ADMIN:DOCS

Ubuntu-Upgrade: Von Version 22.04 LTS auf 24.04 LTS

Schritt-für-Schritt-Anleitung für das Upgrade von Ubuntu 22.04 LTS (Jammy Jellyfish) auf 24.04 LTS (Noble Numbat) – inklusive System-Backup, Release-Upgrade und Validierung.

Kategorie: Serverumgebungen **Veröffentlicht:** 10.10.2024

Ein Distributions-Upgrade von **Ubuntu 22.04 LTS (Jammy Jellyfish)** auf **Ubuntu 24.04 LTS (Noble Numbat)** bringt moderne Kernel-Features, aktuellere Software-Stacks und fünf Jahre langfristige Unterstützung. Ein geordnetes In-Place-Upgrade erfordert sorgfältige Vorbereitung, saubere Paketquellen und ein erprobtes Backup.

Die hier vermittelten Upgrade- und Recovery-Techniken sind essenziell für den sicheren Betrieb professioneller Linux-Serverumgebungen und orientieren sich an Standards von Prüfungen wie der LPIC-1 oder CompTIA Linux+.

> □□ **Hinweis:** Vorausgesetzt werden fundierte Kenntnisse der Linux-Kommandozeile sowie administrative Rechte (`sudo`) zur Verwaltung von Paketquellen und Systemdiensten (`systemctl`).

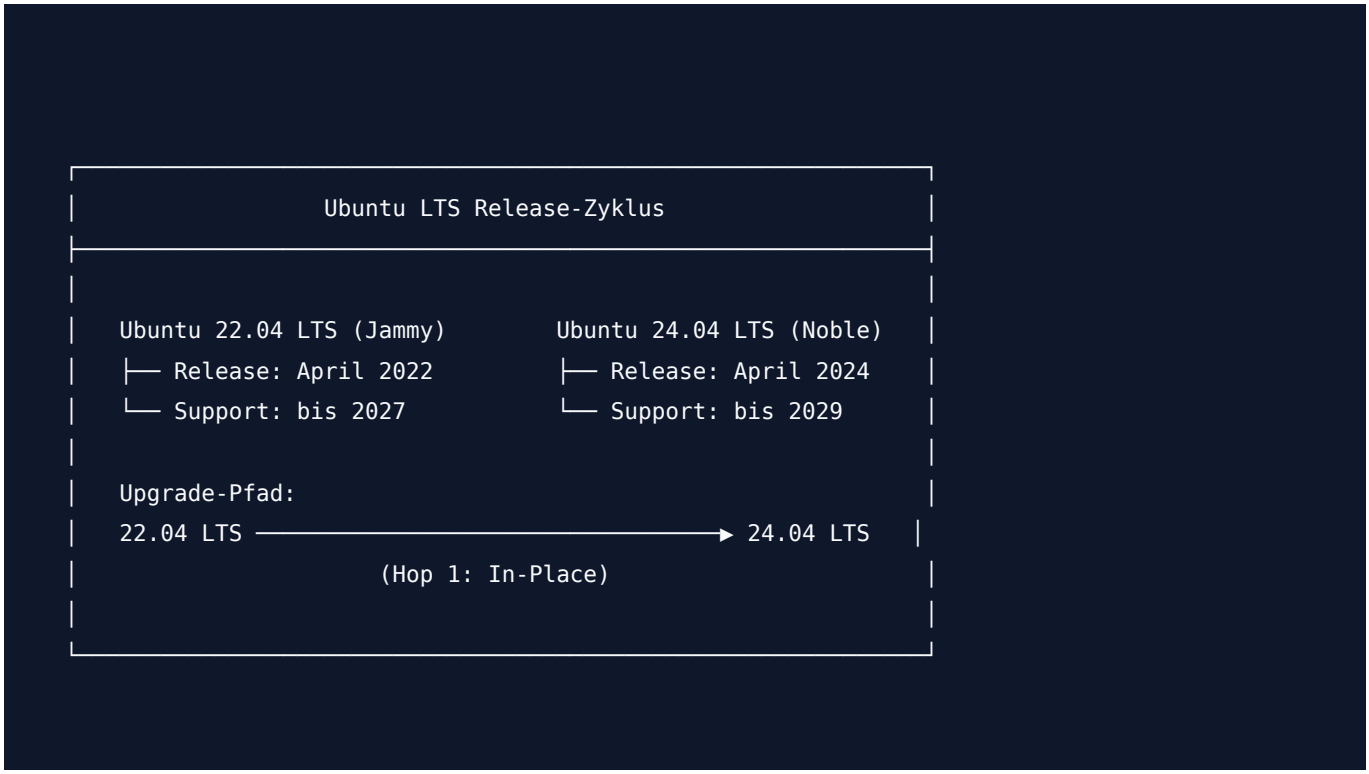
> ⚠ **Warnung:** Führe ein Major-Upgrade niemals direkt auf Produktivsystemen aus, ohne vorherige Tests durchgeführt zu haben. Nutze ein Testsystem oder eine virtuelle Maschine zur Vorbereitung.

Quelle: https://admindocs.de/de/serverumgebungen/ubuntu-upgrade-von-version-22-04-lts-auf-24-04-lts

Grundlagen und Versionsanalyse

LTS-Versionen verstehen

LTS steht für **Long Term Support** – Canonicals Zusage für fünf Jahre standardmäßige Sicherheitsupdates und Bugfixes. Für Server- und Produktionsumgebungen sollten ausschließlich LTS-Releases eingesetzt werden.



Die wichtigsten Neuerungen zwischen 22.04 und 24.04:

Komponente	Ubuntu 22.04 LTS (Jammy)	Ubuntu 24.04 LTS (Noble)	Bedeutung für den Serverbetrieb
Linux Kernel	5.15	6.8	Bessere Hardware-Unterstützung, modernes cgroup-Handling
Python	3.10	3.12	PEP 668 (`externally-managed-environment` für pip)
systemd	249	255	TPM2-Integration, moderne Service-Watchdogs

Komponente	Ubuntu 22.04 LTS (Jammy)	Ubuntu 24.04 LTS (Noble)	Bedeutung für den Serverbetrieb
OpenSSH	8.9p1	9.6p1	Modernisierte Kex-Algorithmen, verbesserte Sicherheit
AppArmor	3.0	4.0	Verbesserte Unprivileged User Namespace Policies
Netplan	0.104	1.0	Status-Befehle (`netplan status`), stabilisierte API

> □□ **Upgrade-Regel von Ubuntu:** Ubuntu erlaubt direkte Release-Upgrades immer nur zwischen aufeinanderfolgenden LTS-Versionen. Ein direkter Sprung von 20.04 auf 24.04 ist nicht möglich (der Pfad lautet: 20.04 → 22.04 → 24.04).

System-Analyse vor dem Upgrade

Vor jedem Eingriff in die Paketverwaltung musst du den genauen Ausgangszustand deines Systems erfassen:

1. Aktuelle Version bestätigen:

```
lsb_release -a
```

Die Ausgabe sollte `Release: 22.04` und `Codename: jammy` bestätigen.

2. Hardware-Ressourcen und Speicherplatz prüfen:

```
# CPU-Architektur und Kerne
lscpu | head -20

# RAM-Verfügbarkeit (mindestens 2 GB, empfohlen: 4 GB+)
free -h

# Festplattenplatz auf Root / (mindestens 20-25 GB frei empfohlen)
df -h /
```

> ⚠ **Warnung:** Stelle sicher, dass auf der Root-Partition (`/`) und in `/boot` ausreichend freier Speicherplatz vorhanden ist. Ein volllaufendes Dateisystem während des Paket-Downloads oder Kernel-Entpackens führt zu schwerwiegenden Paketinkonsistenzen.

Vollständige Datensicherung

Vor dem Upgrade ist ein verifiziertes Backup aller Konfigurationen und Benutzerdaten Pflicht.

```
# Prüfe wichtige Dienste und Datenpfade
sudo mysql --version 2>/dev/null && echo "MySQL/MariaDB gefunden"
sudo apache2 -v 2>/dev/null && echo "Apache gefunden"
sudo nginx -v 2>/dev/null && echo "Nginx gefunden"
```

Praxis-Backup-Script:

Das Sicherungsverzeichnis sollte idealerweise auf einem separaten Speichermedium (z. B. NFS-Share, dedizierter Backup-Storage oder USB-Laufwerk) liegen, um den für das Upgrade benötigten Speicherplatz auf der Root-Partition (`/`) nicht zu belasten.

```
#!/bin/bash
# backup_vor_upgrade.sh
set -euo pipefail

BACKUP_DIR="/backup/ubuntu22_upgrade_$(date +%Y%m%d_%H%M%S)"
echo "=== Erstelle Backup in: $BACKUP_DIR ==="

sudo mkdir -p "$BACKUP_DIR"

# Wichtige Systemdateien sichern
sudo cp -a /etc "$BACKUP_DIR/etc"
sudo cp -a /home "$BACKUP_DIR/home"
sudo cp -a /var/log/dpkg.log "$BACKUP_DIR/" 2>/dev/null || true

# Installierte Paketliste sichern
dpkg --get-selections > "$BACKUP_DIR/installed_packages.txt"

# Paketquellen-Status sichern
sudo cp -a /etc/apt/sources.list* "$BACKUP_DIR/"

echo "Backup erfolgreich abgeschlossen. Größe:"
sudo du -sh "$BACKUP_DIR"
```

Script ausführbar machen und ausführen:

```
chmod +x backup_vor_upgrade.sh
./backup_vor_upgrade.sh
```

Paket- und Systemvorbereitung

1. Bestehende Pakete vollständig aktualisieren

Bevor der Sprung auf die neue Version vollzogen werden kann, muss das bestehende 22.04-System auf dem absolut neuesten Stand sein.

```
# Paketlisten aktualisieren
sudo apt update

# Alle Updates inklusive Abhängigkeitsanpassungen installieren
sudo apt full-upgrade -y

# Nicht mehr benötigte alte Bibliotheken entfernen
sudo apt autoremove --purge -y

# Paket-Cache bereinigen
sudo apt autoclean
```

Was tun bei zurückgehaltenen Paketen (*packages kept back*)?

```
# Falls Pakete zurückgehalten werden, diese explizit installieren:
sudo apt install --only-upgrade linux-generic linux-headers-generic
```

2. Fremdquellen und PPAs prüfen

Drittanbieter-Paketquellen (PPAs) können während des Upgrades zu Abhängigkeitskonflikten führen. Der Upgrade-Manager deaktiviert Fremdquellen in der Regel automatisch, es ist jedoch sauberer, problematische Quellen vorab zu prüfen:

```
# Drittanbieter-PPAs auflisten
grep -r "ppa" /etc/apt/sources.list.d/ || true

# Snap-Pakete aktualisieren
snap list
sudo snap refresh
```

3. Upgrade-Manager konfigurieren

Das offizielle Tool `do-release-upgrade` steuert den Upgrade-Prozess.

```
# Update-Manager installieren
sudo apt install update-manager-core -y

# Release-Upgrade-Konfiguration prüfen
sudo nano /etc/update-manager/release-upgrades
```

Stelle sicher, dass die Zeile für den Prompt auf `lts` steht:

```
[DEFAULT]
Prompt=lts
```

Der Upgrade-Prozess (22.04 → 24.04)

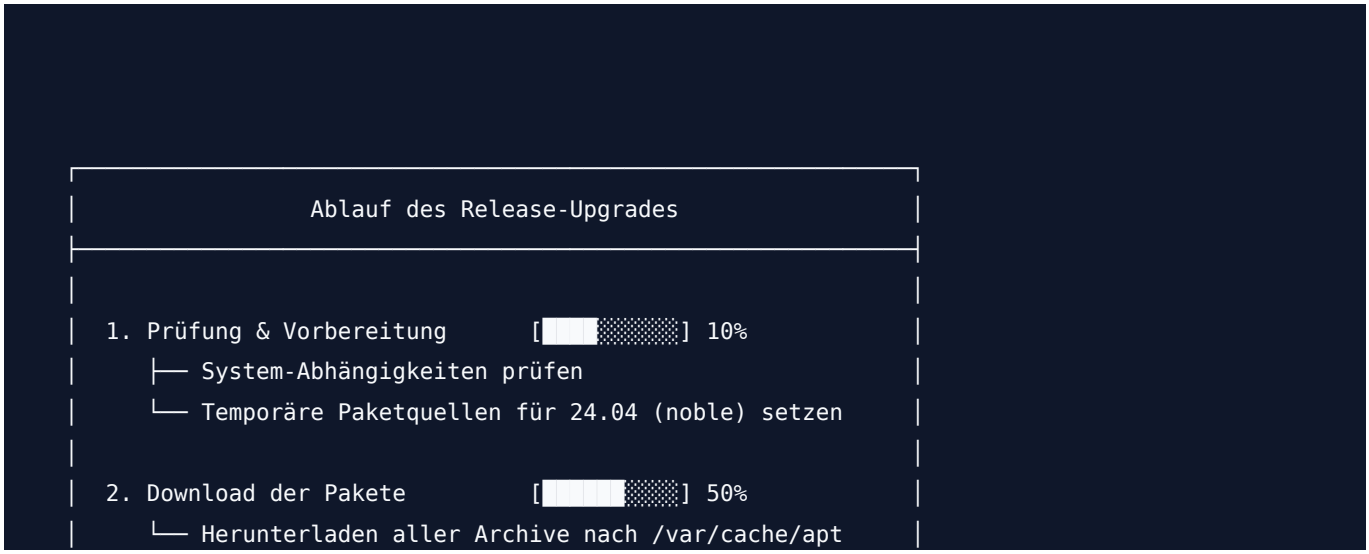
Upgrade in einer Terminal-Session starten

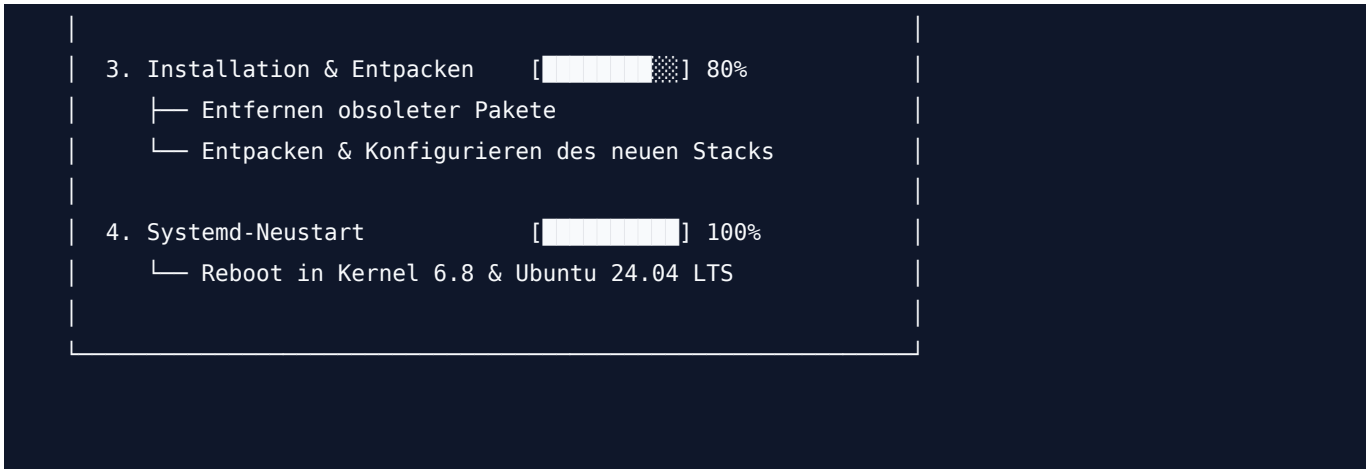
Wenn du das Upgrade über eine SSH-Verbindung ausführst, solltest du unbedingt `screen` oder `tmux` verwenden. Bricht die Netzwerkverbindung ab, läuft das Upgrade im Hintergrund ungestört weiter.

```
# Screen-Sitzung starten
screen -S noble-upgrade

# Release-Upgrade einleiten
sudo do-release-upgrade
```

> **SSH-Notfallzugang:** Bei Ausführung über SSH startet `do-release-upgrade` automatisch einen zusätzlichen SSH-Daemon auf Port `1022`. Falls Port 1022 in deiner Firewall geblockt ist, halte die Ports offen oder arbeite direkt an der Konsole.





Typische Dialoge während des Upgrades

1. **Abfrage zu geänderten Konfigurationsdateien:**

- Wenn eine Konfigurationsdatei (z. B. `/etc/ssh/sshd_config` oder `/etc/issue`) lokal modifiziert wurde, fragt der Installer, ob die Paketbetreuer-Version übernommen oder die lokale Datei behalten werden soll.
- **Empfehlung:** Mit `D` die Diffs prüfen. Eigene Anpassungen in Serverdiensten behältst du in der Regel mit `N` (Keep existing version).

1. **Obsolete Pakete entfernen:**

- Am Ende des Installationsprozesses fragt der Wizard: `Remove obsolete packages?`.
- **Empfehlung:** Bestätige mit `y`, um Altlasten sauber zu bereinigen.

Upgrade abschließen und System neu starten

Sobald die Meldung `System upgrade is complete. Restart required` erscheint, starte das System neu:

```
# Neustart durchführen
sudo reboot
```

Validierung nach dem Upgrade

Nach dem Neustart loggst du dich wieder ein und überprüfst die Systemgesundheit:

1. Versions- und Kernel-Check

```
# OS-Release prüfen
lsb_release -a

# Erwartete Ausgabe:
# Description: Ubuntu 24.04 LTS (Noble Numbat)
# Release:     24.04

# Kernel-Version prüfen (sollte 6.8.x sein)
uname -r
```

2. Service-Status und Logfiles kontrollieren

```
# Fehlgeschlagene Systemd-Units finden
systemctl --failed

# Kritische Kernel-Meldungen prüfen
sudo dmesg -l err,crit

# Netplan und Netzwerkverbindungen prüfen
sudo netplan status
ip addr show
```

3. Paketverwaltung bereinigen

```
# Letzten Stand der Paketdatenbank sicherstellen
sudo apt update
sudo apt autoremove --purge -y
sudo apt autoclean
```

Problembehandlung & Notfall-Strategien

Szenario 1: Abgebrochenes Upgrade / Halb-konfigurierte Pakete

Falls das Upgrade durch einen Verbindungsabbruch oder Stromausfall unterbrochen wurde:

```
# Paketverwaltung entsperren und konfigurieren
sudo dpkg --configure -a

# Defekte Abhängigkeiten auflösen
sudo apt install -f -y

# Paketstatus prüfen
sudo dpkg --audit
```

Szenario 2: SSH- oder Netzwerkprobleme nach dem Reboot

Wenn das Netzwerk nach dem Upgrade nicht automatisch startet:

```
# Netplan-Konfigurationsdateien prüfen
cat /etc/netplan/*.yaml

# Konfiguration neu kompilieren und anwenden
sudo netplan generate
sudo netplan apply
```

Befehlsreferenz (Cheatsheet)

Die wichtigsten operativen Befehle für die Vorbereitung, Durchführung und Verifikation des Upgrades auf einen Blick:

Phase / Befehl	Funktion / Erläuterung
`lsb_release -a && uname -r`	Prüft die aktive Ubuntu- und Kernel-Version vor oder nach dem Upgrade
`df -h / && free -h`	Überprüft freien Festplattenspeicher und verfügbaren Arbeitsspeicher
`sudo apt update && sudo apt full-upgrade -y`	Aktualisiert bestehende Pakete und löst Abhängigkeiten vorab vollständig auf
`sudo apt autoremove --purge -y && sudo apt autoclean`	Bereinigt verwaiste Altpakete und leert den lokalen Paketcache
`screen -S noble-upgrade`	Startet eine getrennte Terminalsitzung zum Schutz vor SSH-Verbindungsabbrüchen
`sudo do-release-upgrade`	Startet den offiziellen, interaktiven Release-Upgrade-Prozess
`systemctl --failed`	Listet Dienste auf, die nach dem Systemstart fehlgeschlagen sind

Phase / Befehl	Funktion / Erläuterung
`sudo netplan status`	Zeigt Schnittstellen, Adressen und Routing-Status des modernisierten Netplan 1.0
`sudo dpkg --configure -a`	Konfiguriert unvollständig installierte Pakete nach einem unterbrochenen Upgrade
`sudo apt install -f -y`	Repariert beschädigte Paketabhängigkeiten im Notfall

Weiterführende Ressourcen

Die folgenden internen Dokumentationen und Referenzen vertiefen den sicheren Betrieb und weiterführende Upgrades:

Ressource / Dokumentation	Beschreibung
Ubuntu-Upgrade: 24.04 auf 26.04 LTS	Der nächste logische Schritt: Migration auf Kernel 7.0, `sudo-rs` und APT 3.2 Rollback
LPIC-1 Serie: Linux-Grundlagen	Fundiertes Wissen zu Paketmanagement, Shell-Workflows und Linux-Systemarchitektur
Virtuelle Maschinen für Tests	Testumgebungen für risikofreie Probe-Upgrades unter Proxmox VE und KVM einrichten
Boot-Management und Systemstart	GRUB-Konfiguration, Initramfs und Systemstart-Diagnose bei Kernel-Problemen
Ubuntu Serverumgebungen	Übersicht aller Leitfäden für Administration, Härtung und Betrieb von Linux-Servern

Fazit

Ein geordnetes Release-Upgrade von Ubuntu 22.04 LTS auf 24.04 LTS sichert die offizielle Wartung und Sicherheitsunterstützung der Plattform bis 2029. Durch den Wechsel auf Linux-Kernel 6.8, Netplan 1.0 und systemd 255 profitiert das System von breiterer Hardware-Unterstützung, modernem cgroup-Handling und verbesserter Sicherheitsarchitektur. Der Erfolg

eines solchen Upgrades basiert auf disziplinierter Vorbereitung: Ein verifiziertes Backup außerhalb der Root-Partition, ausreichende Speicherplatzreserven und die Absicherung der Terminalsitzung über `screen` minimieren das Betriebsrisiko während des In-Place-Upgrades auf ein Minimum.

> **Praxistipp für den laufenden Betrieb:** Richte nach erfolgreichem Upgrade zwingend das Paket `unattended-upgrades` ein, um sicherheitsrelevante Bugfixes und Patches automatisch im Hintergrund einzuspielen. Beachte zudem, dass Python 3.12 unter 24.04 LTS die Richtlinie PEP 668 strikt erzwingt: Globale `pip`-Installationen sind gesperrt, weshalb systemnahe Automatisierungsskripte in separaten Virtual Environments (`python3 -m venv`) oder über `pipx` gepflegt werden müssen.

Für Systeme, die stets auf der neuesten LTS-Linie betrieben werden sollen, schließt sich an diesen Meilenstein die nächste Upgrade-Stufe an: Im Folgeartikel **Ubuntu-Upgrade: Von Version 24.04 LTS auf 26.04 LTS** führen wir den nächsten Distributionssprung durch – inklusive `cgroup` v2-only Migration, Rust-basiertem `sudo-rs` und APT 3.2 Rollback-Mechanismen.