance Optimization

CPU Optimization

```
# Check CPU governor
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
powersave

# Set performance governor
$ sudo cpupower frequency-set -g performance
$ echo performance | sudo tee
/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor

# Install and configure cpufrequtils
$ sudo apt install cpufrequtils
$ sudo nano /etc/default/cpufrequtils
GOVERNOR="performance"
MAX_SPEED="0"
MIN_SPEED="0"

# CPU affinity for processes
$ taskset -c 0,1 command            # Run on CPUs 0 and 1
$ taskset -p 0x3 1234               # Set CPU affinity for PID 1234

# Process priority (nice values)
$ nice -n -10 command               # Higher priority
$ renice -10 1234                   # Change priority of running process

# Real-time scheduling
$ chrt -f 99 command                # FIFO real-time scheduling
$ chrt -r 50 command                # Round-robin real-time scheduling

# Disable CPU mitigations for performance (security trade-off)
$ sudo nano /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash mitigations=off"
$ sudo update-grub
```

Memory Optimization

```
# Tune virtual memory settings
$ sudo nano /etc/sysctl.conf

# Reduce swappiness (prefer RAM over swap)
vm.swappiness=10

# Increase dirty page cache
vm.dirty_ratio=15
vm.dirty_background_ratio=5
```

```
# Optimize for database workloads
vm.dirty_expire_centisecs=500
vm.dirty_writeback_centisecs=100

# Increase shared memory
kernel.shmmax=68719476736
kernel.shmall=4294967296

# Apply changes
$ sudo sysctl -p

# Huge pages configuration
$ echo 1024 | sudo tee /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages
$ sudo nano /etc/sysctl.conf
vm.nr_hugepages=1024

# Memory compaction
$ echo 1 | sudo tee /proc/sys/vm/compact_memory

# Drop caches (for testing)
$ sudo sync
$ echo 3 | sudo tee /proc/sys/vm/drop_caches

# Memory allocation policies
$ numactl --hardware                  # Show NUMA topology
$ numactl --cpubind=0 --membind=0 command  # Bind to specific NUMA node
```

Disk I/O Optimization

```
# Check current I/O scheduler
$ cat /sys/block/sda/queue/scheduler
none mq-deadline [kyber] bfq

# Change I/O scheduler
$ echo mq-deadline | sudo tee /sys/block/sda/queue/scheduler

# Permanent I/O scheduler change
$ sudo nano /etc/default/grub
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash elevator=mq-deadline"
$ sudo update-grub

# Optimize mount options
$ sudo nano /etc/fstab
/dev/sda1 / ext4 defaults,noatime,nodiratime 0 1
/dev/sda2 /home ext4 defaults,noatime,nodiratime 0 2

# SSD optimizations
/dev/ssd1 /data ext4 defaults,noatime,discard 0 2

# Tune filesystem parameters
```

```
$ sudo tune2fs -o journal_data_writeback /dev/sda1
$ sudo tune2fs -O ^has_journal /dev/sda1  # Remove journal (risky)

# Adjust read-ahead
$ sudo blockdev --setra 4096 /dev/sda

# Configure I/O limits with cgroups
$ sudo nano /etc/systemd/system/myapp.service
[Service]
IOWeight=500
IODeviceWeight=/dev/sda 750
IOReadBandwidthMax=/dev/sda 100M
IOWriteBandwidthMax=/dev/sda 50M

# Optimize database storage
$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
[mysqld]
innodb_buffer_pool_size=2G
innodb_log_file_size=256M
innodb_flush_log_at_trx_commit=2
innodb_flush_method=O_DIRECT
```

Network Optimization

```
# Network buffer tuning
$ sudo nano /etc/sysctl.conf

# Increase network buffers
net.core.rmem_default=262144
net.core.rmem_max=16777216
net.core.wmem_default=262144
net.core.wmem_max=16777216

# TCP buffer tuning
net.ipv4.tcp_rmem=4096 65536 16777216
net.ipv4.tcp_wmem=4096 65536 16777216

# TCP congestion control
net.ipv4.tcp_congestion_control=bbr

# Increase connection tracking
net.netfilter.nf_conntrack_max=1048576

# Optimize for high-performance networking
net.core.netdev_max_backlog=5000
net.ipv4.tcp_window_scaling=1
net.ipv4.tcp_timestamps=1
net.ipv4.tcp_sack=1
net.ipv4.tcp_no_metrics_save=1

# Apply changes
```

```
$ sudo sysctl -p

# Network interface optimization
$ sudo ethtool -G eth0 rx 4096 tx 4096  # Increase ring buffers
$ sudo ethtool -K eth0 gro on           # Generic receive offload
$ sudo ethtool -K eth0 tso on           # TCP segmentation offload

# CPU affinity for network interrupts
$ sudo nano /etc/rc.local
echo 2 > /proc/irq/24/smp_affinity     # Bind network IRQ to CPU 1
```

📈 Automated Monitoring and Alerting

Custom Monitoring Scripts

```
#!/bin/bash
# /usr/local/bin/system-monitor.sh

LOGFILE="/var/log/system-monitor.log"
ALERT_EMAIL="admin@example.com"
HOSTNAME=$(hostname)
DATE=$(date '+%Y-%m-%d %H:%M:%S')

# Function to send alerts
send_alert() {
    local subject="$1"
    local message="$2"
    echo "[$DATE] ALERT: $subject" >> $LOGFILE
    echo "$message" | mail -s "[$HOSTNAME] $subject" $ALERT_EMAIL
}

# Check CPU load
LOAD_AVG=$(uptime | awk -F'load average:' '{print $2}' | awk '{print $1}' | sed
's/,//')
LOAD_THRESHOLD="4.0"
if (( $(echo "$LOAD_AVG > $LOAD_THRESHOLD" | bc -l) )); then
    send_alert "High CPU Load" "Current load average: $LOAD_AVG (threshold:
$LOAD_THRESHOLD)"
fi

# Check memory usage
MEM_USAGE=$(free | awk 'NR==2{printf "%.1f", $3*100/$2}')
MEM_THRESHOLD="90"
if (( $(echo "$MEM_USAGE > $MEM_THRESHOLD" | bc -l) )); then
    send_alert "High Memory Usage" "Current memory usage: ${MEM_USAGE}%
(threshold: ${MEM_THRESHOLD}%)"
fi

# Check disk usage
while read output; do
    usage=$(echo $output | awk '{print $5}' | sed 's/%//g')
    partition=$(echo $output | awk '{print $6}')
    if [ $usage -ge 90 ]; then
        send_alert "High Disk Usage" "Partition $partition is ${usage}% full"
    fi
done <<< "$(df -h | grep -vE '^Filesystem|tmpfs|cdrom')"

# Check service status
SERVICES=("apache2" "mysql" "ssh")
for service in "${SERVICES[@]}"; do
    if ! systemctl is-active --quiet $service; then
        send_alert "Service Down" "Service $service is not running"
    fi
done

# Check network connectivity
if ! ping -c 1 8.8.8.8 &> /dev/null; then
    send_alert "Network Connectivity" "Cannot reach external network (8.8.8.8)"
fi

# Log successful check
echo "[$DATE] System check completed" >> $LOGFILE

# Schedule with cron
# */5 * * * * /usr/local/bin/system-monitor.sh
```

Performance Baseline Script

```
#!/bin/bash
# /usr/local/bin/performance-baseline.sh

OUTPUT_DIR="/var/log/performance-baseline"
DATE=$(date '+%Y%m%d_%H%M%S')
REPORT_FILE="$OUTPUT_DIR/baseline_$DATE.txt"

mkdir -p $OUTPUT_DIR

echo "Performance Baseline Report - $DATE" > $REPORT_FILE
echo "==========================================" >> $REPORT_FILE
echo "" >> $REPORT_FILE

# System information
echo "SYSTEM INFORMATION:" >> $REPORT_FILE
echo "Hostname: $(hostname)" >> $REPORT_FILE
echo "Uptime: $(uptime)" >> $REPORT_FILE
echo "Kernel: $(uname -r)" >> $REPORT_FILE
echo "" >> $REPORT_FILE

# CPU information
echo "CPU INFORMATION:" >> $REPORT_FILE
lscpu >> $REPORT_FILE
echo "" >> $REPORT_FILE
```

```
# Memory information
echo "MEMORY INFORMATION:" >> $REPORT_FILE
free -h >> $REPORT_FILE
echo "" >> $REPORT_FILE

# Disk information
echo "DISK INFORMATION:" >> $REPORT_FILE
df -h >> $REPORT_FILE
echo "" >> $REPORT_FILE

# Network information
echo "NETWORK INFORMATION:" >> $REPORT_FILE
ip addr show >> $REPORT_FILE
echo "" >> $REPORT_FILE

# Performance metrics
echo "PERFORMANCE METRICS:" >> $REPORT_FILE
echo "Load Average: $(cat /proc/loadavg)" >> $REPORT_FILE
echo "Memory Usage: $(free | awk 'NR==2{printf "%.1f%%", $3*100/$2}')" >>
$REPORT_FILE
echo "Disk I/O: $(iostat -x 1 1 | tail -n +4)" >> $REPORT_FILE
echo "" >> $REPORT_FILE

# Top processes
echo "TOP PROCESSES (CPU):" >> $REPORT_FILE
ps aux --sort=-%cpu | head -10 >> $REPORT_FILE
echo "" >> $REPORT_FILE

echo "TOP PROCESSES (Memory):" >> $REPORT_FILE
ps aux --sort=-%mem | head -10 >> $REPORT_FILE
echo "" >> $REPORT_FILE

# Network connections
echo "NETWORK CONNECTIONS:" >> $REPORT_FILE
netstat -tuln >> $REPORT_FILE
echo "" >> $REPORT_FILE

echo "Baseline report saved to: $REPORT_FILE"
```

🧠 Knowledge Check

Quick Quiz

1. **What does a load average of 2.0 mean on a 4-core system?**

   ▶ Answer

   A load average of 2.0 on a 4-core system means the system is 50% utilized on average. The system can handle up to 4.0 before being fully utilized.

2. **What's the difference between buffer and cache in memory usage?**

▶ Answer

Buffers are used for block device I/O operations, while cache stores frequently accessed file contents. Both can be freed when applications need more memory.

3. **What does %iowait in CPU statistics indicate?**

▶ Answer

%iowait shows the percentage of time the CPU is idle while waiting for I/O operations to complete. High iowait indicates disk or network bottlenecks.

4. **How do you identify which process is causing high disk I/O?**

▶ Answer

Use `iotop` for real-time I/O monitoring or `pidstat -d` to see per-process disk statistics.

Hands-On Challenges

Challenge 1: Complete Monitoring Setup

```
# Set up comprehensive monitoring:
# - Install and configure Prometheus + Grafana
# - Create custom dashboards for system metrics
# - Set up alerting rules for critical thresholds
# - Implement automated reporting
# - Configure log aggregation and analysis
```

Challenge 2: Performance Optimization

```
# Optimize a system for specific workload:
# - Identify performance bottlenecks
# - Tune kernel parameters for the workload
# - Optimize application configurations
# - Implement caching strategies
# - Measure and document improvements
```

Challenge 3: Capacity Planning

```
# Create a capacity planning system:
# - Collect historical performance data
# - Analyze growth trends
# - Predict future resource requirements
# - Create scaling recommendations
# - Implement automated scaling triggers
```

🚀 Next Steps

Excellent! You've mastered performance monitoring and optimization. You can now:

- Monitor all system resources (CPU, memory, disk, network)
- Use advanced monitoring tools (SAR, Nagios, Prometheus, Grafana)
- Identify and resolve performance bottlenecks
- Optimize system performance for specific workloads
- Set up automated monitoring and alerting
- Plan for future capacity requirements
- Troubleshoot complex performance issues

Ready for automation and scripting? Continue to 14-automation-scripting.md to learn about automating system administration tasks and creating powerful scripts.

> **Pro Tip**: Performance monitoring is an ongoing process. Establish baselines, monitor trends, and optimize proactively rather than reactively. Always measure before and after changes to validate improvements! 📊

🤖 Automation & Scripting: Mastering System Administration Automation

> **Automate repetitive tasks, create powerful scripts, and build efficient workflows**

📖 What You'll Learn

Automation is the key to efficient system administration. This chapter covers comprehensive automation techniques and scripting:

- Bash scripting fundamentals and advanced techniques
- Task automation with cron and systemd timers
- Configuration management with Ansible
- Infrastructure as Code (IaC) concepts
- Log management and rotation automation
- Backup automation strategies
- System maintenance automation
- Monitoring and alerting automation
- Error handling and debugging scripts

🌍 Why This Matters

Critical applications:

- **Efficiency**: Automate repetitive tasks to save time and reduce errors
- **Consistency**: Ensure tasks are performed the same way every time
- **Scalability**: Manage multiple systems efficiently
- **Reliability**: Reduce human error and improve system reliability

- **Cost Reduction**: Minimize manual intervention and operational costs

📝 Bash Scripting Fundamentals

Script Structure and Best Practices

```
#!/bin/bash
# script-template.sh - Professional script template

# Script metadata
SCRIPT_NAME="$(basename "$0")"
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
SCRIPT_VERSION="1.0.0"
SCRIPT_AUTHOR="Your Name"
SCRIPT_DATE="$(date '+%Y-%m-%d')"

# Configuration
LOG_FILE="/var/log/${SCRIPT_NAME%.sh}.log"
CONFIG_FILE="${SCRIPT_DIR}/${SCRIPT_NAME%.sh}.conf"
LOCK_FILE="/var/run/${SCRIPT_NAME%.sh}.lock"
DEBUG=${DEBUG:-0}
VERBOSE=${VERBOSE:-0}

# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

# Logging functions
log() {
    local level="$1"
    shift
    local message="$*"
    local timestamp="$(date '+%Y-%m-%d %H:%M:%S')"
    echo "[$timestamp] [$level] $message" | tee -a "$LOG_FILE"
}

log_info() { log "INFO" "$@"; }
log_warn() { log "WARN" "$@"; }
log_error() { log "ERROR" "$@"; }
log_debug() { [[ $DEBUG -eq 1 ]] && log "DEBUG" "$@"; }

# Output functions
print_info() { echo -e "${BLUE}[INFO]${NC} $*"; }
print_success() { echo -e "${GREEN}[SUCCESS]${NC} $*"; }
print_warning() { echo -e "${YELLOW}[WARNING]${NC} $*"; }
print_error() { echo -e "${RED}[ERROR]${NC} $*" >&2; }

# Error handling
set -euo pipefail  # Exit on error, undefined vars, pipe failures
```

```
error_handler() {
    local line_number="$1"
    local error_code="$2"
    log_error "Script failed at line $line_number with exit code $error_code"
    cleanup
    exit "$error_code"
}

trap 'error_handler ${LINENO} $?' ERR

# Cleanup function
cleanup() {
    log_info "Performing cleanup..."
    [[ -f "$LOCK_FILE" ]] && rm -f "$LOCK_FILE"
    # Add other cleanup tasks here
}

trap cleanup EXIT

# Lock mechanism to prevent multiple instances
acquire_lock() {
    if [[ -f "$LOCK_FILE" ]]; then
        local pid=$(cat "$LOCK_FILE")
        if kill -0 "$pid" 2>/dev/null; then
            log_error "Script is already running (PID: $pid)"
            exit 1
        else
            log_warn "Removing stale lock file"
            rm -f "$LOCK_FILE"
        fi
    fi
    echo $$ > "$LOCK_FILE"
    log_info "Lock acquired (PID: $$)"
}

# Usage function
usage() {
    cat << EOF
Usage: $SCRIPT_NAME [OPTIONS]

Description:
    Professional script template with logging, error handling, and best practices.

Options:
    -h, --help          Show this help message
    -v, --verbose       Enable verbose output
    -d, --debug         Enable debug mode
    -c, --config FILE   Use custom config file
    --version           Show version information

Examples:
    $SCRIPT_NAME --verbose
    $SCRIPT_NAME --config /path/to/config.conf
    $SCRIPT_NAME --debug

Author: $SCRIPT_AUTHOR
Version: $SCRIPT_VERSION
Date: $SCRIPT_DATE
EOF
}

# Configuration loading
load_config() {
    if [[ -f "$CONFIG_FILE" ]]; then
        log_info "Loading configuration from $CONFIG_FILE"
        source "$CONFIG_FILE"
    else
        log_warn "Configuration file not found: $CONFIG_FILE"
    fi
}

# Argument parsing
parse_arguments() {
    while [[ $# -gt 0 ]]; do
        case $1 in
            -h|--help)
                usage
                exit 0
                ;;
            -v|--verbose)
                VERBOSE=1
                shift
                ;;
            -d|--debug)
                DEBUG=1
                shift
                ;;
            -c|--config)
                CONFIG_FILE="$2"
                shift 2
                ;;
            --version)
                echo "$SCRIPT_NAME version $SCRIPT_VERSION"
                exit 0
                ;;
            *)
                print_error "Unknown option: $1"
                usage
                exit 1
                ;;
        esac
    done
}

# Validation functions
validate_requirements() {
    local required_commands=("curl" "jq" "awk")
    local missing_commands=()

    for cmd in "${required_commands[@]}"; do
        if ! command -v "$cmd" &> /dev/null; then
            missing_commands+=("$cmd")
        fi
    done

    if [[ ${#missing_commands[@]} -gt 0 ]]; then
        log_error "Missing required commands: ${missing_commands[*]}"
        exit 1
    fi
}

validate_permissions() {
    if [[ $EUID -eq 0 ]]; then
        log_warn "Running as root user"
    fi

    if [[ ! -w "$(dirname "$LOG_FILE")" ]]; then
        log_error "Cannot write to log directory: $(dirname "$LOG_FILE")"
        exit 1
    fi
}

# Main function
main() {
    log_info "Starting $SCRIPT_NAME v$SCRIPT_VERSION"

    # Your main script logic here
    print_info "Script execution started"

    # Example operations
    if [[ $VERBOSE -eq 1 ]]; then
        print_info "Verbose mode enabled"
    fi

    if [[ $DEBUG -eq 1 ]]; then
        print_info "Debug mode enabled"
        set -x  # Enable command tracing
    fi

    # Simulate some work
    sleep 2

    print_success "Script completed successfully"
    log_info "$SCRIPT_NAME completed successfully"
}

# Script execution
if [[ "${BASH_SOURCE[0]}" == "${0}" ]]; then
    parse_arguments "$@"
    acquire_lock
    load_config
    validate_requirements
    validate_permissions
    main
fi
```

Advanced Bash Techniques

```
#!/bin/bash
# advanced-bash-examples.sh

# Arrays and associative arrays
declare -a servers=("web1" "web2" "db1")
declare -A server_ips=(
    ["web1"]="192.168.1.10"
    ["web2"]="192.168.1.11"
    ["db1"]="192.168.1.20"
)

# Function with return values
check_service() {
    local service="$1"
    local host="$2"

    if ssh "$host" "systemctl is-active --quiet $service"; then
        return 0  # Service is running
    else
        return 1  # Service is not running
    fi
}

# Process arrays
for server in "${servers[@]}"; do
    ip="${server_ips[$server]}"
    echo "Checking server: $server ($ip)"

    if check_service "apache2" "$ip"; then
        echo "✓ Apache is running on $server"
    else
        echo "✗ Apache is not running on $server"
    fi
done

# Here documents for multi-line strings
generate_config() {
    local server_name="$1"
    local server_ip="$2"

    cat << EOF > "/etc/nginx/sites-available/$server_name"
server {
    listen 80;
    server_name $server_name;
    location / {
        proxy_pass http://$server_ip:8080;
        proxy_set_header Host \$host;
        proxy_set_header X-Real-IP \$remote_addr;
    }
}
EOF
}

# Parameter expansion and string manipulation
process_filename() {
    local filepath="$1"

    # Extract components
    local filename="${filepath##*/}"          # basename
    local directory="${filepath%/*}"          # dirname
    local extension="${filename##*.}"         # file extension
    local basename="${filename%.*}"           # filename without extension

    echo "Full path: $filepath"
    echo "Directory: $directory"
    echo "Filename: $filename"
    echo "Basename: $basename"
    echo "Extension: $extension"

    # String replacement
    local new_filename="${filename/old/new}"  # Replace first occurrence
    local all_replaced="${filename//old/new}" # Replace all occurrences

    echo "Modified: $new_filename"
    echo "All replaced: $all_replaced"
}

# Process substitution
compare_processes() {
    # Compare running processes on two servers
    diff <(ssh server1 "ps aux | sort") <(ssh server2 "ps aux | sort")
}

# Command substitution and arithmetic
calculate_disk_usage() {
    local path="$1"
    local size_bytes=$(du -sb "$path" | cut -f1)
    local size_mb=$((size_bytes / 1024 / 1024))
    local size_gb=$((size_mb / 1024))

    echo "Path: $path"
    echo "Size: ${size_bytes} bytes"
    echo "Size: ${size_mb} MB"
    echo "Size: ${size_gb} GB"
}

# Conditional expressions
```

```
validate_input() {
    local input="$1"

    # Multiple conditions
    if [[ -n "$input" && "$input" =~ ^[0-9]+$ && $input -gt 0 && $input -lt 100
]]; then
        echo "Valid input: $input"
        return 0
    else
        echo "Invalid input: $input"
        return 1
    fi
}

# File operations with error checking
safe_file_operations() {
    local source="$1"
    local destination="$2"

    # Check if source exists and is readable
    if [[ ! -r "$source" ]]; then
        echo "Error: Cannot read source file: $source" >&2
        return 1
    fi

    # Create destination directory if it doesn't exist
    local dest_dir="$(dirname "$destination")"
    if [[ ! -d "$dest_dir" ]]; then
        mkdir -p "$dest_dir" || {
            echo "Error: Cannot create directory: $dest_dir" >&2
            return 1
        }
    fi

    # Copy with verification
    if cp "$source" "$destination"; then
        echo "Successfully copied $source to $destination"

        # Verify copy
        if cmp -s "$source" "$destination"; then
            echo "Copy verified successfully"
        else
            echo "Warning: Copy verification failed" >&2
            return 1
        fi
    else
        echo "Error: Failed to copy $source to $destination" >&2
        return 1
    fi
}

# Parallel processing
parallel_ping() {
    local hosts=("8.8.8.8" "1.1.1.1" "208.67.222.222")
    local pids=()

    # Start background processes
    for host in "${hosts[@]}"; do
        {
            if ping -c 3 "$host" &>/dev/null; then
                echo "$host: REACHABLE"
            else
                echo "$host: UNREACHABLE"
            fi
        } &
        pids+=("$!")
    done

    # Wait for all processes to complete
    for pid in "${pids[@]}"; do
        wait "$pid"
    done
}

# Signal handling
handle_signals() {
    local cleanup_needed=false

    cleanup_handler() {
        echo "Received signal, cleaning up..."
        cleanup_needed=true
        # Perform cleanup operations
        exit 0
    }

    trap cleanup_handler SIGINT SIGTERM

    # Long-running operation
    for i in {1..60}; do
        if [[ $cleanup_needed == true ]]; then
            break
        fi
        echo "Working... $i/60"
        sleep 1
    done
}
```

⏰ Task Scheduling and Automation

Cron Jobs

```
# View current cron jobs
$ crontab -l

# Edit cron jobs
```

```
$ crontab -e

# Cron syntax: minute hour day month weekday command
# * * * * * command
# | | | | |
# | | | | +-- Day of week (0-7, Sunday = 0 or 7)
# | | | +---- Month (1-12)
# | | +------ Day of month (1-31)
# | +-------- Hour (0-23)
# +---------- Minute (0-59)

# Common cron examples
# Run every minute
* * * * * /usr/local/bin/check-services.sh

# Run every 5 minutes
*/5 * * * * /usr/local/bin/monitor-disk.sh

# Run every hour at minute 0
0 * * * * /usr/local/bin/cleanup-logs.sh

# Run daily at 2:30 AM
30 2 * * * /usr/local/bin/backup-database.sh

# Run weekly on Sunday at 3:00 AM
0 3 * * 0 /usr/local/bin/weekly-maintenance.sh

# Run monthly on the 1st at 4:00 AM
0 4 1 * * /usr/local/bin/monthly-report.sh

# Run on weekdays at 9:00 AM
0 9 * * 1-5 /usr/local/bin/business-hours-check.sh

# Multiple times per day
0 6,12,18 * * * /usr/local/bin/three-times-daily.sh

# System-wide cron jobs
$ sudo nano /etc/crontab

# Example system crontab
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=admin@example.com

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report
/etc/cron.monthly )

# Cron directories for different intervals
```

```
/etc/cron.hourly/
/etc/cron.daily/
/etc/cron.weekly/
/etc/cron.monthly/

# Example cron script with logging
#!/bin/bash
# /etc/cron.daily/system-maintenance

LOGFILE="/var/log/system-maintenance.log"
DATE=$(date '+%Y-%m-%d %H:%M:%S')

echo "[$DATE] Starting system maintenance" >> $LOGFILE

# Update package database
apt update >> $LOGFILE 2>&1

# Clean package cache
apt autoclean >> $LOGFILE 2>&1

# Remove orphaned packages
apt autoremove -y >> $LOGFILE 2>&1

# Clean temporary files
find /tmp -type f -atime +7 -delete >> $LOGFILE 2>&1

# Rotate logs
logrotate /etc/logrotate.conf >> $LOGFILE 2>&1

echo "[$DATE] System maintenance completed" >> $LOGFILE
```

Systemd Timers

```
# Create a systemd service
$ sudo nano /etc/systemd/system/backup.service

[Unit]
Description=Database Backup Service
Wants=backup.timer

[Service]
Type=oneshot
User=backup
Group=backup
ExecStart=/usr/local/bin/backup-database.sh
StandardOutput=journal
StandardError=journal

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

```
# Create a systemd timer
$ sudo nano /etc/systemd/system/backup.timer

[Unit]
Description=Run backup service daily
Requires=backup.service

[Timer]
# Run daily at 2:00 AM
OnCalendar=daily
Persistent=true
RandomizedDelaySec=300

[Install]
WantedBy=timers.target

# Enable and start the timer
$ sudo systemctl daemon-reload
$ sudo systemctl enable backup.timer
$ sudo systemctl start backup.timer

# Check timer status
$ sudo systemctl status backup.timer
$ sudo systemctl list-timers

# Advanced timer examples
# /etc/systemd/system/monitoring.timer
[Timer]
# Every 5 minutes
OnCalendar=*:0/5

# /etc/systemd/system/weekly-report.timer
[Timer]
# Every Monday at 9:00 AM
OnCalendar=Mon *-*-* 09:00:00

# /etc/systemd/system/hourly-check.timer
[Timer]
# Every hour, 10 minutes after the hour
OnCalendar=*-*-* *:10:00

# /etc/systemd/system/business-hours.timer
[Timer]
# Weekdays from 9 AM to 5 PM, every hour
OnCalendar=Mon..Fri 09..17:00:00
```

🔧 Configuration Management with Ansible

Ansible Installation and Setup

```
# Install Ansible
$ sudo apt update
$ sudo apt install ansible

# Or install via pip
$ pip3 install ansible

# Verify installation
$ ansible --version
ansible [core 2.12.1]
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/home/user/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location =
/home/user/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.9.2 (default, Feb 28 2021, 17:03:44) [GCC 10.2.1 20210110]
  jinja version = 2.11.3
  libyaml = True

# Create Ansible directory structure
$ mkdir -p ~/ansible/{inventories,playbooks,roles,group_vars,host_vars}
$ cd ~/ansible

# Create inventory file
$ nano inventories/production

[webservers]
web1 ansible_host=192.168.1.10 ansible_user=ubuntu
web2 ansible_host=192.168.1.11 ansible_user=ubuntu

[databases]
db1 ansible_host=192.168.1.20 ansible_user=ubuntu

[loadbalancers]
lb1 ansible_host=192.168.1.5 ansible_user=ubuntu

[production:children]
webservers
databases
loadbalancers

[production:vars]
ansible_ssh_private_key_file=~/.ssh/id_rsa
ansible_python_interpreter=/usr/bin/python3

# Test connectivity
$ ansible all -i inventories/production -m ping
web1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

Ansible Playbooks

```
# playbooks/webserver-setup.yml
---
- name: Configure Web Servers
  hosts: webservers
  become: yes
  vars:
    nginx_port: 80
    app_user: webapp
    app_dir: /var/www/html

  tasks:
    - name: Update package cache
      apt:
        update_cache: yes
        cache_valid_time: 3600

    - name: Install required packages
      apt:
        name:
          - nginx
          - ufw
          - certbot
          - python3-certbot-nginx
        state: present

    - name: Create application user
      user:
        name: "{{ app_user }}"
        system: yes
        shell: /bin/bash
        home: "{{ app_dir }}"
        create_home: yes

    - name: Configure nginx
      template:
        src: nginx.conf.j2
        dest: /etc/nginx/sites-available/default
        backup: yes
      notify: restart nginx

    - name: Enable nginx site
      file:
        src: /etc/nginx/sites-available/default
        dest: /etc/nginx/sites-enabled/default
        state: link
      notify: restart nginx

    - name: Configure firewall
      ufw:
        rule: allow
        port: "{{ item }}"
        proto: tcp
      loop:
        - "22"
        - "80"
        - "443"

    - name: Enable firewall
      ufw:
        state: enabled
        policy: deny
        direction: incoming

    - name: Start and enable services
      systemd:
        name: "{{ item }}"
        state: started
        enabled: yes
      loop:
        - nginx
        - ufw

    - name: Deploy application files
      copy:
        src: "{{ item.src }}"
        dest: "{{ app_dir }}/{{ item.dest }}"
        owner: "{{ app_user }}"
        group: "{{ app_user }}"
        mode: "0644"
      loop:
        - { src: "index.html", dest: "index.html" }
        - { src: "style.css", dest: "css/style.css" }
      notify: restart nginx

  handlers:
    - name: restart nginx
      systemd:
        name: nginx
        state: restarted
```

```
# playbooks/database-setup.yml
---
- name: Configure Database Servers
  hosts: databases
  become: yes
  vars:
    mysql_root_password: "{{ vault_mysql_root_password }}"
    mysql_databases:
      - name: webapp_prod
        encoding: utf8mb4
        collation: utf8mb4_unicode_ci
    mysql_users:
      - name: webapp_user
        password: "{{ vault_mysql_webapp_password }}"
        priv: "webapp_prod.*:ALL"
        host: "%"

  tasks:
    - name: Install MySQL server
      apt:
        name:
          - mysql-server
          - mysql-client
          - python3-pymysql
        state: present

    - name: Start and enable MySQL
      systemd:
        name: mysql
        state: started
        enabled: yes

    - name: Set MySQL root password
      mysql_user:
        name: root
        password: "{{ mysql_root_password }}"
        login_unix_socket: /var/run/mysqld/mysqld.sock

    - name: Create MySQL databases
      mysql_db:
        name: "{{ item.name }}"
        encoding: "{{ item.encoding }}"
        collation: "{{ item.collation }}"
        state: present
        login_user: root
        login_password: "{{ mysql_root_password }}"
      loop: "{{ mysql_databases }}"

    - name: Create MySQL users
      mysql_user:
        name: "{{ item.name }}"
        password: "{{ item.password }}"
        priv: "{{ item.priv }}"
        host: "{{ item.host }}"
        state: present
        login_user: root
        login_password: "{{ mysql_root_password }}"
      loop: "{{ mysql_users }}"

    - name: Configure MySQL
      template:
        src: my.cnf.j2
        dest: /etc/mysql/mysql.conf.d/mysqld.cnf
        backup: yes
      notify: restart mysql

    - name: Configure firewall for MySQL
      ufw:
        rule: allow
        port: "3306"
        src: "{{ hostvars[item]['ansible_default_ipv4']['address'] }}"
      loop: "{{ groups['webservers'] }}"

  handlers:
    - name: restart mysql
      systemd:
        name: mysql
        state: restarted
```

Ansible Roles

```
# Create role structure
$ ansible-galaxy init roles/common
$ ansible-galaxy init roles/webserver
$ ansible-galaxy init roles/database

# Role structure
roles/
├── common/
│   ├── tasks/
│   │   └── main.yml
│   ├── handlers/
│   │   └── main.yml
│   ├── templates/
│   ├── files/
│   ├── vars/
│   │   └── main.yml
│   ├── defaults/
│   │   └── main.yml
│   └── meta/
│       └── main.yml
```

```
# roles/common/tasks/main.yml
---
- name: Update package cache
  apt:
    update_cache: yes
    cache_valid_time: 3600
```

```yaml
  - name: Install common packages
    apt:
      name:
        - curl
        - wget
        - vim
        - htop
        - git
        - unzip
        - software-properties-common
      state: present

  - name: Configure timezone
    timezone:
      name: "{{ system_timezone | default('UTC') }}"

  - name: Configure NTP
    apt:
      name: ntp
      state: present

  - name: Start and enable NTP
    systemd:
      name: ntp
      state: started
      enabled: yes

  - name: Create admin user
    user:
      name: "{{ admin_user }}"
      groups: sudo
      shell: /bin/bash
      create_home: yes

  - name: Add SSH key for admin user
    authorized_key:
      user: "{{ admin_user }}"
      key: "{{ admin_ssh_key }}"
      state: present

  - name: Configure SSH
    template:
      src: sshd_config.j2
      dest: /etc/ssh/sshd_config
      backup: yes
    notify: restart ssh

  - name: Disable root login
    lineinfile:
      path: /etc/ssh/sshd_config
      regexp: "^PermitRootLogin"
      line: "PermitRootLogin no"
    notify: restart ssh
```

```yaml
# playbooks/site.yml - Main playbook using roles
---
- name: Configure all servers
  hosts: all
  become: yes
  roles:
    - common

- name: Configure web servers
  hosts: webservers
  become: yes
  roles:
    - webserver

- name: Configure database servers
  hosts: databases
  become: yes
  roles:
    - database
```

📊 Log Management Automation

Automated Log Rotation

```
# Configure logrotate
$ sudo nano /etc/logrotate.d/webapp

/var/log/webapp/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 0644 webapp webapp
    postrotate
        systemctl reload webapp
    endscript
}

# Custom logrotate configuration
/var/log/nginx/*.log {
    daily
    missingok
    rotate 52
    compress
    delaycompress
    notifempty
    create 0644 www-data adm
    sharedscripts
    prerotate
        if [ -d /etc/logrotate.d/httpd-prerotate ]; then \
            run-parts /etc/logrotate.d/httpd-prerotate; \
        fi \
    endscript
    postrotate
        invoke-rc.d nginx rotate >/dev/null 2>&1
    endscript
}

# Test logrotate configuration
$ sudo logrotate -d /etc/logrotate.d/webapp
$ sudo logrotate -f /etc/logrotate.d/webapp  # Force rotation

# Custom log rotation script
#!/bin/bash
# /usr/local/bin/custom-log-rotate.sh

LOG_DIR="/var/log/myapp"
RETENTION_DAYS=30
COMPRESS_AFTER_DAYS=1

# Rotate logs
find "$LOG_DIR" -name "*.log" -type f | while read logfile; do
    # Get file modification time
    file_age=$(stat -c %Y "$logfile")
    current_time=$(date +%s)
    age_days=$(( (current_time - file_age) / 86400 ))

    # Compress old logs
    if [[ $age_days -gt $COMPRESS_AFTER_DAYS ]] && [[ ! "$logfile" =~ \.gz$ ]];
then
        gzip "$logfile"
        echo "Compressed: $logfile"
    fi

    # Remove very old logs
    if [[ $age_days -gt $RETENTION_DAYS ]]; then
        rm -f "$logfile" "${logfile}.gz"
        echo "Removed: $logfile"
    fi
done

# Restart application to release file handles
systemctl reload myapp
```

Centralized Logging with rsyslog

```
# Configure rsyslog server
$ sudo nano /etc/rsyslog.conf
```

```
# Enable UDP reception
$ModLoad imudp
$UDPServerRun 514
$UDPServerAddress 0.0.0.0

# Enable TCP reception
$ModLoad imtcp
$InputTCPServerRun 514

# Template for remote logs
$template RemoteLogs,"/var/log/remote/%HOSTNAME%/%PROGRAMNAME%.log"
*.* ?RemoteLogs
& stop

# Configure rsyslog client
$ sudo nano /etc/rsyslog.d/50-remote.conf

# Send all logs to remote server
*.* @@log-server.example.com:514

# Send specific logs
mail.* @@log-server.example.com:514
auth.* @@log-server.example.com:514

# Restart rsyslog
$ sudo systemctl restart rsyslog

# Automated log analysis script
#!/bin/bash
# /usr/local/bin/log-analyzer.sh

LOG_FILE="/var/log/auth.log"
ALERT_EMAIL="admin@example.com"
FAILED_LOGIN_THRESHOLD=5
TIME_WINDOW=300  # 5 minutes

# Check for failed login attempts
check_failed_logins() {
    local current_time=$(date +%s)
    local start_time=$((current_time - TIME_WINDOW))

    # Get recent failed login attempts
    local failed_attempts=$(awk -v start="$start_time" '
        BEGIN {
            "date -d \"" $1 " " $2 " " $3 "\" +%s" | getline timestamp
            if (timestamp >= start && /Failed password/) {
                print $0
            }
        }
    ' "$LOG_FILE" | wc -l)

    if [[ $failed_attempts -gt $FAILED_LOGIN_THRESHOLD ]]; then
        local message="Warning: $failed_attempts failed login attempts in the last
$((TIME_WINDOW/60)) minutes"
        echo "$message" | mail -s "Security Alert: Failed Logins" "$ALERT_EMAIL"
        logger "SECURITY_ALERT: $message"
    fi
}

# Check for suspicious activities
check_suspicious_activity() {
    # Check for privilege escalation
    if grep -q "sudo.*COMMAND" "$LOG_FILE"; then
        local sudo_commands=$(grep "sudo.*COMMAND" "$LOG_FILE" | tail -10)
        echo "Recent sudo commands:" | mail -s "Sudo Activity Report"
"$ALERT_EMAIL"
        echo "$sudo_commands" | mail -s "Sudo Activity Report" "$ALERT_EMAIL"
    fi

    # Check for new user creation
    if grep -q "useradd" "$LOG_FILE"; then
        local new_users=$(grep "useradd" "$LOG_FILE" | tail -5)
        echo "New user accounts created:" | mail -s "New User Alert"
"$ALERT_EMAIL"
        echo "$new_users" | mail -s "New User Alert" "$ALERT_EMAIL"
    fi
}

check_failed_logins
check_suspicious_activity
```

💾 Backup Automation

Database Backup Script

```
#!/bin/bash
# /usr/local/bin/backup-database.sh

# Configuration
DB_USER="backup_user"
DB_PASS="backup_password"
DB_HOST="localhost"
BACKUP_DIR="/backup/mysql"
RETENTION_DAYS=30
S3_BUCKET="my-backup-bucket"
ENCRYPTION_KEY="/etc/backup/encryption.key"

# Logging
LOG_FILE="/var/log/backup-database.log"
exec 1> >(tee -a "$LOG_FILE")
exec 2>&1

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*"
}

# Create backup directory
mkdir -p "$BACKUP_DIR"

# Get list of databases
DATABASES=$(mysql -u"$DB_USER" -p"$DB_PASS" -h"$DB_HOST" -e "SHOW DATABASES;" |
grep -Ev "^(Database|information_schema|performance_schema|mysql|sys)$")

log "Starting database backup"

for db in $DATABASES; do
    log "Backing up database: $db"

    # Create backup filename
    BACKUP_FILE="$BACKUP_DIR/${db}_$(date +%Y%m%d_%H%M%S).sql"

    # Create database dump
    if mysqldump -u"$DB_USER" -p"$DB_PASS" -h"$DB_HOST" \
        --single-transaction \
        --routines \
        --triggers \
        --events \
        "$db" > "$BACKUP_FILE"; then

        log "Database dump created: $BACKUP_FILE"

        # Compress backup
        if gzip "$BACKUP_FILE"; then
            BACKUP_FILE="${BACKUP_FILE}.gz"
            log "Backup compressed: $BACKUP_FILE"
        else
            log "ERROR: Failed to compress backup"
            continue
        fi

        # Encrypt backup
        if [[ -f "$ENCRYPTION_KEY" ]]; then
            if openssl enc -aes-256-cbc -salt -in "$BACKUP_FILE" -out
"${BACKUP_FILE}.enc" -pass file:"$ENCRYPTION_KEY"; then
                rm "$BACKUP_FILE"
                BACKUP_FILE="${BACKUP_FILE}.enc"
                log "Backup encrypted: $BACKUP_FILE"
            else
                log "ERROR: Failed to encrypt backup"
            fi
        fi

        # Upload to S3
        if command -v aws &> /dev/null; then
            if aws s3 cp "$BACKUP_FILE" "s3://$S3_BUCKET/mysql/$(basename
"$BACKUP_FILE")"; then
                log "Backup uploaded to S3: $BACKUP_FILE"
            else
```

```
                log "ERROR: Failed to upload backup to S3"
            fi
        fi

    else
        log "ERROR: Failed to create database dump for $db"
    fi
done

# Clean up old backups
log "Cleaning up old backups (older than $RETENTION_DAYS days)"
find "$BACKUP_DIR" -name "*.sql.gz*" -mtime +$RETENTION_DAYS -delete

# Clean up old S3 backups
if command -v aws &> /dev/null; then
    aws s3 ls "s3://$S3_BUCKET/mysql/" | while read -r line; do
        file_date=$(echo "$line" | awk '{print $1}')
        file_name=$(echo "$line" | awk '{print $4}')

        if [[ -n "$file_date" && -n "$file_name" ]]; then
            file_timestamp=$(date -d "$file_date" +%s)
            cutoff_timestamp=$(date -d "$RETENTION_DAYS days ago" +%s)

            if [[ $file_timestamp -lt $cutoff_timestamp ]]; then
                aws s3 rm "s3://$S3_BUCKET/mysql/$file_name"
                log "Removed old S3 backup: $file_name"
            fi
        fi
    done
fi

log "Database backup completed"

# Send notification
if command -v mail &> /dev/null; then
    echo "Database backup completed successfully at $(date)" | \
        mail -s "Database Backup Report" admin@example.com
fi
```

File System Backup Script

```
#!/bin/bash
# /usr/local/bin/backup-filesystem.sh

# Configuration
SOURCE_DIRS=("/etc" "/home" "/var/www" "/opt")
BACKUP_DEST="/backup/filesystem"
REMOTE_DEST="backup@backup-server:/backup/$(hostname)"
EXCLUDE_FILE="/etc/backup/exclude.txt"
RETENTION_DAYS=14
MAX_BACKUP_SIZE="10G"
```

```
# Logging
LOG_FILE="/var/log/backup-filesystem.log"
exec 1> >(tee -a "$LOG_FILE")
exec 2>&1

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*"
}

# Create exclude file if it doesn't exist
if [[ ! -f "$EXCLUDE_FILE" ]]; then
    cat > "$EXCLUDE_FILE" << EOF
*.tmp
*.cache
*.log
/home/*/.cache/
/var/cache/
/var/tmp/
/tmp/
*.iso
*.img
EOF
fi

# Create backup directory
mkdir -p "$BACKUP_DEST"

log "Starting filesystem backup"

# Create backup filename
BACKUP_FILE="$BACKUP_DEST/filesystem_$(date +%Y%m%d_%H%M%S).tar.gz"

# Create tar archive with compression
log "Creating compressed archive: $BACKUP_FILE"

if tar -czf "$BACKUP_FILE" \
    --exclude-from="$EXCLUDE_FILE" \
    --one-file-system \
    --preserve-permissions \
    --preserve-order \
    --totals \
    "${SOURCE_DIRS[@]}" 2>&1; then

    log "Archive created successfully: $BACKUP_FILE"

    # Check backup size
    BACKUP_SIZE=$(du -h "$BACKUP_FILE" | cut -f1)
    log "Backup size: $BACKUP_SIZE"

    # Verify archive integrity
    if tar -tzf "$BACKUP_FILE" >/dev/null 2>&1; then
        log "Archive integrity verified"
    else
```

```
        log "ERROR: Archive integrity check failed"
        exit 1
    fi

    # Copy to remote location
    if [[ -n "$REMOTE_DEST" ]]; then
        log "Copying backup to remote location: $REMOTE_DEST"
        if rsync -avz --progress "$BACKUP_FILE" "$REMOTE_DEST/"; then
            log "Backup copied to remote location successfully"
        else
            log "ERROR: Failed to copy backup to remote location"
        fi
    fi

else
    log "ERROR: Failed to create archive"
    exit 1
fi

# Clean up old backups
log "Cleaning up old backups (older than $RETENTION_DAYS days)"
find "$BACKUP_DEST" -name "filesystem_*.tar.gz" -mtime +$RETENTION_DAYS -delete

# Clean up remote backups
if [[ -n "$REMOTE_DEST" ]]; then
    ssh "${REMOTE_DEST%:*}" "find ${REMOTE_DEST#*:} -name 'filesystem_*.tar.gz' -
mtime +$RETENTION_DAYS -delete"
fi

log "Filesystem backup completed"

# Generate backup report
REPORT_FILE="/tmp/backup-report-$(date +%Y%m%d).txt"
cat > "$REPORT_FILE" << EOF
Filesystem Backup Report
========================
Date: $(date)
Hostname: $(hostname)
Backup File: $BACKUP_FILE
Backup Size: $BACKUP_SIZE
Source Directories: ${SOURCE_DIRS[*]}
Retention: $RETENTION_DAYS days
Remote Destination: $REMOTE_DEST

Backup Status: SUCCESS
EOF

# Send email report
if command -v mail &> /dev/null; then
    mail -s "Filesystem Backup Report - $(hostname)" admin@example.com <
"$REPORT_FILE"
fi

rm -f "$REPORT_FILE"
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between cron and systemd timers?**

   ▶ Answer

   Systemd timers offer better logging, dependency management, and integration with systemd services. They also support more flexible scheduling options and better error handling compared to traditional cron jobs.

2. **How do you ensure a script runs only one instance at a time?**

   ▶ Answer

   Use a lock file mechanism with PID checking, or use `flock` command to create file locks that prevent multiple instances from running simultaneously.

3. **What's the purpose of the `set -euo pipefail` command in bash scripts?**

   ▶ Answer
   - `set -e`: Exit immediately if any command fails
   - `set -u`: Treat unset variables as errors
   - `set -o pipefail`: Make pipelines fail if any command in the pipeline fails

4. **How do you securely store passwords in Ansible?**

   ▶ Answer

   Use Ansible Vault to encrypt sensitive data: `ansible-vault create secrets.yml` or `ansible-vault encrypt_string 'password' --name 'vault_mysql_password'`

Hands-On Challenges

Challenge 1: Complete Automation Suite

```
# Create a comprehensive automation suite:
# - System monitoring with automated alerts
# - Automated backup system with encryption
# - Log rotation and analysis
# - Security hardening automation
# - Performance optimization scripts
# - Disaster recovery procedures
```

Challenge 2: Infrastructure as Code

```
# Implement Infrastructure as Code:
# - Create Ansible playbooks for complete infrastructure
# - Implement configuration management
# - Set up automated testing and validation
# - Create rollback procedures
# - Implement blue-green deployment automation
```

Challenge 3: Advanced Monitoring Automation

```
# Build advanced monitoring automation:
# - Custom metrics collection
# - Predictive alerting based on trends
# - Automated remediation scripts
# - Integration with external services
# - Comprehensive reporting and dashboards
```

🚀 Next Steps

Congratulations! You've mastered automation and scripting. You can now:

- Write professional bash scripts with proper error handling
- Automate tasks using cron and systemd timers
- Manage infrastructure with Ansible
- Implement automated backup and monitoring systems
- Create comprehensive automation workflows
- Handle complex system administration tasks programmatically

Ready for security and firewalls? Continue to 15-troubleshooting-debugging.md to learn advanced troubleshooting and debugging techniques.

> **Pro Tip**: Start small with automation - automate one task at a time, test thoroughly, and gradually build more complex workflows. Always include proper logging, error handling, and rollback procedures! 🤖

🔍 Troubleshooting & Debugging: Advanced Problem-Solving Techniques

> **Master systematic troubleshooting, debugging tools, and problem resolution strategies**

📖 What You'll Learn

Troubleshooting is a critical skill for any system administrator. This chapter covers comprehensive problem-solving techniques:

- Systematic troubleshooting methodology

- Advanced debugging tools and techniques
- Network troubleshooting and analysis
- System performance debugging
- Application and service debugging
- Log analysis and correlation
- Root cause analysis techniques
- Documentation and knowledge management
- Preventive measures and monitoring

🌍 Why This Matters

Critical applications:

- **Rapid Problem Resolution**: Minimize downtime and service disruption
- **Root Cause Analysis**: Prevent recurring issues
- **System Reliability**: Maintain stable and predictable systems
- **Cost Reduction**: Reduce operational costs through efficient problem resolution
- **Knowledge Building**: Create organizational knowledge base for future issues

🔧 Systematic Troubleshooting Methodology

The ITIL Problem-Solving Process

```
# 1. IDENTIFY - What is the problem?
# 2. INVESTIGATE - Gather information
# 3. DIAGNOSE - Analyze the data
# 4. RESOLVE - Implement solution
# 5. VERIFY - Confirm resolution
# 6. DOCUMENT - Record findings

#!/bin/bash
# troubleshooting-template.sh - Systematic troubleshooting framework

TROUBLE_LOG="/var/log/troubleshooting-$(date +%Y%m%d_%H%M%S).log"
ISSUE_ID="ISSUE-$(date +%Y%m%d-%H%M%S)"

log_step() {
    local step="$1"
    local description="$2"
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] [$ISSUE_ID] [$step] $description" | tee -
a "$TROUBLE_LOG"
}

# Step 1: Identify the problem
identify_problem() {
    log_step "IDENTIFY" "Problem identification started"

    echo "Problem Description:"
    echo "- What is happening?"
    echo "- When did it start?"
    echo "- Who is affected?"
    echo "- What is the impact?"

    # Gather initial symptoms
    log_step "IDENTIFY" "System uptime: $(uptime)"
    log_step "IDENTIFY" "Load average: $(cat /proc/loadavg)"
    log_step "IDENTIFY" "Memory usage: $(free -h | grep Mem)"
    log_step "IDENTIFY" "Disk usage: $(df -h / | tail -1)"

    # Check recent system changes
    log_step "IDENTIFY" "Recent package changes:"
    grep "$(date +%Y-%m-%d)" /var/log/dpkg.log | tail -10 | tee -a "$TROUBLE_LOG"

    # Check system logs for errors
    log_step "IDENTIFY" "Recent system errors:"
    journalctl --since "1 hour ago" --priority=err | tail -20 | tee -a
"$TROUBLE_LOG"
}

# Step 2: Investigate and gather information
investigate_issue() {
    log_step "INVESTIGATE" "Information gathering started"

    # System information
    log_step "INVESTIGATE" "Hostname: $(hostname)"
    log_step "INVESTIGATE" "Kernel: $(uname -r)"
    log_step "INVESTIGATE" "OS: $(lsb_release -d | cut -f2)"

    # Network connectivity
    log_step "INVESTIGATE" "Network connectivity test:"
    if ping -c 3 8.8.8.8 &>/dev/null; then
        log_step "INVESTIGATE" "External connectivity: OK"
    else
        log_step "INVESTIGATE" "External connectivity: FAILED"
    fi

    # Service status
    log_step "INVESTIGATE" "Critical services status:"
    for service in ssh nginx apache2 mysql postgresql; do
        if systemctl is-active --quiet "$service" 2>/dev/null; then
            log_step "INVESTIGATE" "Service $service: RUNNING"
        else
            log_step "INVESTIGATE" "Service $service: NOT RUNNING"
        fi
    done

    # Resource utilization
    log_step "INVESTIGATE" "Top CPU consumers:"
    ps aux --sort=-%cpu | head -10 | tee -a "$TROUBLE_LOG"

    log_step "INVESTIGATE" "Top memory consumers:"
    ps aux --sort=-%mem | head -10 | tee -a "$TROUBLE_LOG"

    # Disk I/O
    log_step "INVESTIGATE" "Disk I/O statistics:"
    iostat -x 1 3 | tee -a "$TROUBLE_LOG"
}

# Step 3: Diagnose the issue
diagnose_issue() {
    log_step "DIAGNOSE" "Analysis started"

    # Analyze patterns in logs
    log_step "DIAGNOSE" "Analyzing error patterns:"

    # Common error patterns
    local error_patterns=(
        "out of memory"
        "disk full"
        "connection refused"
        "timeout"
        "permission denied"
        "segmentation fault"
        "kernel panic"
    )

    for pattern in "${error_patterns[@]}"; do
        local count=$(journalctl --since "1 hour ago" | grep -i "$pattern" | wc -
l)
        if [[ $count -gt 0 ]]; then
            log_step "DIAGNOSE" "Found $count occurrences of: $pattern"
        fi
    done

    # Check for resource exhaustion
    local mem_usage=$(free | awk 'NR==2{printf "%.1f", $3*100/$2}')
    local disk_usage=$(df / | awk 'NR==2{print $5}' | sed 's/%//')
    local load_avg=$(uptime | awk -F'load average:' '{print $2}' | awk '{print
$1}' | sed 's/,//')

    log_step "DIAGNOSE" "Resource analysis:"
    log_step "DIAGNOSE" "Memory usage: ${mem_usage}%"
    log_step "DIAGNOSE" "Disk usage: ${disk_usage}%"
    log_step "DIAGNOSE" "Load average: $load_avg"

    # Threshold checks
    if (( $(echo "$mem_usage > 90" | bc -l) )); then
        log_step "DIAGNOSE" "CRITICAL: High memory usage detected"
    fi

    if [[ $disk_usage -gt 90 ]]; then
        log_step "DIAGNOSE" "CRITICAL: High disk usage detected"
    fi

    if (( $(echo "$load_avg > 4.0" | bc -l) )); then
        log_step "DIAGNOSE" "WARNING: High load average detected"
    fi
}
```

```
# Step 4: Implement resolution
resolve_issue() {
    log_step "RESOLVE" "Resolution implementation started"

    echo "Common resolution steps:"
    echo "1. Restart affected services"
    echo "2. Clear temporary files"
    echo "3. Adjust configuration"
    echo "4. Apply patches/updates"
    echo "5. Scale resources"

    # Example automated fixes
    # Clear temporary files if disk is full
    local disk_usage=$(df / | awk 'NR==2{print $5}' | sed 's/%//')
    if [[ $disk_usage -gt 90 ]]; then
        log_step "RESOLVE" "Attempting to clear temporary files"
        find /tmp -type f -atime +7 -delete 2>/dev/null
        find /var/tmp -type f -atime +7 -delete 2>/dev/null
        apt-get clean 2>/dev/null
        log_step "RESOLVE" "Temporary files cleared"
    fi
}

# Step 5: Verify resolution
verify_resolution() {
    log_step "VERIFY" "Resolution verification started"

    # Re-run initial checks
    log_step "VERIFY" "Post-resolution system status:"
    log_step "VERIFY" "Load average: $(cat /proc/loadavg)"
    log_step "VERIFY" "Memory usage: $(free -h | grep Mem)"
    log_step "VERIFY" "Disk usage: $(df -h / | tail -1)"

    # Test functionality
    if ping -c 3 8.8.8.8 &>/dev/null; then
        log_step "VERIFY" "Network connectivity: VERIFIED"
    else
        log_step "VERIFY" "Network connectivity: STILL FAILING"
    fi
}

# Step 6: Document findings
document_findings() {
    log_step "DOCUMENT" "Documentation started"

    cat >> "$TROUBLE_LOG" << EOF

=== TROUBLESHOOTING SUMMARY ===
Issue ID: $ISSUE_ID
Date: $(date)
Duration: [TO BE FILLED]
Severity: [TO BE FILLED]
Root Cause: [TO BE FILLED]
Resolution: [TO BE FILLED]
Preventive Measures: [TO BE FILLED]
Lessons Learned: [TO BE FILLED]

=== FOLLOW-UP ACTIONS ===
- [ ] Monitor for recurrence
- [ ] Update documentation
- [ ] Implement preventive measures
- [ ] Review and improve monitoring
EOF

    log_step "DOCUMENT" "Troubleshooting log saved to: $TROUBLE_LOG"
}

# Main execution
echo "Starting systematic troubleshooting for Issue ID: $ISSUE_ID"
identify_problem
investigate_issue
diagnose_issue
resolve_issue
verify_resolution
document_findings

echo "Troubleshooting completed. Log file: $TROUBLE_LOG"
```

🔍 Advanced Debugging Tools

System Call Tracing with strace

```
# Trace system calls for a running process
$ strace -p 1234
execve("/bin/ls", ["ls", "-la"], 0x7fff8b7e1d40 /* 23 vars */) = 0
brk(NULL)                               = 0x55a8c9e1a000
access("/etc/ld.so.noinherit", F_OK)      = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) =
0x7f8b5c9e1000

# Trace specific system calls
$ strace -e trace=open,read,write -p 1234

# Trace file operations
$ strace -e trace=file ls /home

# Trace network operations
$ strace -e trace=network curl http://example.com

# Save trace to file
$ strace -o trace.log -p 1234

# Trace with timestamps
$ strace -t -p 1234
```

```
# Trace child processes
$ strace -f -p 1234

# Count system calls
$ strace -c ls /home
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 28.57    0.000020           4         5           mmap
 21.43    0.000015           3         5           close
 14.29    0.000010           2         5           fstat
 14.29    0.000010           5         2           getdents64

# Advanced strace usage
#!/bin/bash
# debug-application.sh

APP_PID="$1"
OUTPUT_DIR="/tmp/debug-$(date +%Y%m%d_%H%M%S)"

if [[ -z "$APP_PID" ]]; then
    echo "Usage: $0 <PID>"
    exit 1
fi

mkdir -p "$OUTPUT_DIR"

echo "Starting comprehensive debugging for PID: $APP_PID"
echo "Output directory: $OUTPUT_DIR"

# System call trace
echo "Collecting system call trace..."
strace -o "$OUTPUT_DIR/syscalls.log" -f -t -p "$APP_PID" &
STRACE_PID=$!

# Library call trace
echo "Collecting library call trace..."
ltrace -o "$OUTPUT_DIR/libcalls.log" -f -t -p "$APP_PID" &
LTRACE_PID=$!

# Memory usage
echo "Monitoring memory usage..."
while kill -0 "$APP_PID" 2>/dev/null; do
    echo "$(date '+%Y-%m-%d %H:%M:%S') $(cat /proc/$APP_PID/status | grep VmRSS)"
>> "$OUTPUT_DIR/memory.log"
    sleep 5
done &
MEMORY_PID=$!

# File descriptor usage
echo "Monitoring file descriptors..."
while kill -0 "$APP_PID" 2>/dev/null; do
    echo "$(date '+%Y-%m-%d %H:%M:%S') FDs: $(ls /proc/$APP_PID/fd | wc -l)" >>
"$OUTPUT_DIR/fds.log"
```

```
    sleep 5
done &
FD_PID=$!

echo "Debugging started. Press Ctrl+C to stop."

# Cleanup function
cleanup() {
    echo "Stopping debugging..."
    kill $STRACE_PID $LTRACE_PID $MEMORY_PID $FD_PID 2>/dev/null
    echo "Debug data saved to: $OUTPUT_DIR"
}

trap cleanup SIGINT SIGTERM

# Wait for user interrupt
wait
```

Process Debugging with gdb

```
# Attach to running process
$ gdb -p 1234
(gdb) bt                     # Show backtrace
(gdb) info threads           # Show all threads
(gdb) thread 2               # Switch to thread 2
(gdb) print variable_name    # Print variable value
(gdb) continue               # Continue execution
(gdb) detach                 # Detach from process

# Debug core dump
$ gdb /path/to/binary /path/to/core
(gdb) bt                     # Show backtrace at crash
(gdb) info registers         # Show register values
(gdb) disassemble           # Show assembly code

# Generate core dump
$ ulimit -c unlimited        # Enable core dumps
$ echo '/tmp/core.%e.%p' | sudo tee /proc/sys/kernel/core_pattern

# Debug script for application crashes
#!/bin/bash
# crash-debugger.sh

APP_NAME="$1"
CORE_DIR="/tmp"
DEBUG_LOG="/var/log/crash-debug.log"

if [[ -z "$APP_NAME" ]]; then
    echo "Usage: $0 <application_name>"
    exit 1
fi
```

```
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*" | tee -a "$DEBUG_LOG"
}

# Find the binary
BINARY_PATH=$(which "$APP_NAME")
if [[ -z "$BINARY_PATH" ]]; then
    log "ERROR: Binary not found for $APP_NAME"
    exit 1
fi

# Find core dump
CORE_FILE=$(find "$CORE_DIR" -name "core.$APP_NAME.*" -type f -printf '%T@ %p\n' |
sort -n | tail -1 | cut -d' ' -f2-)

if [[ -z "$CORE_FILE" ]]; then
    log "ERROR: No core dump found for $APP_NAME"
    exit 1
fi

log "Analyzing crash for $APP_NAME"
log "Binary: $BINARY_PATH"
log "Core dump: $CORE_FILE"

# Generate debug report
DEBUG_REPORT="/tmp/crash-report-$(date +%Y%m%d_%H%M%S).txt"

cat > "$DEBUG_REPORT" << EOF
Crash Analysis Report
====================
Application: $APP_NAME
Binary: $BINARY_PATH
Core Dump: $CORE_FILE
Analysis Date: $(date)

Backtrace:
EOF

# Get backtrace
gdb -batch -ex "bt" -ex "quit" "$BINARY_PATH" "$CORE_FILE" >> "$DEBUG_REPORT" 2>&1

echo "" >> "$DEBUG_REPORT"
echo "Thread Information:" >> "$DEBUG_REPORT"
gdb -batch -ex "info threads" -ex "quit" "$BINARY_PATH" "$CORE_FILE" >>
"$DEBUG_REPORT" 2>&1

echo "" >> "$DEBUG_REPORT"
echo "Register Information:" >> "$DEBUG_REPORT"
gdb -batch -ex "info registers" -ex "quit" "$BINARY_PATH" "$CORE_FILE" >>
"$DEBUG_REPORT" 2>&1

log "Debug report generated: $DEBUG_REPORT"
```

```
# Send alert
if command -v mail &> /dev/null; then
    mail -s "Application Crash: $APP_NAME" admin@example.com < "$DEBUG_REPORT"
fi
```

Network Debugging

```
# Comprehensive network debugging script
#!/bin/bash
# network-debug.sh

DEBUG_LOG="/tmp/network-debug-$(date +%Y%m%d_%H%M%S).log"

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*" | tee -a "$DEBUG_LOG"
}

log "Starting network debugging"

# Basic connectivity tests
log "=== BASIC CONNECTIVITY ==="
log "Testing localhost connectivity:"
if ping -c 3 127.0.0.1 &>/dev/null; then
    log "Localhost: OK"
else
    log "Localhost: FAILED"
fi

log "Testing gateway connectivity:"
GATEWAY=$(ip route | grep default | awk '{print $3}' | head -1)
if [[ -n "$GATEWAY" ]]; then
    if ping -c 3 "$GATEWAY" &>/dev/null; then
        log "Gateway ($GATEWAY): OK"
    else
        log "Gateway ($GATEWAY): FAILED"
    fi
else
    log "No default gateway found"
fi

log "Testing external connectivity:"
if ping -c 3 8.8.8.8 &>/dev/null; then
    log "External (8.8.8.8): OK"
else
    log "External (8.8.8.8): FAILED"
fi

log "Testing DNS resolution:"
if nslookup google.com &>/dev/null; then
    log "DNS resolution: OK"
else
```

```
    log "DNS resolution: FAILED"
fi

# Interface information
log "=== INTERFACE INFORMATION ==="
log "Network interfaces:"
ip addr show | tee -a "$DEBUG_LOG"

log "Routing table:"
ip route show | tee -a "$DEBUG_LOG"

log "ARP table:"
arp -a | tee -a "$DEBUG_LOG"

# Port and connection analysis
log "=== PORT AND CONNECTION ANALYSIS ==="
log "Listening ports:"
netstat -tuln | tee -a "$DEBUG_LOG"

log "Active connections:"
netstat -tun | head -20 | tee -a "$DEBUG_LOG"

log "Connection states summary:"
netstat -tun | awk 'NR>2 {print $6}' | sort | uniq -c | tee -a "$DEBUG_LOG"

# DNS configuration
log "=== DNS CONFIGURATION ==="
log "DNS servers:"
cat /etc/resolv.conf | tee -a "$DEBUG_LOG"

log "DNS resolution test:"
for dns in 8.8.8.8 1.1.1.1 208.67.222.222; do
    response_time=$(dig @"$dns" google.com | grep "Query time" | awk '{print $4}')
    if [[ -n "$response_time" ]]; then
        log "DNS $dns: ${response_time}ms"
    else
        log "DNS $dns: FAILED"
    fi
done

# Firewall status
log "=== FIREWALL STATUS ==="
if command -v ufw &> /dev/null; then
    log "UFW status:"
    ufw status verbose | tee -a "$DEBUG_LOG"
fi

if command -v iptables &> /dev/null; then
    log "iptables rules:"
    iptables -L -n | tee -a "$DEBUG_LOG"
fi

# Network performance
log "=== NETWORK PERFORMANCE ==="
log "Network interface statistics:"
cat /proc/net/dev | tee -a "$DEBUG_LOG"

log "Network errors:"
for interface in $(ip link show | grep -E '^[0-9]+:' | cut -d: -f2 | tr -d ' ');
do
    if [[ "$interface" != "lo" ]]; then
        errors=$(cat "/sys/class/net/$interface/statistics/rx_errors" 2>/dev/null)
        dropped=$(cat "/sys/class/net/$interface/statistics/rx_dropped"
2>/dev/null)
        log "Interface $interface: RX errors=$errors, dropped=$dropped"
    fi
done

# Advanced diagnostics
log "=== ADVANCED DIAGNOSTICS ==="
log "MTU discovery test:"
ping -M do -s 1472 -c 1 8.8.8.8 &>/dev/null
if [[ $? -eq 0 ]]; then
    log "MTU 1500: OK"
else
    log "MTU 1500: FRAGMENTATION NEEDED"
fi

log "Traceroute to 8.8.8.8:"
traceroute -n 8.8.8.8 | head -10 | tee -a "$DEBUG_LOG"

log "Network debugging completed. Log saved to: $DEBUG_LOG"

# Specific service debugging
debug_web_service() {
    local url="$1"
    local service_log="/tmp/web-debug-$(date +%Y%m%d_%H%M%S).log"

    log "Debugging web service: $url"

    # HTTP response test
    log "HTTP response test:"
    curl -I -s -w "HTTP Code: %{http_code}\nTotal Time: %{time_total}s\nConnect
Time: %{time_connect}s\nSSL Time: %{time_appconnect}s\n" "$url" | tee -a
"$service_log"

    # SSL certificate check
    if [[ "$url" =~ ^https ]]; then
        log "SSL certificate check:"
        echo | openssl s_client -connect "${url#https://}:443" -servername
"${url#https://}" 2>/dev/null | openssl x509 -noout -dates | tee -a "$service_log"
    fi

    # DNS resolution for the domain
    domain=$(echo "$url" | sed -E 's|^https?://([^/]+).*|\1|')
    log "DNS resolution for $domain:"
    dig "$domain" | tee -a "$service_log"
}
```

```
# Usage example
# debug_web_service "https://example.com"
```

📊 Log Analysis and Correlation

Advanced Log Analysis

```
#!/bin/bash
# log-analyzer.sh - Comprehensive log analysis tool

LOG_DIR="/var/log"
ANALYSIS_DIR="/tmp/log-analysis-$(date +%Y%m%d_%H%M%S)"
REPORT_FILE="$ANALYSIS_DIR/analysis-report.txt"

mkdir -p "$ANALYSIS_DIR"

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*" | tee -a "$REPORT_FILE"
}

log "Starting comprehensive log analysis"

# Error pattern analysis
analyze_error_patterns() {
    log "=== ERROR PATTERN ANALYSIS ==="

    local error_patterns=(
        "error"
        "fail"
        "exception"
        "timeout"
        "refused"
        "denied"
        "critical"
        "fatal"
    )

    for pattern in "${error_patterns[@]}"; do
        local count=$(find "$LOG_DIR" -name "*.log" -type f -exec grep -i
"$pattern" {} \; 2>/dev/null | wc -l)
        if [[ $count -gt 0 ]]; then
            log "Pattern '$pattern': $count occurrences"

            # Get recent occurrences
            log "Recent occurrences of '$pattern':"
            find "$LOG_DIR" -name "*.log" -type f -exec grep -i "$pattern" {} \;
2>/dev/null | tail -5 | while read line; do
                log "  $line"
            done
        fi
    done
}

# Time-based analysis
analyze_time_patterns() {
    log "=== TIME-BASED ANALYSIS ==="

    # Analyze log entries by hour
    log "Log entries by hour (last 24 hours):"
    for hour in {0..23}; do
        local hour_pattern=$(printf "%02d:" $hour)
        local count=$(journalctl --since "24 hours ago" | grep "$hour_pattern" |
wc -l)
        log "Hour $hour_pattern $count entries"
    done

    # Peak activity analysis
    log "Peak activity analysis:"
    journalctl --since "24 hours ago" --output=short-iso | awk '{print $1" "$2}' |
cut -d'T' -f2 | cut -d':' -f1 | sort | uniq -c | sort -nr | head -5 | while read
count hour; do
        log "Peak hour $hour:00 with $count entries"
    done
}

# Service-specific analysis
analyze_services() {
    log "=== SERVICE-SPECIFIC ANALYSIS ==="

    local services=("ssh" "nginx" "apache2" "mysql" "postgresql")

    for service in "${services[@]}"; do
        if systemctl is-enabled "$service" &>/dev/null; then
            log "Analyzing service: $service"

            # Service status
            local status=$(systemctl is-active "$service")
            log "  Status: $status"

            # Recent service logs
            local error_count=$(journalctl -u "$service" --since "24 hours ago" --
priority=err | wc -l)
            local warning_count=$(journalctl -u "$service" --since "24 hours ago"
--priority=warning | wc -l)

            log "  Errors (24h): $error_count"
            log "  Warnings (24h): $warning_count"

            # Service restarts
            local restart_count=$(journalctl -u "$service" --since "24 hours ago"
| grep -i "start\|restart" | wc -l)
            log "  Restarts (24h): $restart_count"

            if [[ $restart_count -gt 0 ]]; then
```

```
                log "  Recent restart events:"
                journalctl -u "$service" --since "24 hours ago" | grep -i
"start\|restart" | tail -3 | while read line; do
                    log "    $line"
                done
            fi
        fi
    done
}

# Security analysis
analyze_security() {
    log "=== SECURITY ANALYSIS ==="

    # Failed login attempts
    local failed_logins=$(grep "Failed password" /var/log/auth.log 2>/dev/null |
wc -l)
    log "Failed login attempts: $failed_logins"

    if [[ $failed_logins -gt 0 ]]; then
        log "Top failed login sources:"
        grep "Failed password" /var/log/auth.log 2>/dev/null | awk '{print $(NF-
3)}' | sort | uniq -c | sort -nr | head -5 | while read count ip; do
            log "  $ip: $count attempts"
        done
    fi

    # Sudo usage
    local sudo_count=$(grep "sudo:" /var/log/auth.log 2>/dev/null | wc -l)
    log "Sudo commands executed: $sudo_count"

    if [[ $sudo_count -gt 0 ]]; then
        log "Recent sudo activity:"
        grep "sudo:" /var/log/auth.log 2>/dev/null | tail -5 | while read line; do
            log "  $line"
        done
    fi

    # UFW blocks
    local ufw_blocks=$(grep "UFW BLOCK" /var/log/kern.log 2>/dev/null | wc -l)
    log "UFW blocked connections: $ufw_blocks"

    if [[ $ufw_blocks -gt 0 ]]; then
        log "Top blocked sources:"
        grep "UFW BLOCK" /var/log/kern.log 2>/dev/null | awk '{print $13}' | cut -
d'=' -f2 | sort | uniq -c | sort -nr | head -5 | while read count ip; do
            log "  $ip: $count blocks"
        done
    fi
}

# Performance analysis
analyze_performance() {
    log "=== PERFORMANCE ANALYSIS ==="
    # System load analysis
    log "System load analysis:"
    sar -q 1 1 | tail -1 | awk '{print "Load average: "$4" "$5" "$6}' | tee -a
"$REPORT_FILE"

    # Memory pressure
    local oom_kills=$(dmesg | grep -i "killed process" | wc -l)
    log "OOM kills detected: $oom_kills"

    if [[ $oom_kills -gt 0 ]]; then
        log "Recent OOM kills:"
        dmesg | grep -i "killed process" | tail -3 | while read line; do
            log "  $line"
        done
    fi

    # Disk I/O issues
    local io_errors=$(dmesg | grep -i "i/o error" | wc -l)
    log "I/O errors detected: $io_errors"

    if [[ $io_errors -gt 0 ]]; then
        log "Recent I/O errors:"
        dmesg | grep -i "i/o error" | tail -3 | while read line; do
            log "  $line"
        done
    fi
}

# Generate correlation matrix
generate_correlation() {
    log "=== EVENT CORRELATION ==="

    # Create timeline of events
    local timeline_file="$ANALYSIS_DIR/timeline.txt"

    # Collect events from different sources
    {
        journalctl --since "24 hours ago" --output=short-iso | awk '{print $1"
"$2" SYSTEM "$0}'
        grep "Failed password" /var/log/auth.log 2>/dev/null | awk '{print $1"
"$2" "$3" AUTH "$0}'
        grep "error\|Error\|ERROR" /var/log/nginx/error.log 2>/dev/null | awk
'{print $1" "$2" WEB "$0}'
    } | sort > "$timeline_file"

    log "Event timeline created: $timeline_file"

    # Identify event clusters
    log "Event clustering analysis:"
    awk '{
        timestamp = $1" "$2
        gsub(/[0-9]{2}:[0-9]{2}:[0-9]{2}/, "XX:XX:XX", timestamp)
        count[timestamp]++
    }
    END {
        for (ts in count) {
            if (count[ts] > 5) {
                print "High activity period: "ts" ("count[ts]" events)"
            }
        }
    }' "$timeline_file" | tee -a "$REPORT_FILE"
}

# Main execution
analyze_error_patterns
analyze_time_patterns
analyze_services
analyze_security
analyze_performance
generate_correlation

log "Log analysis completed. Report saved to: $REPORT_FILE"

# Generate summary
log "=== ANALYSIS SUMMARY ==="
log "Analysis completed at: $(date)"
log "Total log files analyzed: $(find "$LOG_DIR" -name "*.log" -type f | wc -l)"
log "Analysis artifacts saved to: $ANALYSIS_DIR"

# Send report if email is configured
if command -v mail &> /dev/null; then
    mail -s "Log Analysis Report - $(hostname)" admin@example.com < "$REPORT_FILE"
    log "Report emailed to admin@example.com"
fi
```

Real-time Log Monitoring

```
#!/bin/bash
# real-time-monitor.sh - Real-time log monitoring with alerting

MONITOR_LOGS=(
    "/var/log/syslog"
    "/var/log/auth.log"
    "/var/log/nginx/error.log"
    "/var/log/apache2/error.log"
    "/var/log/mysql/error.log"
)

ALERT_PATTERNS=(
    "CRITICAL|critical|Critical"
    "ERROR|error|Error"
    "FAILED|failed|Failed"
    "DENIED|denied|Denied"
    "TIMEOUT|timeout|Timeout"
```

```
)

ALERT_EMAIL="admin@example.com"
ALERT_THRESHOLD=5  # Alert after 5 occurrences in 5 minutes
CHECK_INTERVAL=60  # Check every minute

# Alert tracking
declare -A alert_counts
declare -A last_alert_time

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*"
}

send_alert() {
    local pattern="$1"
    local count="$2"
    local logfile="$3"
    local sample_lines="$4"

    local subject="Alert: Pattern '$pattern' detected $count times"
    local body="Alert Details:
Pattern: $pattern
Count: $count
Log file: $logfile
Time: $(date)

Sample log entries:
$sample_lines"

    echo "$body" | mail -s "$subject" "$ALERT_EMAIL"
    log "ALERT SENT: $subject"
}

monitor_logs() {
    local current_time=$(date +%s)

    for logfile in "${MONITOR_LOGS[@]}"; do
        if [[ ! -f "$logfile" ]]; then
            continue
        fi

        for pattern in "${ALERT_PATTERNS[@]}"; do
            # Count occurrences in the last 5 minutes
            local count=$(tail -1000 "$logfile" | awk -v pattern="$pattern" -v
cutoff="$(date -d '5 minutes ago' '+%b %d %H:%M')" '
                $0 ~ cutoff && $0 ~ pattern { count++ }
                END { print count+0 }
            ')

            if [[ $count -ge $ALERT_THRESHOLD ]]; then
                local alert_key="${logfile}_${pattern}"
                local last_alert=${last_alert_time[$alert_key]:-0}
                # Only send alert if it's been more than 30 minutes since last
alert
                if [[ $((current_time - last_alert)) -gt 1800 ]]; then
                    local sample_lines=$(tail -1000 "$logfile" | grep -E
"$pattern" | tail -5)
                    send_alert "$pattern" "$count" "$logfile" "$sample_lines"
                    last_alert_time[$alert_key]=$current_time
                fi
            fi
        done
    done
}

# Real-time monitoring with tail
real_time_monitor() {
    log "Starting real-time log monitoring"

    # Create named pipes for each log file
    local pipes=()
    for logfile in "${MONITOR_LOGS[@]}"; do
        if [[ -f "$logfile" ]]; then
            local pipe="/tmp/monitor_$(basename "$logfile").pipe"
            mkfifo "$pipe" 2>/dev/null
            tail -f "$logfile" > "$pipe" &
            pipes+=("$pipe")
        fi
    done

    # Monitor all pipes simultaneously
    while true; do
        for pipe in "${pipes[@]}"; do
            if read -t 1 line < "$pipe"; then
                # Check line against alert patterns
                for pattern in "${ALERT_PATTERNS[@]}"; do
                    if echo "$line" | grep -qE "$pattern"; then
                        log "PATTERN MATCH: $pattern in $line"

                        # Immediate alert for critical patterns
                        if echo "$pattern" | grep -qi "critical"; then
                            echo "CRITICAL ALERT: $line" | mail -s "CRITICAL:
Immediate attention required" "$ALERT_EMAIL"
                        fi
                    fi
                done
            fi
        done
    done
}

# Cleanup function
cleanup() {
    log "Cleaning up monitoring processes"
    pkill -f "tail -f"
    rm -f /tmp/monitor_*.pipe
    exit 0
}

trap cleanup SIGINT SIGTERM

# Main execution
case "${1:-periodic}" in
    "realtime")
        real_time_monitor
        ;;
    "periodic")
        log "Starting periodic log monitoring"
        while true; do
            monitor_logs
            sleep $CHECK_INTERVAL
        done
        ;;
    "once")
        log "Running single log check"
        monitor_logs
        ;;
    *)
        echo "Usage: $0 [realtime|periodic|once]"
        echo "  realtime - Monitor logs in real-time"
        echo "  periodic - Check logs periodically (default)"
        echo "  once     - Run single check"
        exit 1
        ;;
esac
```

🧠 Knowledge Check

Quick Quiz

1. **What's the difference between strace and ltrace?**

   ▶ Answer

   strace traces system calls made by a process to the kernel, while ltrace traces library function calls made by a process to shared libraries.

2. **How do you generate a core dump for a running process?**

   ▶ Answer

   Use `gcore <PID>` to generate a core dump, or send a SIGQUIT signal with `kill -3 <PID>` (if the application handles it properly).

3. **What information does the /proc filesystem provide for troubleshooting?**

   ▶ Answer

/proc provides real-time information about processes, system resources, kernel parameters, memory usage, file descriptors, network connections, and system statistics.

4. **How do you correlate events across multiple log files?**

   ▶ Answer

   Use timestamps to create a unified timeline, employ log aggregation tools like ELK stack, or create custom scripts that merge and sort log entries by timestamp.

Hands-On Challenges

Challenge 1: Complete Debugging Suite

```
# Create a comprehensive debugging toolkit:
# - Automated problem detection and classification
# - Multi-layer debugging (system, network, application)
# - Real-time monitoring with intelligent alerting
# - Automated log correlation and analysis
# - Performance bottleneck identification
# - Root cause analysis automation
```

Challenge 2: Incident Response System

```
# Build an incident response system:
# - Automated incident detection and classification
# - Escalation procedures and notifications
# - Evidence collection and preservation
# - Recovery procedures and rollback capabilities
# - Post-incident analysis and reporting
```

Challenge 3: Predictive Problem Detection

```
# Implement predictive problem detection:
# - Trend analysis and anomaly detection
# - Machine learning for pattern recognition
# - Proactive alerting before issues occur
# - Capacity planning and resource prediction
# - Automated preventive actions
```

🚀 Next Steps

Excellent! You've mastered troubleshooting and debugging. You can now:

- Apply systematic troubleshooting methodologies
- Use advanced debugging tools (strace, gdb, network analyzers)
- Perform comprehensive log analysis and correlation

- Implement real-time monitoring and alerting
- Conduct root cause analysis effectively
- Build automated debugging and incident response systems
- Create comprehensive documentation and knowledge bases

Ready for the final chapter? Continue to 16-best-practices.md to learn industry best practices and advanced techniques for Linux and networking mastery.

> **Pro Tip**: Great troubleshooters are made, not born. Practice systematic approaches, build comprehensive toolkits, and always document your findings. The key is to remain calm, methodical, and persistent! 🔍

🏆 Best Practices: Linux & Networking Mastery

> **Industry best practices, advanced techniques, and professional workflows for Linux and networking excellence**

📖 What You'll Learn

This final chapter consolidates everything into professional best practices and advanced techniques:

- System administration best practices
- Security hardening and compliance
- Performance optimization strategies
- Disaster recovery and business continuity
- Documentation and knowledge management
- Team collaboration and workflows
- Continuous learning and skill development
- Career advancement strategies
- Industry standards and certifications

🌍 Why This Matters

Professional excellence requires:

- **Consistency**: Standardized approaches and procedures
- **Reliability**: Predictable and stable systems
- **Security**: Robust protection against threats
- **Efficiency**: Optimized performance and resource utilization
- **Scalability**: Systems that grow with business needs
- **Maintainability**: Easy to manage and troubleshoot

🔧 System Administration Best Practices

Infrastructure as Code (IaC)

```yaml
# ansible-playbook: infrastructure-setup.yml
---
- name: Complete Infrastructure Setup
  hosts: all
  become: yes
  vars:
    # Environment-specific variables
    environment: "{{ env | default('production') }}"
    backup_retention_days: 30
    monitoring_enabled: true
    security_hardening: true

  pre_tasks:
    - name: Validate environment
      assert:
        that:
          - environment in ['development', 'staging', 'production']
          - ansible_distribution == 'Ubuntu'
          - ansible_distribution_major_version|int >= 20
        fail_msg: "Environment validation failed"

    - name: Create backup directory
      file:
        path: /backup/{{ inventory_hostname }}
        state: directory
        mode: "0750"
        owner: backup
        group: backup

  roles:
    - role: common
      tags: ["common", "base"]
    - role: security
      tags: ["security", "hardening"]
      when: security_hardening
    - role: monitoring
      tags: ["monitoring", "observability"]
      when: monitoring_enabled
    - role: backup
      tags: ["backup", "disaster-recovery"]

  post_tasks:
    - name: Verify system state
      include_tasks: verify-system.yml
      tags: ["verification", "testing"]

    - name: Generate system report
      template:
        src: system-report.j2
        dest: /var/log/system-setup-{{ ansible_date_time.epoch }}.log
      tags: ["documentation", "reporting"]
```

```yaml
# roles/common/tasks/main.yml
---
- name: Update package cache
  apt:
    update_cache: yes
    cache_valid_time: 3600
  tags: ["packages"]

- name: Install essential packages
  apt:
    name:
      - curl
      - wget
      - vim
      - htop
      - iotop
      - nload
      - tree
      - git
      - unzip
      - software-properties-common
      - apt-transport-https
      - ca-certificates
      - gnupg
      - lsb-release
    state: present
  tags: ["packages"]

- name: Configure timezone
  timezone:
    name: "{{ system_timezone | default('UTC') }}"
  notify: restart rsyslog
  tags: ["system"]

- name: Configure NTP
  template:
    src: ntp.conf.j2
    dest: /etc/ntp.conf
    backup: yes
  notify: restart ntp
  tags: ["system", "time"]

- name: Create system users
  user:
    name: "{{ item.name }}"
    groups: "{{ item.groups | default([]) }}"
    shell: "{{ item.shell | default('/bin/bash') }}"
    create_home: yes
    state: present
  loop: "{{ system_users | default([]) }}"
  tags: ["users"]

- name: Configure SSH keys
```

```
    authorized_key:
      user: "{{ item.0.name }}"
      key: "{{ item.1 }}"
      state: present
    loop: "{{ system_users | subelements('ssh_keys', skip_missing=True) }}"
    tags: ["users", "ssh"]

  - name: Configure system limits
    pam_limits:
      domain: "{{ item.domain }}"
      limit_type: "{{ item.type }}"
      limit_item: "{{ item.item }}"
      value: "{{ item.value }}"
    loop:
      - { domain: "*", type: "soft", item: "nofile", value: "65536" }
      - { domain: "*", type: "hard", item: "nofile", value: "65536" }
      - { domain: "*", type: "soft", item: "nproc", value: "32768" }
      - { domain: "*", type: "hard", item: "nproc", value: "32768" }
    tags: ["system", "limits"]

  - name: Configure kernel parameters
    sysctl:
      name: "{{ item.name }}"
      value: "{{ item.value }}"
      state: present
      reload: yes
    loop:
      - { name: "vm.swappiness", value: "10" }
      - { name: "vm.dirty_ratio", value: "15" }
      - { name: "vm.dirty_background_ratio", value: "5" }
      - { name: "net.core.rmem_max", value: "16777216" }
      - { name: "net.core.wmem_max", value: "16777216" }
      - { name: "net.ipv4.tcp_window_scaling", value: "1" }
      - { name: "net.ipv4.tcp_timestamps", value: "1" }
      - { name: "net.ipv4.tcp_sack", value: "1" }
    tags: ["system", "kernel", "performance"]
```

Configuration Management Standards

```
#!/bin/bash
# config-management-standards.sh

# Configuration file standards
CONFIG_DIR="/etc/myapp"
CONFIG_BACKUP_DIR="/etc/myapp/backups"
CONFIG_TEMPLATE_DIR="/etc/myapp/templates"

# Ensure proper directory structure
create_config_structure() {
    local dirs=(
        "$CONFIG_DIR"
        "$CONFIG_BACKUP_DIR"
        "$CONFIG_TEMPLATE_DIR"
        "$CONFIG_DIR/conf.d"
        "$CONFIG_DIR/ssl"
        "$CONFIG_DIR/scripts"
    )

    for dir in "${dirs[@]}"; do
        mkdir -p "$dir"
        chown root:root "$dir"
        chmod 755 "$dir"
    done

    # Secure SSL directory
    chmod 700 "$CONFIG_DIR/ssl"
}

# Configuration file management
manage_config_file() {
    local config_file="$1"
    local template_file="$2"
    local backup_suffix="$(date +%Y%m%d_%H%M%S)"

    # Validate inputs
    if [[ ! -f "$template_file" ]]; then
        echo "Error: Template file not found: $template_file"
        return 1
    fi

    # Backup existing configuration
    if [[ -f "$config_file" ]]; then
        cp "$config_file" "${CONFIG_BACKUP_DIR}/$(basename
"$config_file").${backup_suffix}"
        echo "Backed up existing config: $config_file"
    fi

    # Validate new configuration
    if validate_config "$template_file"; then
        cp "$template_file" "$config_file"
        chown root:root "$config_file"
        chmod 644 "$config_file"
        echo "Configuration updated: $config_file"
        return 0
    else
        echo "Error: Configuration validation failed"
        return 1
    fi
}

# Configuration validation
validate_config() {
    local config_file="$1"

    # Syntax validation (example for nginx)
    if [[ "$config_file" =~ nginx ]]; then
        nginx -t -c "$config_file" 2>/dev/null
        return $?
    fi

    # Syntax validation (example for apache)
    if [[ "$config_file" =~ apache ]]; then
        apache2ctl -t -f "$config_file" 2>/dev/null
        return $?
    fi

    # Generic validation (check for basic syntax)
    if [[ -f "$config_file" ]]; then
        # Check for common syntax errors
        if grep -q "^[[:space:]]*$" "$config_file" && \
           ! grep -q "[{}].*[{}]" "$config_file"; then
            return 0
        fi
    fi

    return 1
}

# Configuration rollback
rollback_config() {
    local config_file="$1"
    local backup_file="$2"

    if [[ -f "$backup_file" ]]; then
        cp "$backup_file" "$config_file"
        echo "Configuration rolled back: $config_file"

        # Restart associated service
        local service_name=$(basename "$config_file" | cut -d'.' -f1)
        if systemctl is-enabled "$service_name" &>/dev/null; then
            systemctl restart "$service_name"
            echo "Service restarted: $service_name"
        fi
    else
        echo "Error: Backup file not found: $backup_file"
        return 1
    fi
}

# Configuration audit
audit_configurations() {
    local audit_log="/var/log/config-audit-$(date +%Y%m%d).log"

    echo "Configuration Audit Report - $(date)" > "$audit_log"
    echo "==========================================" >> "$audit_log"

    # Check file permissions
    echo "File Permissions:" >> "$audit_log"
    find "$CONFIG_DIR" -type f -exec ls -la {} \; >> "$audit_log"
    # Check for sensitive data
    echo "\nSensitive Data Check:" >> "$audit_log"
    grep -r "password\|secret\|key" "$CONFIG_DIR" --exclude-dir=ssl >>
"$audit_log" 2>/dev/null || echo "No sensitive data found in configs" >>
"$audit_log"

    # Check configuration syntax
    echo "\nConfiguration Validation:" >> "$audit_log"
    find "$CONFIG_DIR" -name "*.conf" -o -name "*.cfg" | while read config; do
        if validate_config "$config"; then
            echo "✓ Valid: $config" >> "$audit_log"
        else
            echo "✗ Invalid: $config" >> "$audit_log"
        fi
    done

    echo "Audit completed: $audit_log"
}

# Example usage
create_config_structure
# manage_config_file "/etc/nginx/nginx.conf"
"/etc/myapp/templates/nginx.conf.template"
# audit_configurations
```

🔒 Security Hardening Best Practices

Comprehensive Security Hardening

```
#!/bin/bash
# security-hardening.sh - Comprehensive security hardening script

SECURITY_LOG="/var/log/security-hardening.log"
BACKUP_DIR="/backup/security-$(date +%Y%m%d_%H%M%S)"

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*" | tee -a "$SECURITY_LOG"
}

# Create backup directory
mkdir -p "$BACKUP_DIR"

log "Starting comprehensive security hardening"

# 1. System Updates
harden_system_updates() {
    log "=== SYSTEM UPDATES ==="

    # Update package database
    apt update
    # Upgrade all packages
    apt upgrade -y

    # Remove unnecessary packages
    apt autoremove -y

    # Configure automatic security updates
    cat > /etc/apt/apt.conf.d/20auto-upgrades << EOF
APT::Periodic::Update-Package-Lists "1";
APT::Periodic::Unattended-Upgrade "1";
APT::Periodic::AutocleanInterval "7";
EOF

    # Configure unattended upgrades
    cat > /etc/apt/apt.conf.d/50unattended-upgrades << EOF
Unattended-Upgrade::Allowed-Origins {
    "\${distro_id}:\${distro_codename}-security";
    "\${distro_id}ESMApps:\${distro_codename}-apps-security";
    "\${distro_id}ESM:\${distro_codename}-infra-security";
};
Unattended-Upgrade::AutoFixInterruptedDpkg "true";
Unattended-Upgrade::MinimalSteps "true";
Unattended-Upgrade::Remove-Unused-Dependencies "true";
Unattended-Upgrade::Automatic-Reboot "false";
EOF

    log "System updates configured"
}

# 2. SSH Hardening
harden_ssh() {
    log "=== SSH HARDENING ==="

    # Backup original SSH config
    cp /etc/ssh/sshd_config "$BACKUP_DIR/sshd_config.backup"

    # Create hardened SSH configuration
    cat > /etc/ssh/sshd_config << EOF
# SSH Hardened Configuration
Port 2222
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
HostKey /etc/ssh/ssh_host_ed25519_key

# Authentication
PermitRootLogin no
PasswordAuthentication no
PubkeyAuthentication yes
AuthenticationMethods publickey
PermitEmptyPasswords no
ChallengeResponseAuthentication no
UsePAM yes
```

```
# Security settings
X11Forwarding no
AllowTcpForwarding no
AllowAgentForwarding no
PermitTunnel no
GatewayPorts no
PermitUserEnvironment no

# Connection settings
ClientAliveInterval 300
ClientAliveCountMax 2
MaxAuthTries 3
MaxSessions 2
MaxStartups 2
LoginGraceTime 60

# Logging
SyslogFacility AUTH
LogLevel VERBOSE

# Allowed users/groups
AllowGroups ssh-users

# Ciphers and algorithms
Ciphers chacha20-poly1305@openssh.com,aes256-gcm@openssh.com,aes128-
gcm@openssh.com,aes256-ctr,aes192-ctr,aes128-ctr
MACs hmac-sha2-256-etm@openssh.com,hmac-sha2-512-etm@openssh.com,hmac-sha2-
256,hmac-sha2-512
KexAlgorithms curve25519-sha256@libssh.org,ecdh-sha2-nistp521,ecdh-sha2-
nistp384,ecdh-sha2-nistp256,diffie-hellman-group16-sha512,diffie-hellman-group18-
sha512,diffie-hellman-group14-sha256
EOF

    # Create SSH users group
    groupadd -f ssh-users

    # Test SSH configuration
    if sshd -t; then
        systemctl restart ssh
        log "SSH hardening completed successfully"
    else
        log "ERROR: SSH configuration invalid, restoring backup"
        cp "$BACKUP_DIR/sshd_config.backup" /etc/ssh/sshd_config
        systemctl restart ssh
    fi
}

# 3. Firewall Configuration
harden_firewall() {
    log "=== FIREWALL HARDENING ==="

    # Reset UFW to defaults
    ufw --force reset
    # Set default policies
    ufw default deny incoming
    ufw default allow outgoing
    ufw default deny forward

    # Allow SSH (custom port)
    ufw allow 2222/tcp comment 'SSH'

    # Allow HTTP/HTTPS
    ufw allow 80/tcp comment 'HTTP'
    ufw allow 443/tcp comment 'HTTPS'

    # Rate limiting for SSH
    ufw limit 2222/tcp

    # Enable logging
    ufw logging on

    # Enable firewall
    ufw --force enable

    log "Firewall hardening completed"
}

# 4. Kernel Hardening
harden_kernel() {
    log "=== KERNEL HARDENING ==="

    # Backup original sysctl configuration
    cp /etc/sysctl.conf "$BACKUP_DIR/sysctl.conf.backup"

    # Create hardened sysctl configuration
    cat >> /etc/sysctl.conf << EOF

# Security hardening parameters
# Network security
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv6.conf.all.accept_redirects = 0
net.ipv6.conf.default.accept_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.default.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
net.ipv6.conf.default.accept_source_route = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.log_martians = 1
net.ipv4.icmp_echo_ignore_broadcasts = 1
net.ipv4.icmp_ignore_bogus_error_responses = 1
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1

# Memory protection
kernel.dmesg_restrict = 1
kernel.kptr_restrict = 2
kernel.yama.ptrace_scope = 1
kernel.kexec_load_disabled = 1

# File system protection
fs.protected_hardlinks = 1
fs.protected_symlinks = 1
fs.suid_dumpable = 0

# Process restrictions
kernel.core_uses_pid = 1
kernel.ctrl-alt-del = 0
EOF

    # Apply sysctl settings
    sysctl -p

    log "Kernel hardening completed"
}

# 5. File System Hardening
harden_filesystem() {
    log "=== FILESYSTEM HARDENING ==="

    # Set proper permissions on critical files
    chmod 600 /etc/shadow
    chmod 600 /etc/gshadow
    chmod 644 /etc/passwd
    chmod 644 /etc/group

    # Secure /tmp with proper mount options
    if ! grep -q "/tmp" /etc/fstab; then
        echo "tmpfs /tmp tmpfs defaults,nodev,nosuid,noexec,size=1G 0 0" >>
/etc/fstab
    fi

    # Remove unnecessary SUID/SGID binaries
    local suid_binaries=(
        "/usr/bin/at"
        "/usr/bin/wall"
        "/usr/bin/write"
        "/usr/bin/chfn"
        "/usr/bin/chsh"
        "/usr/bin/newgrp"
    )

    for binary in "${suid_binaries[@]}"; do
        if [[ -f "$binary" ]]; then
```

```
                chmod u-s "$binary"
                log "Removed SUID bit from $binary"
            fi
        done

        # Set umask for better default permissions
        echo "umask 027" >> /etc/profile
        echo "umask 027" >> /etc/bash.bashrc

        log "Filesystem hardening completed"
    }

    # 6. Service Hardening
    harden_services() {
        log "=== SERVICE HARDENING ==="

        # Disable unnecessary services
        local services_to_disable=(
            "avahi-daemon"
            "cups"
            "bluetooth"
            "whoopsie"
            "apport"
        )

        for service in "${services_to_disable[@]}"; do
            if systemctl is-enabled "$service" &>/dev/null; then
                systemctl disable "$service"
                systemctl stop "$service"
                log "Disabled service: $service"
            fi
        done

        # Configure fail2ban
        apt install -y fail2ban

        cat > /etc/fail2ban/jail.local << EOF
    [DEFAULT]
    bantime = 3600
    findtime = 600
    maxretry = 3
    backend = systemd

    [sshd]
    enabled = true
    port = 2222
    logpath = /var/log/auth.log
    maxretry = 3
    bantime = 3600

    [nginx-http-auth]
    enabled = true
    port = http,https
    logpath = /var/log/nginx/error.log
```

```
maxretry = 3

[nginx-noscript]
enabled = true
port = http,https
logpath = /var/log/nginx/access.log
maxretry = 6
EOF

    systemctl enable fail2ban
    systemctl start fail2ban

    log "Service hardening completed"
}

# 7. Audit Configuration
harden_audit() {
    log "=== AUDIT HARDENING ==="

    # Install auditd
    apt install -y auditd audispd-plugins

    # Configure audit rules
    cat > /etc/audit/rules.d/hardening.rules << EOF
# Delete all existing rules
-D

# Buffer size
-b 8192

# Failure mode (0=silent, 1=printk, 2=panic)
-f 1

# Monitor authentication events
-w /etc/passwd -p wa -k identity
-w /etc/group -p wa -k identity
-w /etc/shadow -p wa -k identity
-w /etc/gshadow -p wa -k identity

# Monitor system configuration changes
-w /etc/ssh/sshd_config -p wa -k ssh_config
-w /etc/sudoers -p wa -k sudo_config
-w /etc/hosts -p wa -k network_config

# Monitor privilege escalation
-a always,exit -F arch=b64 -S execve -F euid=0 -F auid>=1000 -F auid!=4294967295 -
k privilege_escalation
-a always,exit -F arch=b32 -S execve -F euid=0 -F auid>=1000 -F auid!=4294967295 -
k privilege_escalation

# Monitor file access
-a always,exit -F arch=b64 -S open,openat,creat -F exit=-EACCES -k file_access
-a always,exit -F arch=b64 -S open,openat,creat -F exit=-EPERM -k file_access
```

```
# Lock the configuration
-e 2
EOF

    # Restart auditd
    systemctl enable auditd
    systemctl restart auditd

    log "Audit hardening completed"
}

# 8. Generate security report
generate_security_report() {
    log "=== GENERATING SECURITY REPORT ==="

    local report_file="/var/log/security-report-$(date +%Y%m%d_%H%M%S).txt"

    cat > "$report_file" << EOF
Security Hardening Report
========================
Date: $(date)
Hostname: $(hostname)
OS: $(lsb_release -d | cut -f2)
Kernel: $(uname -r)

Hardening Steps Completed:
- System updates and automatic security updates
- SSH hardening (port 2222, key-only auth)
- Firewall configuration (UFW)
- Kernel parameter hardening
- File system permissions hardening
- Service hardening and fail2ban
- Audit system configuration

Security Status:
- SSH Port: 2222
- Root Login: Disabled
- Password Auth: Disabled
- Firewall: Enabled
- Fail2ban: Active
- Audit: Enabled

Next Steps:
- Regular security updates
- Monitor audit logs
- Review fail2ban logs
- Periodic security assessments

Backup Location: $BACKUP_DIR
EOF

    log "Security report generated: $report_file"

    # Send report via email if configured
    if command -v mail &> /dev/null; then
        mail -s "Security Hardening Report - $(hostname)" admin@example.com <
"$report_file"
    fi
}

# Main execution
if [[ $EUID -ne 0 ]]; then
    echo "This script must be run as root"
    exit 1
fi

echo "WARNING: This script will make significant security changes to your system."
echo "Ensure you have console access and backups before proceeding."
read -p "Continue? (y/N): " -n 1 -r
echo

if [[ $REPLY =~ ^[Yy]$ ]]; then
    harden_system_updates
    harden_ssh
    harden_firewall
    harden_kernel
    harden_filesystem
    harden_services
    harden_audit
    generate_security_report

    log "Security hardening completed successfully"
    echo "IMPORTANT: SSH port changed to 2222. Update your connection settings."
    echo "Reboot recommended to apply all changes."
else
    echo "Security hardening cancelled."
fi
```

📈 Performance Optimization Strategies

System Performance Tuning

```
#!/bin/bash
# performance-optimization.sh - Comprehensive performance tuning

PERF_LOG="/var/log/performance-tuning.log"
BENCHMARK_DIR="/tmp/benchmarks-$(date +%Y%m%d_%H%M%S)"

log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $*" | tee -a "$PERF_LOG"
}

mkdir -p "$BENCHMARK_DIR"

log "Starting comprehensive performance optimization"
```

```
# Baseline performance measurement
baseline_performance() {
    log "=== BASELINE PERFORMANCE MEASUREMENT ==="

    # CPU performance
    log "CPU Performance Test:"
    sysbench cpu --cpu-max-prime=20000 --threads=$(nproc) run >
"$BENCHMARK_DIR/cpu_baseline.txt" 2>&1

    # Memory performance
    log "Memory Performance Test:"
    sysbench memory --memory-total-size=10G run >
"$BENCHMARK_DIR/memory_baseline.txt" 2>&1

    # Disk I/O performance
    log "Disk I/O Performance Test:"
    sysbench fileio --file-total-size=5G prepare > /dev/null 2>&1
    sysbench fileio --file-total-size=5G --file-test-mode=rndrw run >
"$BENCHMARK_DIR/disk_baseline.txt" 2>&1
    sysbench fileio --file-total-size=5G cleanup > /dev/null 2>&1

    # Network performance (if iperf3 is available)
    if command -v iperf3 &> /dev/null; then
        log "Network Performance Test (loopback):"
        iperf3 -s -D -p 5201
        sleep 2
        iperf3 -c 127.0.0.1 -p 5201 -t 10 > "$BENCHMARK_DIR/network_baseline.txt"
2>&1
        pkill iperf3
    fi

    log "Baseline measurements completed"
}

# CPU optimization
optimize_cpu() {
    log "=== CPU OPTIMIZATION ==="

    # Set CPU governor to performance
    if [[ -f /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor ]]; then
        echo performance | tee
/sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
        log "CPU governor set to performance"
    fi

    # Disable CPU mitigations for performance (security trade-off)
    if ! grep -q "mitigations=off" /proc/cmdline; then
        log "WARNING: Consider adding 'mitigations=off' to GRUB for better
performance (reduces security)"
    fi

    # Configure CPU affinity for critical services
    local critical_services=("nginx" "mysql" "postgresql")
    local cpu_count=$(nproc)
    local reserved_cpus=$((cpu_count - 1))

    for service in "${critical_services[@]}"; do
        if systemctl is-active --quiet "$service"; then
            local pid=$(systemctl show --property MainPID --value "$service")
            if [[ "$pid" != "0" ]]; then
                taskset -cp "0-$reserved_cpus" "$pid" 2>/dev/null
                log "Set CPU affinity for $service (PID: $pid)"
            fi
        fi
    done

    log "CPU optimization completed"
}

# Memory optimization
optimize_memory() {
    log "=== MEMORY OPTIMIZATION ==="

    # Configure swap settings
    sysctl vm.swappiness=10
    sysctl vm.vfs_cache_pressure=50
    sysctl vm.dirty_ratio=15
    sysctl vm.dirty_background_ratio=5

    # Configure huge pages for databases
    local total_mem_kb=$(grep MemTotal /proc/meminfo | awk '{print $2}')
    local hugepage_size_kb=$(grep Hugepagesize /proc/meminfo | awk '{print $2}')
    local hugepages_count=$((total_mem_kb / hugepage_size_kb / 4))  # Use 25% for
huge pages

    echo "$hugepages_count" >
/sys/kernel/mm/hugepages/hugepages-${hugepage_size_kb}kB/nr_hugepages
    log "Configured $hugepages_count huge pages"

    # Memory compaction
    echo 1 > /proc/sys/vm/compact_memory

    # Configure NUMA policy for multi-socket systems
    if [[ $(numactl --hardware | grep "available:" | awk '{print $2}') -gt 1 ]];
then
        log "Multi-NUMA system detected, configuring NUMA policies"
        # Set NUMA balancing
        echo 1 > /proc/sys/kernel/numa_balancing
    fi

    log "Memory optimization completed"
}

# Disk I/O optimization
optimize_disk_io() {
    log "=== DISK I/O OPTIMIZATION ==="
    # Optimize I/O scheduler for different disk types
    for disk in /sys/block/*/queue/scheduler; do
        local device=$(echo "$disk" | cut -d'/' -f4)

        # Check if it's an SSD
        if [[ $(cat "/sys/block/$device/queue/rotational") == "0" ]]; then
            echo "none" > "$disk" 2>/dev/null || echo "mq-deadline" > "$disk"
            log "Set scheduler for SSD $device: $(cat "$disk")"
        else
            echo "mq-deadline" > "$disk"
            log "Set scheduler for HDD $device: mq-deadline"
        fi
    done

    # Optimize read-ahead for better sequential performance
    for disk in /sys/block/*/queue/read_ahead_kb; do
        local device=$(echo "$disk" | cut -d'/' -f4)
        echo 4096 > "$disk"
        log "Set read-ahead for $device: 4096KB"
    done

    # Configure I/O limits with systemd
    mkdir -p /etc/systemd/system/user.slice.d
    cat > /etc/systemd/system/user.slice.d/io-limits.conf << EOF
[Slice]
IOWeight=100
IODeviceWeight=/dev/sda 200
IOReadBandwidthMax=/dev/sda 100M
IOWriteBandwidthMax=/dev/sda 50M
EOF

    systemctl daemon-reload

    log "Disk I/O optimization completed"
}

# Network optimization
optimize_network() {
    log "=== NETWORK OPTIMIZATION ==="

    # TCP buffer tuning
    sysctl net.core.rmem_default=262144
    sysctl net.core.rmem_max=16777216
    sysctl net.core.wmem_default=262144
    sysctl net.core.wmem_max=16777216
    sysctl net.ipv4.tcp_rmem="4096 65536 16777216"
    sysctl net.ipv4.tcp_wmem="4096 65536 16777216"

    # TCP congestion control
    sysctl net.ipv4.tcp_congestion_control=bbr

    # Network device optimization
    for interface in $(ip link show | grep -E '^[0-9]+:' | cut -d: -f2 | tr -d ' '
| grep -v lo); do
```

```
        # Increase ring buffer sizes
        ethtool -G "$interface" rx 4096 tx 4096 2>/dev/null || true

        # Enable offloading features
        ethtool -K "$interface" gro on 2>/dev/null || true
        ethtool -K "$interface" tso on 2>/dev/null || true
        ethtool -K "$interface" gso on 2>/dev/null || true

        log "Optimized network interface: $interface"
    done

    # Increase connection tracking table size
    sysctl net.netfilter.nf_conntrack_max=1048576

    log "Network optimization completed"
}

# Application-specific optimization
optimize_applications() {
    log "=== APPLICATION OPTIMIZATION ==="

    # Nginx optimization
    if systemctl is-active --quiet nginx; then
        log "Optimizing Nginx configuration"

        # Backup original config
        cp /etc/nginx/nginx.conf "/etc/nginx/nginx.conf.backup.$(date
+%Y%m%d_%H%M%S)"

        # Create optimized nginx config
        cat > /etc/nginx/conf.d/performance.conf << EOF
# Performance optimizations
worker_processes auto;
worker_rlimit_nofile 65535;

events {
    worker_connections 4096;
    use epoll;
    multi_accept on;
}

http {
    # Basic optimizations
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 30;
    keepalive_requests 1000;

    # Buffer optimizations
    client_body_buffer_size 128k;
    client_max_body_size 10m;
    client_header_buffer_size 1k;
    large_client_header_buffers 4 4k;
    output_buffers 1 32k;
    postpone_output 1460;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_proxied any;
    gzip_comp_level 6;
    gzip_types text/plain text/css text/xml text/javascript application/javascript
application/xml+rss application/json;

    # File caching
    open_file_cache max=200000 inactive=20s;
    open_file_cache_valid 30s;
    open_file_cache_min_uses 2;
    open_file_cache_errors on;
}
EOF

        nginx -t && systemctl reload nginx
        log "Nginx optimization completed"
    fi

    # MySQL optimization
    if systemctl is-active --quiet mysql; then
        log "Optimizing MySQL configuration"

        local total_mem_mb=$(($(grep MemTotal /proc/meminfo | awk '{print $2}') /
1024))
        local innodb_buffer_pool_size=$((total_mem_mb * 70 / 100))  # 70% of total
memory

        cat > /etc/mysql/conf.d/performance.cnf << EOF
[mysqld]
# Performance optimizations
innodb_buffer_pool_size = ${innodb_buffer_pool_size}M
innodb_log_file_size = 256M
innodb_log_buffer_size = 16M
innodb_flush_log_at_trx_commit = 2
innodb_flush_method = O_DIRECT
innodb_file_per_table = 1
innodb_read_io_threads = 8
innodb_write_io_threads = 8

# Query cache (if using MySQL < 8.0)
query_cache_type = 1
query_cache_size = 128M
query_cache_limit = 2M

# Connection settings
max_connections = 200
max_connect_errors = 10000
thread_cache_size = 50
```

```
table_open_cache = 4000

# Buffer settings
join_buffer_size = 256K
sort_buffer_size = 2M
read_buffer_size = 128K
read_rnd_buffer_size = 256K
EOF

        systemctl restart mysql
        log "MySQL optimization completed"
    fi

    log "Application optimization completed"
}

# Post-optimization benchmarks
post_optimization_benchmarks() {
    log "=== POST-OPTIMIZATION BENCHMARKS ==="

    # Wait for system to stabilize
    sleep 30

    # CPU performance
    log "CPU Performance Test (Post-optimization):"
    sysbench cpu --cpu-max-prime=20000 --threads=$(nproc) run >
"$BENCHMARK_DIR/cpu_optimized.txt" 2>&1

    # Memory performance
    log "Memory Performance Test (Post-optimization):"
    sysbench memory --memory-total-size=10G run >
"$BENCHMARK_DIR/memory_optimized.txt" 2>&1

    # Disk I/O performance
    log "Disk I/O Performance Test (Post-optimization):"
    sysbench fileio --file-total-size=5G prepare > /dev/null 2>&1
    sysbench fileio --file-total-size=5G --file-test-mode=rndrw run >
"$BENCHMARK_DIR/disk_optimized.txt" 2>&1
    sysbench fileio --file-total-size=5G cleanup > /dev/null 2>&1

    log "Post-optimization benchmarks completed"
}

# Generate performance report
generate_performance_report() {
    log "=== GENERATING PERFORMANCE REPORT ==="

    local report_file="/var/log/performance-report-$(date +%Y%m%d_%H%M%S).txt"

    cat > "$report_file" << EOF
Performance Optimization Report
==============================
Date: $(date)
Hostname: $(hostname)
```

```
CPU Cores: $(nproc)
Total Memory: $(free -h | grep Mem | awk '{print $2}')
Kernel: $(uname -r)

Optimizations Applied:
- CPU governor set to performance
- Memory settings optimized (swappiness, cache pressure)
- Huge pages configured
- Disk I/O schedulers optimized
- Network buffers and TCP settings tuned
- Application-specific optimizations (Nginx, MySQL)

Benchmark Results:
See detailed results in: $BENCHMARK_DIR

Recommendations:
- Monitor system performance regularly
- Adjust settings based on workload patterns
- Consider hardware upgrades for bottlenecks
- Implement application-level caching

Next Steps:
- Regular performance monitoring
- Workload-specific tuning
- Capacity planning
EOF

    log "Performance report generated: $report_file"

    # Compare baseline vs optimized results
    if [[ -f "$BENCHMARK_DIR/cpu_baseline.txt" && -f
"$BENCHMARK_DIR/cpu_optimized.txt" ]]; then
        local baseline_cpu=$(grep "events per second:"
"$BENCHMARK_DIR/cpu_baseline.txt" | awk '{print $4}')
        local optimized_cpu=$(grep "events per second:"
"$BENCHMARK_DIR/cpu_optimized.txt" | awk '{print $4}')

        if [[ -n "$baseline_cpu" && -n "$optimized_cpu" ]]; then
            local improvement=$(echo "scale=2; ($optimized_cpu - $baseline_cpu) /
$baseline_cpu * 100" | bc)
            echo "CPU Performance Improvement: ${improvement}%" >> "$report_file"
            log "CPU Performance Improvement: ${improvement}%"
        fi
    fi
}

# Main execution
if [[ $EUID -ne 0 ]]; then
    echo "This script must be run as root"
    exit 1
fi

# Install required tools
apt update
apt install -y sysbench bc

baseline_performance
optimize_cpu
optimize_memory
optimize_disk_io
optimize_network
optimize_applications
post_optimization_benchmarks
generate_performance_report

log "Performance optimization completed successfully"
echo "Performance report and benchmarks saved to: $BENCHMARK_DIR"
echo "Reboot recommended to ensure all optimizations are active."
```

🧠 Knowledge Check

Quick Quiz

1. **What are the key principles of Infrastructure as Code (IaC)?**

   ▶ Answer
   - Version control for infrastructure
   - Reproducible and consistent deployments
   - Automated provisioning and configuration
   - Documentation through code
   - Testing and validation of infrastructure changes

2. **What security measures should be implemented for SSH hardening?**

   ▶ Answer
   - Disable root login and password authentication
   - Use non-standard port and key-based authentication
   - Implement connection limits and timeouts
   - Use strong ciphers and algorithms
   - Enable logging and monitoring

3. **How do you balance security and performance in system optimization?**

   ▶ Answer
   - Risk assessment and threat modeling
   - Layered security approach
   - Performance testing with security measures
   - Regular security audits and performance monitoring
   - Documentation of trade-offs and decisions

4. **What are the essential components of a disaster recovery plan?**

   ▶ Answer
   - Regular backups with tested restore procedures

- Documentation of recovery processes
- Alternative infrastructure and failover procedures
- Communication plans and contact information
- Regular testing and updates of the plan

Final Challenges

Challenge 1: Complete Infrastructure Automation

```
# Build a complete infrastructure automation solution:
# - Infrastructure as Code with Ansible/Terraform
# - Automated security hardening and compliance
# - Performance optimization and monitoring
# - Disaster recovery and backup automation
# - Documentation and knowledge management
# - Testing and validation frameworks
```

Challenge 2: Enterprise-Grade Monitoring

```
# Implement enterprise monitoring and alerting:
# - Multi-tier monitoring (infrastructure, application, business)
# - Predictive analytics and anomaly detection
# - Automated incident response and remediation
# - Comprehensive dashboards and reporting
# - Integration with external systems and APIs
```

Challenge 3: Security Operations Center (SOC)

```
# Create a mini Security Operations Center:
# - Centralized log collection and analysis
# - Threat detection and incident response
# - Vulnerability management and patching
# - Compliance monitoring and reporting
# - Security awareness and training programs
```

🎓 Career Development and Certifications

Recommended Learning Path

```
# Linux Certifications
# 1. CompTIA Linux+ (Entry level)
# 2. LPIC-1, LPIC-2, LPIC-3 (Linux Professional Institute)
# 3. Red Hat Certified System Administrator (RHCSA)
# 4. Red Hat Certified Engineer (RHCE)
# Networking Certifications
# 1. CompTIA Network+ (Foundation)
# 2. Cisco CCNA (Cisco networking)
# 3. CompTIA Security+ (Network security)
# 4. Certified Ethical Hacker (CEH)

# Cloud and DevOps
# 1. AWS Certified Solutions Architect
# 2. Certified Kubernetes Administrator (CKA)
# 3. Docker Certified Associate
# 4. Ansible Certified Specialist

# Security Specializations
# 1. CISSP (Information security management)
# 2. CISM (Information security management)
# 3. GSEC (SANS security essentials)
# 4. OSCP (Offensive security)
```

Professional Development Plan

```
# 12-Month Professional Development Plan

## Months 1-3: Foundation Strengthening

- [ ] Complete advanced Linux administration courses
- [ ] Practice with complex networking scenarios
- [ ] Build home lab environment
- [ ] Start contributing to open-source projects
- [ ] Begin studying for first certification

## Months 4-6: Specialization

- [ ] Choose specialization area (security, cloud, networking)
- [ ] Take relevant certification exam
- [ ] Build portfolio projects
- [ ] Attend industry conferences/meetups
- [ ] Start mentoring junior colleagues

## Months 7-9: Advanced Skills

- [ ] Learn automation and orchestration tools
- [ ] Study cloud platforms and services
- [ ] Practice incident response scenarios
- [ ] Develop leadership and communication skills
- [ ] Pursue advanced certifications

## Months 10-12: Leadership and Innovation

- [ ] Lead technical projects
- [ ] Speak at conferences or write technical articles
- [ ] Develop training materials for others
```

```
  - [ ] Explore emerging technologies
  - [ ] Plan next year's development goals
```

🚀 Congratulations!

You've completed the comprehensive Linux & Networking Mastery Series! You now have:

✅ **Solid Foundation**: Linux fundamentals, file operations, permissions, and process management

✅ **System Administration Skills**: User management, package management, system monitoring, and automation

✅ **Networking Expertise**: TCP/IP, DNS, routing, firewalls, and network troubleshooting

✅ **Security Knowledge**: SSH, SSL/TLS, firewalls, intrusion detection, and security hardening

✅ **Advanced Tools**: Network analysis, performance monitoring, automation, and debugging

✅ **Professional Practices**: Best practices, documentation, incident response, and career development

Your Next Steps:

1. **Practice Regularly**: Set up home labs and practice scenarios
2. **Stay Current**: Follow industry news, blogs, and security advisories
3. **Get Certified**: Pursue relevant certifications for your career goals
4. **Contribute**: Join open-source projects and share your knowledge
5. **Network**: Connect with professionals in the field
6. **Keep Learning**: Technology evolves rapidly - never stop learning!

Resources for Continued Learning:

- **Documentation**: Always refer to official documentation
- **Communities**: Reddit r/linux, r/networking, Stack Overflow
- **Blogs**: Red Hat Blog, Ubuntu Blog, Cisco Blog
- **Podcasts**: Linux Action News, Network Break, Security Now
- **Books**: "UNIX and Linux System Administration Handbook", "TCP/IP Illustrated"
- **Labs**: VirtualBox/VMware labs, Cloud provider free tiers

---

> **Remember**: The journey to mastery is continuous. What you've learned here is a strong foundation, but the real learning happens when you apply these skills to solve real-world problems. Stay curious, keep practicing, and never hesitate to dive deeper into topics that interest you!

Good luck on your Linux and networking journey! 🐧🌐