Optimierte LinuxVMs unter Proxmox VE

Du möchtest dich gerne für unsere Hilfe erkenntlich zeigen 😳. Gerne. Wir bedanken uns bei dir für deine Spende!

Spenden

Zum frei verfügbaren Apt-Repository

GITLAB:

Verwendete Systeme:

- Proxmox VE 9.x [0]
- Ubuntu-Server 24.04 LTS [1]

Einleitung

In diesem HowTo zeige ich dir einige Möglichkeiten, um eine LinuxVM auf Proxmox VE optimiert zu gestalten. Es gibt natürlich viele verschiedene Einsatzszenarien. Wobei ich die in diesem Beitrag verwendete Konfiguration für die meisten Anwendungsfälle empfehlen kann. Im Laufe dieses Beitrags werde ich auch die wichtigsten Optionen behandeln.

Technische Grundkenntnisse und KnowHow von Proxmox VE, z.B. wie man eine VM in Proxmox VE installiert, werden hier vorausgesetzt. Ich gehe hier nur auf ganz spezifischen Dinge genauer ein, die meines Achtens in diesem Zusammenhang notwendig sind. Feedback ist natürlich immer willkommen.

Tipp an dieser Stelle: "Cloud-Init für LinuxVMs - ganze einfach unter Proxmox VE"

Voraussetzungen

- Laufende Proxmox VE 9.x Node
- Herunter geladenen Ubuntu-Server 24.04 LTS ISO [1]

Neue LinuxVM erstellen

- Tab: „**General** → Keine spezielle Konfiguration
- Tab: “**OS**„ → Guest OS Type „Linux“ mit Version „6.x - 2.6 Kernel“
- Tab: “**System**„:
  - Graphics card: „VirtIO-GPU“ → Je nach Linuxversion hat der Gast Probleme Bootmeldungen anzuzeigen. Dabei werden nur unleserliche Dinge auf der noVNC Konsole

    angezeigt. Der wechsel zu „VirtIO-GPU“ löst dies. Da wir in späterer Folge dieses Artikel ohnehin die serielle Konsole als Default verwenden möchten, ist dies optional.
    - Machine: q35 → Bringt neuere virtuelle Hardware mit.
    - Das BIOS konfigurieren wir auf UEFI → Secureboot ist nur in Sonderfällen notwendig. In unserem Beispiel verzichten wir auf Secure-Boot und deaktivieren “**Pre-Enroll keys**„.
    - Qemu Agent aktivieren wir selbstverständlich. Features und Vorteile darüber findest du auf [2].
- Tab: “**Disks**„:
    - Bus/Device: SCSI → Dies ist der Default, hat unter normalen Umständen die beste Leistung und die meisten Features.
    - Discard aktivieren [3] → Das macht natürlich nur Sinn, wenn dein verwendeter Storagespeicher auch Thin Provisioning unterstützt [4]. Wenn Discard aktiviert ist und das Gastbetriebssystem TRIM unterstützt, leitet der Controller diese Information an den Speicher weiter, sobald das Dateisystem der VM nach dem Löschen von Dateien Blöcke als ungenutzt markiert. Der Speicher verkleinert daraufhin das Disk-Image entsprechend. Damit der Gast TRIM-Befehle ausgeben kann, musst du die Option Discard auf dem Laufwerk aktivieren. Ubuntu macht dies automatisch in Zyklen. Solltest du Live-Discard verwenden wollen, kannst diese Option in der `/etc/fstab` im Gast bei dem gewünschten Laufwerk hinzufügen [5].
    - SSD emulation → Aktivieren, sofern deine physische Storage SSDs oder NVMEs verwendet [3].
    - Eine Änderung der Cache und Async IO Konfiguration kann in bestimmten Fällen von Vorteil sein [6], [7].
- Tab: “**CPU**„: Hier ist es natürlich ganz individuell wie viel Leistung du benötigst und wie sich das auf die verfügbaren CPU-Sockets verteilen soll. Grundsätzlich empfehle ich immer die Socketanzahl die mir zur Verfügung steht. Ab 2 Sockets aktiviere ich auch gleich mal NUMA [8]. Als CPUtype verwende ich in Linux grundsätzlich „host“ da man unter Linux (in Gegensatz zu Microsoft Windows) so gut wie immer die beste Performance erzielt. Aufpassen musst du natürlich, wenn du einen Cluster betreibst und die VM Online zwischen verschieden physischen CPUtypen migrieren möchtest. Sollte dies der Fall sein, verwende bitte eine generische Variante wie z.B. x86-64-v2-AES.
- Tab: “**Memory**„: Auch wieder individuell → Default OK.
- Tab: “**Network**„: Auch wieder individuell → Solltest du ein 10 Gbit Netzwerk betreiben, wirst du die Option „Multiqueue“ [9] interessant finden.

<u>Nachdem die VM erstellt wurde, fügen wir noch zwei zusätzliche Geräte hinzu:</u>

- Serielles Port → Für die serielle Konsole in Linux. Damit wird dir eine echte Konsole zur Verfügung gestellt, über diese du auch wie gewohnt mit copy/paste kopieren und einfügen kannst. (Konfiguration folgt weiter unten).
- VirtIO RNG Device [10] → Liefer der VM Entropy. Gerade in Linux kann das die Performance in gewissen Situationen erheblich steigern (Bootprozess, SSL/TLS usw.)

Tipp: Verwende pro virtueller Festplatte nur eine Partition, keine LVM oder sonstiges. Damit ist ein nachträgliches Vergrößern am einfachsten (excluded Boot/EFI).

Nach fertiger Installation des Betriebssystems in der VM...

Nachdem du nun deinen Server erfolgreich installiert hast, können wir uns der Konfiguration des Betriebssystems widmen. Für die korrekte Funktion des Qemu-Agents [13] ist in der VM noch die Installation des gleichnamigen Paketes erforderlich:

```
apt install qemu-guest-agent
```

Danach funktionieren auch Sachen wie „Freeze on Backup“ und es werden dir auch hilfreiche Informationen über das Netzwerk in Proxmox VE von der VM angezeigt.

Konfiguration Serielles Port

Damit wir den seriellen Output ansprechen können, musst du die folgende Datei modifizieren `/etc/default/grub` und die Grubconfig aktualisieren. Für das serielle Port ist diese Zeile wichtig `GRUB_CMDLINE_LINUX_DEFAULT`.

```
- GRUB_CMDLINE_LINUX_DEFAULT=""
+ GRUB_CMDLINE_LINUX_DEFAULT="console=ttyS0,115200 fsck.repair=preen
verbose"
```

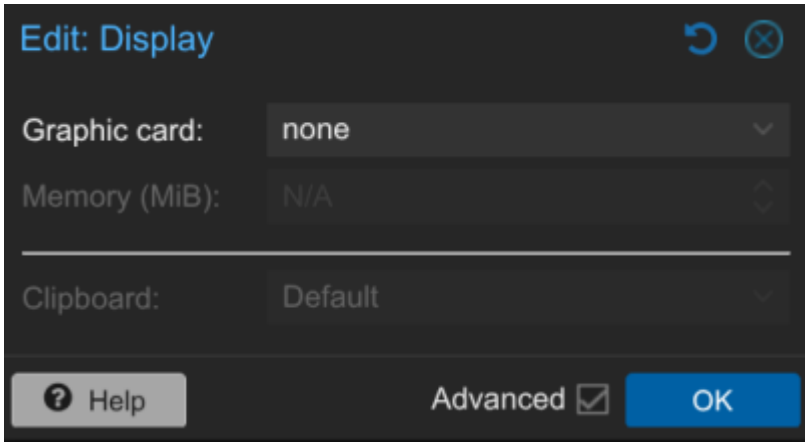
Genau, du hast dies richtig erkannt. Es gibt zwei zusätzliche Optionen. Diese sind für die Verwendung der seriellen Konsole nicht erforderlich, möchte ich aber im Zuge dieser Config empfehlen.

- fsck.repair=preen → Im Falle einer gravierenden Dateisystembeschädigung, die normalerweise manuell repariert werden müsste, tut dies das System selbst, sollte so eine Situation eintreffen.
- verbose → Zeigt dir alle Bootmeldungen immer visuell an.

Nachdem Abspeichern aktualisieren wir noch Grub `update-grub`, und nach dem Reboot kannst du die serielle Konsole in Proxmox VE bereits mit **xterm.js** oder auch auf der Proxmox-Root-Shell direkt verwenden:

```
qm terminal <vmid>
```

Solltest du ausschließlich das serielle Terminal benötigen, könntest du auch einfach die virtuelle Grafikkarte komplett aus der VM entfernen:



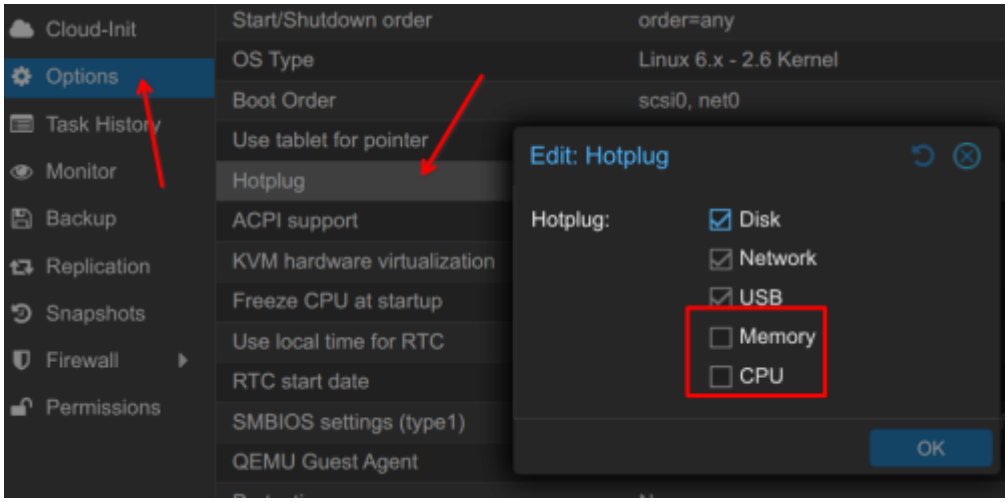
CPU/Memory Hot-add/Hot-remove (Optional)

Unter Proxmox VE ist es recht einfach Linuxgästen Hotplug von Geräten beizubringen. Für CPU und Memory ist zusätzlich noch eine UDEV Regel erforderlich [11].

```
nano /lib/udev/rules.d/80-hotplug-cpu.rules
```

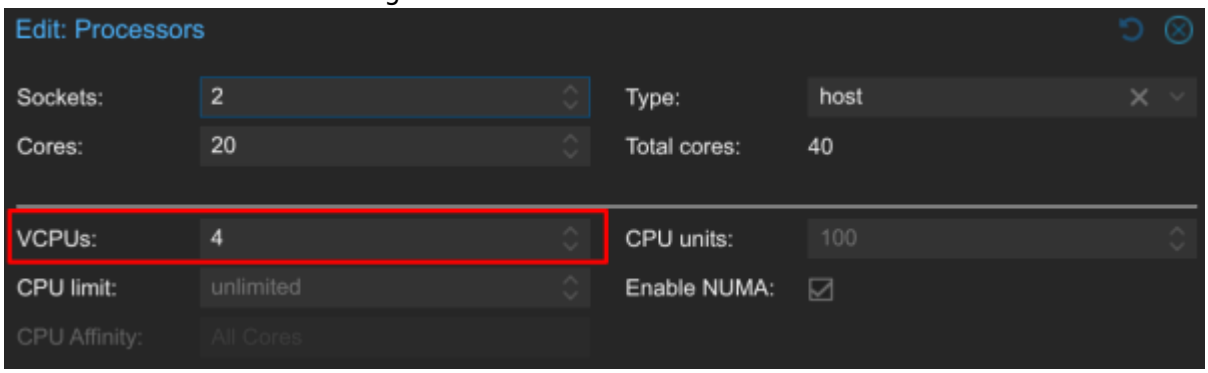
```
SUBSYSTEM=="cpu", ACTION=="add", TEST=="online", ATTR{online}=="0",
ATTR{online}="1"
```

Weiters muss das Feature noch aktiviert werden:



Es ist auch die CPU-Option „NUMA“ [12] erforderlich um Memory-Hotplug verwenden zu können.

Für CPU Hotplug weise der VM alle Sockets Cores zu, die dein physischer Server zur Verfügung stellt. Wie viel Cores die VM wirklich bekommt, steuerst du mit den „VCPUs“. Hier ein Beispiel für einen Server mit 2 Sockets und insgesamt 40 Cores:



Die VM hat nun also 2 Sockets mit insgesamt 4 Cores. Dies kannst du jetzt online verändern. Du hast die Möglichkeit Cores und Memory hinzuzufügen und auch wieder zu entfernen. Dies funktioniert alles im Betrieb, ein Neustart ist nicht erforderlich.

Memory Hot-remove benötigt je nachdem wie viel RAM weggeworfen wird einige Zeit. Achte auch darauf das der RAM, den du entfernst, nicht verwendet wird!! `6 GB RAM vorhanden, 4 GB belegt, du reduzierst auf 2 GB = bad idea`

Installation zusätzlicher Pakete und Konfiguration

Um das ganze noch etwas hübscher zu gestalten, hast du auch die Möglichkeit einige zusätzliche Pakete zu installieren und Konfigurationen anzupassen.

- Avahi MulticastDNS → Damit kannst du deine VMs, gerade für Testing, per FQDN ansprechen auch ohne das du einen DNS-Server pflegen musst. Die Avahidomäne kann natürlich auch auf eine echte Domäne angepasst werden. Default Domäne ist LOCAL.
- ZSH-Shell [14]
- Speedtest-cli → Ein Lowlevel Speedtest. Die Nutzung von Speetests von Webanbietern benötigen ja selbst auch Ressourcen, mit dem kleinen Tool ist das dann kein Thema mehr.
- Nload → zeigt dir die Netzwerkgeschwindigkeit von Interfaces. z.B.: `nload -t 100 -u M <interface>`
- Nmap → Advanced Portscanner, z.B. ein Pingscan des gesamten Netzwerks: `nmap -sn 172.16.10.0/24 |grep “(172.16.10„`

```
apt install avahi-daemon avahi-utils zsh speedtest-cli nload nmap
```

Du kannst auch die Willkommensmeldung auf deinem Server anpassen. Lege hierfür ein File unter `/etc/update-motd.d/` an:

```
nano /etc/update-motd.d/99-welcome
```

99-welcome

```
#!/bin/sh

green="\e[0;92m"
red="\e[31m"
bold="\e[1m"
reset="\e[0m"

echo
echo "${green} ------------------------------------------------"
echo " Willkommen am Ubuntu Cloud-init Template 24.04"
echo " ------------------------------------------------${reset}"
echo
echo "${red}${bold}To update Welcome Message change /etc/update-
motd.d/99-welcome"
echo "and remove this line. ${reset}"
echo
echo
```

Oder für deine Shell noch einen Sprüchklopfer wie Fortune?

```
apt install fortunes fortunes-de fortune fortune-mod
```

Um Fortunes zu aktivieren, füge in deiner `~.zshrc` folgendes am Ende ein (Deutsche Weisheiten):

```
/usr/games/fortune /usr/share/games/fortunes/de/
```

Weitere Kategorien/Sprachen findest du in diesem Verzeichnis `/usr/share/games/fortunes`.

Weitere Empfehlungen

- Aktivierung von automatischen Updates
- Grub Menü beim Boot ohne Submenü einblenden
- Rsyslogserver Remotelogging
- Cloud-Init für LinuxVMs - ganze einfach unter Proxmox VE

Links

- [0] https://pve.proxmox.com/pve-docs/pve-admin-guide.html
- [1] https://ubuntu.com/download/server
- [2] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_qemu_agent
- [3] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_hard_disk_discard
- [4] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#_thin_provisioning
- [5] https://wiki.ubuntuusers.de/SSD/TRIM/#Online-Discard
- [6] https://pve.proxmox.com/wiki/Performance_Tweaks
- [7] https://kb.blockbridge.com/technote/proxmox-aio-vs-iouring/
- [8] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#_numa
- [9] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_network_device
- [10] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_virtio_rng
- [11] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#_vcpu_hot_plug
- [12] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#_numa
- [13] https://pve.proxmox.com/pve-docs/pve-admin-guide.html#qm_qemu_agent
- [14] https://github.com/boospy/ZSH-und-BASH-Configs

From:
https://wiki.deepdoc.at/dokuwiki/ - **DEEPDOC.AT - enjoy your brain**

Permanent link:
https://wiki.deepdoc.at/dokuwiki/doku.php?id=virtualisierung:proxmox_kvm_und_lxc:optimierte_linuxvms_unter_proxmox_ve

Last update: **2026/03/23 11:26**