

Proxmox · Debian-VM-Strategie

Proxmox · Debian-VM-Strategie

“ **abstract:** Zweck Debian-13-VMs auf Proxmox reproduzierbar aus einem Cloud-Init-Template bereitstellen. Dieses Dokument beschreibt den aktuellen CORESTREAM-Standard und verweist für den ausführbaren Stand auf das zentrale OneDev-Repo.

Gültig für: Debian 13 (trixie), Proxmox VE, UEFI/OVMF, qcow2-Images

Inhaltsverzeichnis

- 1. 1 · Standard und Entscheidung
- 2. 2 · Skripte installieren
- 3. 3 · Voraussetzungen vorbereiten
- 4. 4 · Template bauen
- 5. 5 · VM klonen
- 6. 6 · Verifikation
- 7. 7 · User und SSH-Zugriff
- 8. 8 · Goldene Regeln
- 9. 9 · Troubleshooting
- 10. 10 · Source of Truth

1 · Standard und Entscheidung

Modus	Image	Einsatz
cloudinit	debian-13-generic-amd64.qcow2	CORESTREAM-Standard; User, Key, Hostname und IP pro VM

Modus	Image	Einsatz
bake	debian-13-nocloud-amd64.qcow2	Sonderfall für bewusst fest gebackene Images

“ **tip:** Empfehlung Für neue Kunden-, Staging- und Infrastruktur-VMs immer `cloudinit` mit `--img genericcloud` verwenden.

“ **warning:** Image-Falle `nocloud` enthält kein Cloud-Init. Der Skriptparameter `--img genericcloud` verwendet im aktuellen CORESTREAM-Workflow die Datei `debian-13-generic-amd64.qcow2`.

2 · Skripte installieren

Auf jedem neuen Proxmox-Host als `root`:

```
curl -fsSL https://git.corestream.ch/corestream/proxmox-debian-cloudinit-  
scripts/~raw/master/install-on-proxmox.sh | bash
```

Fester Installationspfad:

- `/opt/corestream/proxmox-debian-cloudinit/build-debian-template.sh`
- `/opt/corestream/proxmox-debian-cloudinit/clone-vm.sh`

Kommandos über Symlinks:

- `/usr/local/sbin/build-debian-template`
- `/usr/local/sbin/clone-vm`

“ **note:** > Der OneDev-Raw-Zugriff ist anonym als `Code Reader` freigegeben. Der Installer kann deshalb ohne Token ausgeführt werden.

3 · Voraussetzungen vorbereiten

3.1 Pakete auf Proxmox

```
apt update
apt install -y libguestfs-tools wget curl
```

`qemu-guest-agent` wird nicht für den Proxmox-Host benötigt. Das Build-Skript installiert und verifiziert das Paket innerhalb des Debian-Images.

3.2 Storage und Debian-Image

```
pvesm status --enabled
mkdir -p /mnt/pve/<STORE>/import
cd /mnt/pve/<STORE>/import
wget -O debian-13-generic-amd64.qcow2
https://cloud.debian.org/images/cloud/trixie/latest/debian-13-generic-amd64.qcow2
wget -O SHA512SUMS https://cloud.debian.org/images/cloud/trixie/latest/SHA512SUMS
sha512sum -c --ignore-missing SHA512SUMS
```

Das Build-Skript trennt zwei Rollen:

- **Quellstorage:** NFS mit dem unveränderten qcow2 unter `/mnt/pve/<SOURCE_STORE>/import/`; bei mehreren NFS-Storages erscheint ein Auswahlmenü, alternativ `--source-store <STORE>`.
- **Zielstorage:** lokaler, aktivierter Proxmox-Storage mit Content-Typ `images`; bei mehreren lokalen Zielen erscheint ein Auswahlmenü, alternativ `--target-store <STORE>`.

Die VM-Disk, EFI-Disk und Cloud-Init-Disk liegen immer auf dem gewählten Zielstorage. NFS wird automatisch nicht als VM-Ziel angeboten.

3.3 Admin-Public-Key

Standardpfad:

```
test -s /root/.ssh/id_csgadmin.pub
```

Fehlt die Datei, einen vorhandenen Public-Key beim Build explizit übergeben:

```
build-debian-template -t 9000 -n debian13-init -m cloudinit --img genericcloud --user-key
/root/.ssh/authorized_keys
```

“ **warning:** > Bei `--user-key /root/.ssh/authorized_keys` werden alle dort enthaltenen Keys in die VM übernommen. Für genau einen Admin-Key eine dedizierte Public-Key-Datei verwenden.

4 · Template bauen

Standard:

```
build-debian-template -t 9000 -n debian13-init -m cloudinit --img genericcloud
```

Mit expliziter Key-Datei:

```
build-debian-template -t 9000 -n debian13-init -m cloudinit --img genericcloud --user-key  
/pfad/admin.pub
```

Hermes-Zugriff nur auf ausdrücklichen Wunsch:

```
build-debian-template -t 9000 -n debian13-init -m cloudinit --img genericcloud --hermes --  
hermes-key /pfad/hermes.pub
```

Das Skript:

- kopiert das qcow2 pro Build nach `/var/tmp` und verändert niemals das zentrale NFS-Basisimage
- aktualisiert nur diese Arbeitskopie vor der Paketinstallation
- installiert und verifiziert `qemu-guest-agent`, `sudo` und `openssh-server`
- aktiviert QEMU Guest Agent und serielle Konsole
- deaktiviert Root-SSH und Passwort-SSH
- erzwingt die NoCloud-Datasource
- deaktiviert IPv6 per Sysctl und Kernelparameter `ipv6.disable=1`
- fragt bei mehreren lokalen VM-Storages nach dem Ziel
- importiert VM-, EFI- und Cloud-Init-Disk auf den lokalen Zielstorage
- ermittelt den tatsächlichen importierten Proxmox-Volume-Namen aus `qm config`
- erstellt eine OVMF-/Q35-VM mit SCSI-Disk und Cloud-Init-Drive
- setzt `ciuser=csgadmin`, SSH-Key und DHCP als Ausgangswert
- wandelt die VM in ein Template um

Template-Konfiguration prüfen:

```
qm config 9000 | grep -E 'agent|scsi1|cloudinit|ciuser|boot'
```

Erwartet:

- `agent: 1` beziehungsweise `enabled=1`
- `scsi1` enthält `cloudinit`
- `ciuser: csgadmin`

- Boot-Reihenfolge enthält `scsi0;scsi1`

“ **important:** > Template niemals booten. Immer einen Clone zum Testen verwenden.

5 · VM klonen

Test-VM mit DHCP:

```
clone-vm 9000 901 test-debian dhcp
```

VM mit statischer IPv4-Adresse:

```
clone-vm 9000 121 app-staging 10.10.10.121/24 10.10.10.1
```

Der Clone-Helper:

- erstellt einen Full Clone
- setzt Proxmox-seitig `agent enabled=1`
- setzt DHCP oder statische IPv4-Konfiguration
- startet die VM

Für Throwaway-Tests kann bei Bedarf manuell ein Linked Clone erstellt werden. Kunden-, Staging-, Produktions- und Infrastruktur-VMs bleiben Full Clones.

6 · Verifikation

6.1 Im Gast

```
cloud-init status --long
id csgadmin
hostname
ip -4 a
ls -l /dev/disk/by-label/cidata
dpkg-query -W qemu-guest-agent
systemctl is-active qemu-guest-agent
sysctl net.ipv6.conf.all.disable_ipv6
```

```
cat /proc/cmdline  
ip -6 addr
```

Erwartet:

- Cloud-Init: `status: done`
- `csgadmin` existiert
- Hostname und IPv4-Konfiguration stimmen
- `cidata` ist vorhanden
- `qemu-guest-agent` ist installiert
- Guest-Agent-Dienst ist `active`
- `disable_ipv6 = 1`
- Kernel-Commandline enthält `ipv6.disable=1`
- keine IPv6-Adressen

6.2 Auf Proxmox

```
qm agent <VMID> ping
```

SSH-Test:

```
ssh csgadmin@<IPv4>
```

7 · User und SSH-Zugriff

User	Standard	Key	Rechte
<code>csgadmin</code>	immer im Cloud-Init-Modus	<code>--user-key</code> , standardmässig <code>/root/.ssh/id_csgadmin.pub</code>	Admin-Zugang gemäss Cloud-Image/Cloud-Init
<code>hermes</code>	nur mit <code>--hermes</code>	<code>--hermes-key</code> , standardmässig <code>/root/.ssh/id_ed25519.pub</code>	<code>NOPASSWD:ALL</code> , Passwort gesperrt
<code>root</code>	kein SSH-Login	keiner	<code>PermitRootLogin no</code>

Zusätzlich gilt `PasswordAuthentication no`; SSH-Zugriff erfolgt per Key.

⚡ important: > Hermes ist opt-in. Der User wird nie automatisch angelegt.

8 · Goldene Regeln

1. Für Cloud-Init `debian-13-generic-amd64.qcow2` verwenden.
2. Template nie booten.
3. Cloud-Init-Drive muss vorhanden sein, bei OVMF auf `scsi1`.
4. Nach Änderungen an `ci*` immer `qm cloudinit update <VMID>`.
5. Kunden-/Staging-/Prod-/Infra-VMs als Full Clone erstellen.
6. Admin-Key vor dem Build verifizieren; der Build bricht bei fehlender oder leerer Key-Datei ab.
7. Nach dem ersten Clone Cloud-Init, Guest Agent, IPv4, SSH und deaktiviertes IPv6 prüfen.
8. Skripte nicht aus Obsidian kopieren; immer den aktuellen OneDev-Stand installieren.

9 · Troubleshooting

Symptom	Ursache	Fix
<code>user pubkey missing</code>	Default-Key-Datei fehlt	dedizierte Datei bereitstellen oder <code>--user-key <datei></code> verwenden
<code>storage ... does not support vm images</code>	Quell-NFS fälschlich als VM-Ziel verwendet oder alter Skriptstand	Installer erneut ausführen; lokalen Zielstorage wählen oder <code>--target-store <STORE></code> setzen
VM-ID existiert nach fehlgeschlagenem Import	alter Build hat die VM vor dem Storage-Fehler angelegt	<code>qm config <VMID></code> prüfen, unbrauchbaren Teilbau gezielt löschen oder neue ID wählen
Basis-qcow2 stimmt nicht mehr mit SHA512 überein	alter Skriptstand hat das NFS-Image direkt verändert	Original erneut von Debian laden; aktueller Skriptstand arbeitet mit <code>/var/tmp</code> -Kopie
Kein SSH-Login	Key nicht im Cloud-Init-Seed	<code>qm config</code> , <code>sshkeys</code> , Key-Datei und <code>qm cloudinit update</code> prüfen
<code>cloud-init: command not found</code>	<code>nocloud</code> -Image verwendet	Template mit <code>generic-amd64</code> neu bauen
<code>cidata</code> fehlt	Cloud-Init-Drive fehlt/falscher Bus	<code>scsi1</code> und <code>qm cloudinit update</code> prüfen
Cloud-Init überspringt	Template wurde gebootet	Template sauber neu bauen; künftig nur Clones booten
<code>qemu-guest-agent</code> fehlt	altes/fehlerhaftes Template	aktuelles Build-Skript installieren und Template neu bauen
<code>qm agent <VMID> ping</code> scheitert	Paket/Dienst/Agent-Flag fehlt	Gastpaket, Dienst, <code>/dev/virtio-ports/</code> und <code>qm config</code> prüfen
IPv6 noch aktiv	alter Template-Stand oder kein Neustart	aktuelles Template bauen; Sysctl, Kernelparameter und Cold Boot prüfen

Symptom	Ursache	Fix
DHCP wartet	kein DHCP im Zielnetz	statische IPv4 beim Clone übergeben

10 · Source of Truth

“ **important:** > Ausführbarer Source of Truth ist OneDev, nicht ein eingebetteter Skriptblock in Obsidian.

- **Repo:** `corestream/proxmox-debian-cloudinit-scripts`
- **SSH:** `ssh://git@git.corestream.ch:6611/corestream/proxmox-debian-cloudinit-scripts.git`
- **Installer:** `https://git.corestream.ch/corestream/proxmox-debian-cloudinit-scripts/~raw/master/install-on-proxmox.sh`
- **Installationspfad:** `/opt/corestream/proxmox-debian-cloudinit`

Die früher eingebetteten Vollkopien der Skripte wurden bewusst entfernt, damit Dokumentation und ausführbarer Code nicht erneut auseinanderlaufen.

Verwandt: Staging- und Produktionskonzept

Quelle: Obsidian `Knowledge Base/Proxmox/Proxmox Debian Cloud-Init Strategie.md` · Stand: 2026-09-13 · Automatisiert publiziert durch Hermes.

Revision #3
Created 2026-10-11 22:57:44 UTC by Hermes
Updated 2026-10-11 23:34:27 UTC by Hermes