Linux Server Manual Patching

Production Standard Operating Procedure

📝 **Version:** 1.0

👤 **Author:** Ashwin Saji

📅 **Date:** January 2026

🎯 **Audience:** L1/L2 Linux Admins, SRE

1. Introduction

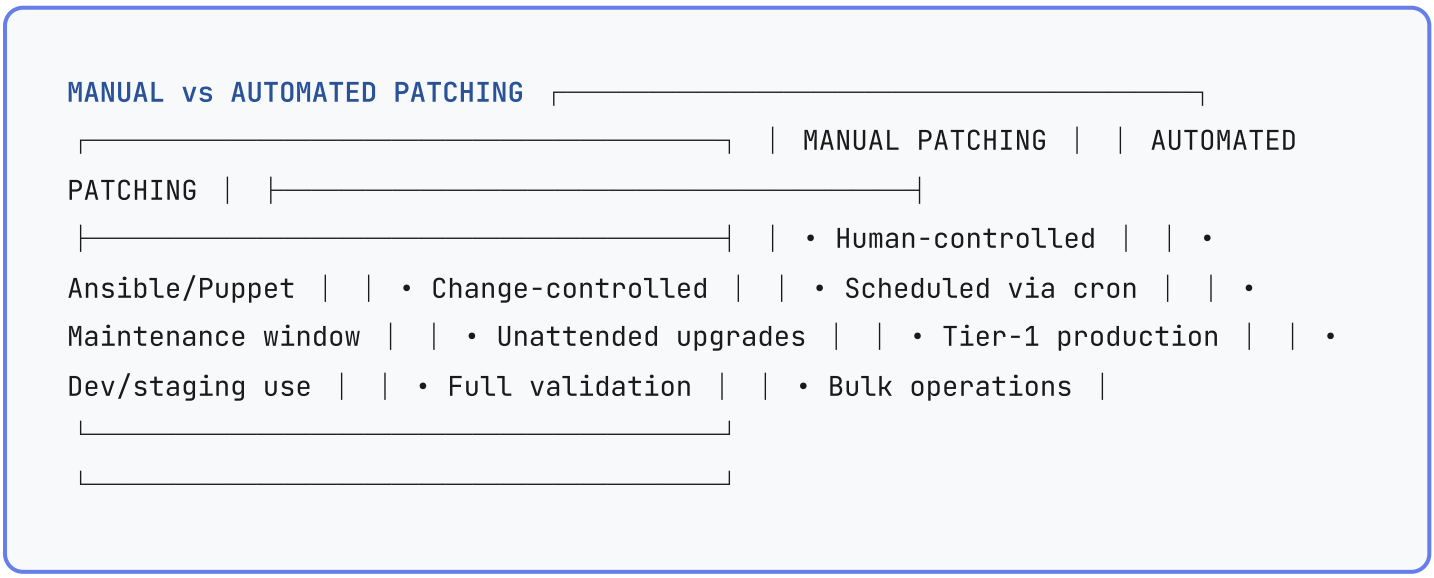
Linux server manual patching is the controlled process of applying operating system updates, security patches, and package upgrades to production Linux servers through human-driven procedures rather than automated orchestration.

✅ INCLUDED

- ✓ OS-level package updates
- ✓ Security patches (CVE)
- ✓ Kernel updates (approved)
- ✓ System library upgrades
- ✓ Dependency resolution

❌ EXCLUDED

- ✗ Application patches
- ✗ Firmware updates
- ✗ Database patching
- ✗ Middleware updates
- ✗ Container images



✅ 2. Pre-Patching Checks (MANDATORY)

> ⚠️ **CRITICAL**
>
> All pre-patching checks MUST be completed before proceeding. Skipping these steps violates production standards.

- [ ] Change ticket approved (CHG number verified)
- [ ] Maintenance window confirmed with stakeholders
- [ ] Application team notified (48hr advance)
- [ ] Backup verified within last 24 hours
- [ ] Monitoring alerts acknowledged/suppressed
- [ ] Emergency rollback plan documented

System Information Gathering

```
# OS identification cat /etc/os-release lsb_release -a # Kernel version check uname -r dpkg -l | grep linux-image # System uptime uptime who -b
```

💾 Disk Space Threshold Validation

FILESYSTEM	MINIMUM FREE	COMMAND	RISK LEVEL
/	≥ 2–5 GB	`df -h /`	CRITICAL
/var	≥ 1–2 GB	`df -h /var`	CRITICAL
/boot	≥ 300 MB	`df -h /boot`	HIGH
/tmp	≥ 500 MB	`df -h /tmp`	MEDIUM

```
# Comprehensive disk space check df -h | grep -E '^Filesystem|/$|/var$|/boot$|/tmp$'
df -i # Inode check # Top disk consumers du -sh /* 2>/dev/null | sort -rh | head -10
```

🚨 Failed Services Check

```
# Identify failed services systemctl --failed systemctl list-units --state=failed
--all
```

🛑 STOP CONDITIONS

DO NOT PROCEED if ANY exist:

- Disk space below thresholds
- Critical services in failed state
- Active P1/P2 incidents
- Change freeze period
- No valid change approval
- Backup verification failed

🔍 3. Patch Impact Analysis

```
# Update package cache and list upgrades sudo apt update apt list --upgradable apt list --upgradable | wc -l # Count packages # Detailed package info apt-cache policy <package-name>
```

📊 Risk Classification Matrix

RISK LEVEL	PACKAGE TYPES	REBOOT	APPROVAL
LOW	Libraries, utilities, docs	❌ No	L2 Admin
MEDIUM	System libs (libc, openssl)	❌ No	L2/L3 Admin
HIGH	Kernel, drivers, init	✅ Yes	Senior SRE + CAB
CRITICAL	DB engines, clusters	✅ Yes	Ops Manager + App Team

🔴 Kernel Detection Logic

```
# Identify kernel packages in update list apt list --upgradable | grep -E 'linux-image|linux-headers' # Check current vs available kernel uname -r # Running apt-cache policy linux-image-generic # Available # Identify high-risk packages apt list --upgradable | grep -E 'systemd|grub|libc6|openssh'
```

4. Patch Execution (Manual)

Step 1: Update Package Cache

```
sudo apt update # Expected: "Reading package lists... Done" with no errors
```

Step 2: Execute Upgrades

```
# Dry-run simulation (recommended first) sudo apt upgrade --dry-run # Execute upgrade
sudo apt upgrade -y # Alternative: Full upgrade (handles dependency changes) sudo apt
full-upgrade -y
```

Step 3: Configuration File Prompts

ℹ️ **Config Prompt Decision Matrix**

FILE TYPE	ACTION	RATIONALE
/etc/ssh/sshd_config	**N** (Keep)	Custom hardening
/etc/systemd/*.conf	**N** (Keep)	Production tuning
/etc/logrotate.d/*	**D** then decide	Review first

Step 4: Hold Kernel (If Required)

```
# Hold kernel packages sudo apt-mark hold linux-image-generic linux-headers-
generic # Verify hold status apt-mark showhold # Proceed with non-kernel upgrades sudo
apt upgrade -y # Later: Remove hold sudo apt-mark unhold linux-image-generic
```

🚨 5. Failure Handling

🔴 Broken Dependencies

```
sudo apt --fix-broken install
sudo dpkg --configure -a sudo apt
install -f
```

🔒 DPKG Interrupted

```
# Remove stale locks sudo rm
/var/lib/dpkg/lock* sudo dpkg --
configure -a sudo apt --fix-
broken install
```

⚠️ Disk Space During Upgrade

```
# Emergency space recovery sudo apt clean sudo apt autoremove -y # Remove old
kernels (keep current + one previous) uname -r # Note current kernel dpkg -l | grep
linux-image sudo apt autoremove --purge -y
```

❌ NEVER DO DURING FAILURES

- Force-remove critical packages (libc6, systemd, init)
- Manually edit /var/lib/dpkg/status
- Reboot server mid-upgrade
- Kill dpkg/apt processes forcefully
- Use --force-* flags without understanding

🔄 6. Reboot Decision Matrix

When Reboot is REQUIRED ✅

```
# Check if reboot required [ -f /var/run/reboot-required ] && echo "Reboot Required" # View reason cat /var/run/reboot-required.pkgs
```

PACKAGE TYPE	REBOOT REQUIRED	REASON
linux-image-*	✅ YES	New kernel loaded only on boot
libc6 (glibc)	✅ YES	Core library - all processes affected
systemd	✅ YES	Init system requires restart
openssl/libssl	❌ NO	Service restart sufficient
Application packages	❌ NO	Application restart only

🧪 Kernel Mismatch Verification

```
# Running kernel uname -r # Output: 5.15.0-91-generic # Installed kernels dpkg -l | grep linux-image | grep ^ii # If versions differ → Reboot required
```

🔄 Controlled Reboot Procedure

```
# Announce reboot wall "REBOOT IN 5 MIN - Maintenance - CHG123456" # Stop application services sudo systemctl stop <application>.service # Scheduled reboot (recommended) sudo shutdown -r +2 "Patching maintenance reboot" # Alternative: Immediate sudo reboot
```

✅ 7. Post-Patch Validation

- [ ] Kernel version verified (if rebooted)
- [ ] All critical services running
- [ ] No failed systemd units
- [ ] System logs reviewed
- [ ] Application health check passed
- [ ] Monitoring metrics normal
- [ ] Disk space acceptable