Linux Server Manual Patching

Standard Operating Procedure

Learning & Practice Project

Author: Ashwin Saji

Version: v1.0

Date: January 13, 2026

Overview

► 1. Purpose

This document describes a manual patching process for Linux servers as part of a learning and practice project. The focus is on building foundational skills in system administration, including basic operational hygiene, script execution, and system health validation.

> **Learning Objectives:**
>
> - Practice basic Linux administration tasks
> - Understand package management workflows
> - Develop pre-check and post-check validation habits
> - Document operational procedures

This SOP is designed for educational purposes and reflects realistic practices suitable for non-production environments and junior administrators.

► 2. Scope

This procedure applies to the following environment:

Category	Details
Operating System	Ubuntu-based Linux distributions
Environment	Non-production (development, test, learning labs)
Execution Method	Manual execution only (no automation schedulers)

Package Manager	apt (Advanced Package Tool)

> **⚠ Not Covered:**
>
> This SOP does not include enterprise change management, production deployments, automated rollback procedures, or compliance auditing requirements.

Tools & Scripts

► 3. Tools Used

- **Bash Shell:** Primary scripting environment
- **apt Package Manager:** For system updates and package management
- **systemd Journal Tools:** For log review (journalctl)
- **GitHub:** Version control for scripts and documentation
- **Standard Linux Utilities:** df, uptime, systemctl, etc.

► 4. Script Overview

The patching workflow is supported by four custom Bash scripts:

#!/ diskcleanbasic.sh

Purpose: Cleans package cache and checks available disk space

Key Actions:

- Removes obsolete cached packages (apt clean)
- Displays current disk usage
- Verifies sufficient space for patching operations

#!/ precheck.sh

Purpose: Performs basic system health checks before patching

Key Actions:

- Checks system uptime and load average
- Verifies critical services are running

- Reviews recent system logs for errors
- Confirms network connectivity

#!/ patching.sh

Purpose: Executes the package update process

Key Actions:

- Updates package lists (apt update)
- Performs package upgrades (apt upgrade)
- Logs all update activity
- Reports packages that were updated

#!/ postcheck.sh

Purpose: Validates system state after patching

Key Actions:

- Verifies critical services remain operational
- Checks for any failed systemd units
- Reviews post-patch system logs
- Confirms system responsiveness

Patching Procedure

► 5. Manual Patching Workflow

1

Disk Cleanup

Execute the disk cleanup script to free up space and verify capacity:

- Run: `sudo ./diskcleanbasic.sh`
- Review disk usage output
- Ensure at least 2GB free space in /var

Expected Outcome: Cache cleaned, sufficient disk space confirmed

2

Pre-Checks

Validate system health before making changes:

- Run: `sudo ./precheck.sh`
- Review all output for warnings or errors
- Verify all critical services show "active (running)"
- Check that system load is reasonable

Expected Outcome: All checks pass with no critical issues

> **⚠ If Issues Found:**
>
> Investigate and resolve any errors before proceeding to patching.

Manual Patching

Execute the patching script to update packages:

3

- Run: `sudo ./patching.sh`
- Monitor the execution for any errors or prompts
- Review the list of packages being updated
- Wait for completion (may take several minutes)

Expected Outcome: All packages successfully updated, log file created

Post-Checks

Confirm system health after patching:

4

- Run: `sudo ./postcheck.sh`
- Verify all services remain operational
- Check for any new errors in system logs
- Confirm system is responsive

Expected Outcome: System healthy, no degraded services

> **Note:**
>
> If kernel was updated, a reboot may be required. Plan accordingly.

Documentation & Constraints

► 6. Logs & Evidence

All script executions generate logs for review and troubleshooting:

> **Log Directory Structure:**
>
> Logs are stored in: `/var/log/patching/` or `~/logs/`
>
> - **precheck_[date].log**
>
> - Pre-patching system state
>
> - **patching_[date].log**
>
> - Update execution details
>
> - **postcheck_[date].log**
>
> - Post-patching validation

Evidence Collection:

- Retain logs from each patching session
- Take screenshots of key execution steps
- Document any issues encountered and resolutions
- Commit logs and notes to GitHub repository

Log retention is based on available disk space, typically 30-90 days for learning environments.

► 7. Limitations

This procedure has the following known limitations:

Technical Constraints:

- **No Automation:**

  Scripts require manual execution; no scheduling tools (cron, systemd timers) are implemented

- **No Automated Rollback:**

  Package rollback requires manual intervention using apt history

- **Limited Error Handling:**

  Scripts provide basic validation but are not production-hardened

- **Single System:**

  No multi-server orchestration or parallel execution

Scope Limitations:

- **Not for Production:**

  This workflow is designed for learning environments only

- **No Change Management:**

  No approval workflows, ticketing integration, or compliance tracking

- **No SLA Commitments:**

  Timing and scheduling are flexible for educational purposes

- **Basic Monitoring:**

  Relies on script output; no integration with monitoring platforms

As skills develop, these limitations can be addressed through additional tooling, automation frameworks, and best practices for production environments.

Version Control

► 8. Version History

Version	Date	Author	Changes
v1.0	January 13, 2026	Ashwin Saji	Initial documented process for learning project

Future Enhancements (Planned):

- Add pre-patching backup verification step
- Expand error handling in scripts
- Include kernel update reboot procedures
- Add service-specific health checks
- Document rollback procedures

► 9. Document Control

Item	Details
Document Owner	Ashwin Saji
Review Cycle	As needed based on script updates
Repository	GitHub (version controlled with scripts)
Distribution	Public (learning project)