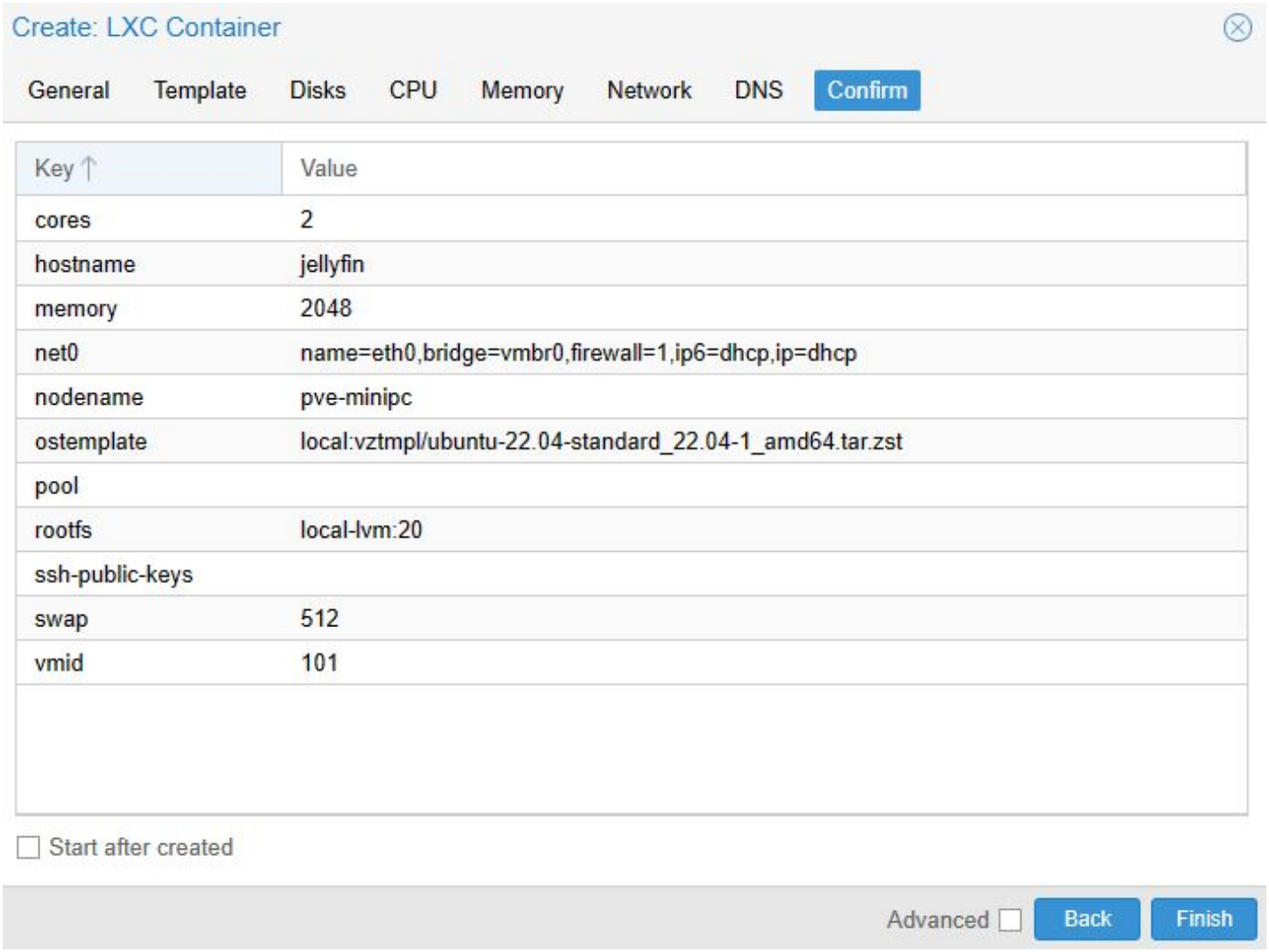
Proxmox

Information and guides for Proxmox VE and Proxmox Backup Server

- Proxmox Host iGPU Drivers & Modifications
- Proxmox Backup Server PBS enlarge root disk
- Removing Proxmox Subscription Notice
- Proxmox Backup Server Disable Subscription
- Proxmox customize shell
- NFS share on OpenMediaVault
- Step-by-Step Guide: Running Ollama in Proxmox Debian 12 LXC with Intel iGPU (i5-1240P)
- Generate Certificate for Proxmox.
- Auto upgrade Proxmox VE (Script)
- Ollama with Intel GPU acceleration on Proxmox
- Backup Proxmox Script
- Proxmox Upgrade Pve8 to Pve9 Script
- Proxmox Kernel Manager

Proxmox Host iGPU Drivers & Modifications

Finish and make sure the Container doesn't start after creation.



The container for Jellyfin on Proxmox is configured, but we need to make a few host changes first. This is going to be the most difficult part as the drivers you need to install will depend on the type of CPU and iGPU you're using.

For the most part, you just have to ensure that the drivers are installed and if they are, you'll have access to your iGPU for Jellyfin on Proxmox. We'll install a package that will tell you if it is or isn't working before you proceed.

1. Access the **Shell** on your Proxmox host and run the commands below to install a few of the packages we'll need:

Add block to other drivers and leave only yours.

```
nano /etc/modprobe.d/blacklist.conf
```

and fill the below inside the file `blacklist.conf`

```
blacklist amdgpu
blacklist radeon
blacklist nouveau
blacklist nvidia\*
#blacklist i915
#blacklist iHD
```

Save Ctrl+x and Enter

Add driver to your Proxmox host

```
nano /etc/modprobe.d/i915.conf
```

and fill the below inside the file `i915.conf`

```
options i915 enable_guc=3
```

Save Ctrl+x and Enter and REBOOT Proxomox host

Install the below GPU driver packages

vainfo: This package will test to ensure the GPU drivers are installed and functioning properly.

```
apt install vainfo -y
```

Intel Drivers: First, we'll add the non-free repository to add the Intel drivers needed, then install the Intel iGPU drivers. Keep in mind that if you aren't using an Intel CPU (or even possibly if you're using an older/newer Intel CPU than I am), this step *might* be different for you.

```
apt install software-properties-common -y && apt install intel-media-va-driver-non-free -y
```

2. Now that both are installed, run the command below to confirm the GPU drivers are installed properly. If they are, you should see a bunch of supported profiles and entrypoints.

Confirm the GPU drivers are installed properly.

```
vainfo
```

If you find this helpful or not please leave a comment.

Ins0mniA

Proxmox Backup Server PBS enlarge root disk

How to enlarge an EXT4 or LVM disk on an PBS Proxmox VM

Quick TL;DR Guide

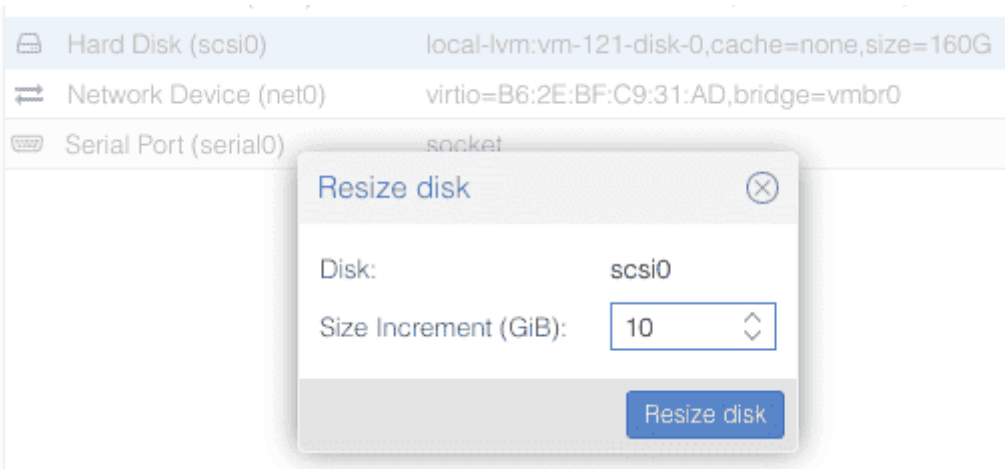
1. First resize the disk in Proxmox
2. Then in the PBS VM terminal:
   parted /dev/sdX (probably a) example

   ```
   parted /dev/sda
   ```

   print / yes to fix / resizepart Y (probably 1)
   quit
3. resize2fs /dev/sdXY
4. df -h

Longer Guide with Explanation

First resize the disk in Proxmox:



At this point you need to START the PBS VM and enter the shell as root follow the below instractions.

```
fdisk -l
```

should show which disks are there.

Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Disk model: QEMU HARDDISK

Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX

Device   Start   End   Sectors Size Type
/dev/sda1   34   2047   2014 1007K BIOS boot
/dev/sda2   2048   1050623   1048576 512M EFI System
/dev/sda3 1050624 104855551 103804928 31.5G Linux LVM

Disk /dev/mapper/pbs-swap: 3.82 GiB, 4106223616 bytes, 8019968 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

Disk /dev/mapper/pbs-root: 27.67 GiB, 49039802368 bytes, 95780864 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes

If you get this message which seems like an error, the resize in Proxmox worked:

```
GPT PMBR size mismatch (167772159 != 314572799) will be corrected by write.
The backup GPT table is not on the end of the device. This problem will be corrected by write.
```

The example above is resizing from 30 GB to 50 GB.

Now we still have to resize the file system from inside the PBS VM cli terminal :

```
lsblk
```

You should see something like the below.

NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINTS
sda   8:0   0   50G 0 disk
├─sda1   8:1   0 1007K 0 part
├─sda2   8:2   0 512M 0 part
└─sda3   8:3   0 31.5G 0 part
  ├─pbs-swap 252:0   0 3.8G 0 lvm [SWAP]
  └─pbs-root 252:1   0 27.7G 0 lvm /

The run the below command to resize the sda3

```
lvextend -l +100%FREE /dev/mapper/pbs-root
resize2fs /dev/mapper/pbs-root
```

Check that root disk resize all free space

```
df -h
```

Output should be something like this.

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	3.9G	0	3.9G	0%	/dev
tmpfs	795M	720K	794M	1%	/run
/dev/mapper/pbs-root	45G	26G	18G	59%	/
tmpfs	3.9G	0	3.9G	0%	/dev/shm
tmpfs	5.0M	0	5.0M	0%	/run/lock
tmpfs	795M	0	795M	0%	/run/user/0

Now your PBS root disk is resize and have free space.

If you find this helpful or not please leave a comment.

Ins0mniA

Removing Proxmox Subscription Notice

First method

To remove the “You do not have a valid subscription for this server” popup message while logging in and when refreshing packages to do updates.

You’ll need to SSH to your Proxmox server or use the node console through the PVE web interface. One note ctl-w to search in nano closes the tab in firefox so using an SSH client like putty works better.

Login to your proxmox server via ssh

You can change directories,

```
cd /usr/share/javascript/proxmox-widget-toolkit
```

Make a backup

```
cp proxmoxlib.js proxmoxlib.js.bak
```

or run,

```
cp /usr/share/javascript/proxmox-widget-toolkit/proxmoxlib.js /usr/share/javascript/proxmox-widget-toolkit/proxmoxlib.js.bak
```

Then open the file in nano

```
nano /usr/share/javascript/proxmox-widget-toolkit/proxmoxlib.js
```

or

```
nano proxmoxlib.js
```

While in nano use **ctrl-w** to search for

```
No valid subscription
```

Before nano edit proxmoxlib.js

You should find the below

```
.data.status.toLowerCase() !== 'active') {
                    Ext.Msg.show({
                       title: gettext('No valid subscription'),
```

Go and REMOVE the symbol **!** before the **!== 'active'**

After nano edit proxmoxlib.js

Show look like the below

```
.data.status.toLowerCase() == 'active') {
                    Ext.Msg.show({
                       title: gettext('No valid subscription'),
```

Now to save the file **ctrl-x** and press **Y** and **enter** to save exit nano.

Restart the ProxMox service

```
systemctl restart pveproxy.service
```

Reload your browser tab and log back in.

This just changes the logic of the code from not active to is active.
If you pay for a subscription then you would need to change it back.

Second method

To use the Proxmox VE Post Install script, run the command below in the shell.

```
curl -fsSL https://docs.greenhome.stream/attachments/28 -o proxmox_subscription_autofix.sh &&
chmod +x proxmox_subscription_autofix.sh && ./proxmox_subscription_autofix.sh
```

Download Script proxmox_subscription_autofix.sh

? One-Line Download & Execute:

```
clear && curl -fsSL https://docs.greenhome.stream/attachments/28 -o
proxmox_subscription_autofix.sh && chmod +x proxmox_subscription_autofix.sh && clear &&
./proxmox_subscription_autofix.sh
```

Attachment Link proxmox_subscription_autofix.sh

? How to Use (Super Simple):

```
# 1. Save the script
nano proxmox_subscription_autofix.sh
# (paste the script content above)

# 2. Make executable
chmod +x proxmox_subscription_autofix.sh

# 3. Run once as root - it does EVERYTHING automatically!
./proxmox_subscription_autofix.sh

# 4. Reboot and enjoy - works forever automatically!
reboot
```

? What Happens When You Run It:

First Run Output Example:

```
[STEP] Checking systemd service setup...
[WARNING] ⚠ Systemd service file not found
[INFO] Setting up automatic service installation...
[STEP] Installing script to system location...
[INFO] ✓ Script installed to: /usr/local/bin/proxmox_subscription_autofix.sh
[STEP] Creating systemd service file...
[INFO] ✓ Service file created: /etc/systemd/system/proxmox-subscription-autofix.service
[STEP] Enabling and configuring systemd service...
[INFO] ✓ Systemd daemon reloaded
[INFO] ✓ Service enabled for automatic startup
[INFO] □ Systemd service installed and enabled successfully!
[INFO] Checking Proxmox subscription notice status...
[WARNING] ⚠ Original logic detected - subscription notice is active
[STEP] Applying Logic Inversion fix...
[INFO] □ Logic Inversion fix applied and verified successfully!
[INFO] □□Proxmox subscription notice has been disabled automatically!

=== AUTO-FIX SUMMARY ===
[INFO] Current Status: SUBSCRIPTION NOTICE DISABLED □
[INFO] Logic Inversion: ACTIVE
[INFO] Systemd Service: ENABLED □
[NOTICE] Next boot will automatically run this check again
```

? Benefits of This Enhanced Version:

□□ FULLY AUTONOMOUS - One script does everything
□□ SELF-INSTALLING - Creates its own service automatically
□□ INTELLIGENT - Only creates service if missing
□□ STATUS AWARE - Shows current service and fix status
□□ ZERO MAINTENANCE - Set once, works forever
⚡ INSTANT SETUP - No multiple files or complex installation

Now you have one single script that handles everything automatically - from fixing the subscription notice to installing itself as a permanent service!

Ins0mniA

Proxmox Backup Server Disable Subscription

Description

The script will give options to Disable the Enterprise Repo, Add/Correct PBS Sources, Enable the No-Subscription Repo, Add Test Repo, Disable Subscription Nag, Update Proxmox Backup Server and Reboot PBS.

Proxmox Backup Server ONLY
Execute **within the Proxmox Backup Server Shell**
It is recommended to answer “yes” (y) to all options presented during the process.

How to use

To use the Proxmox Backup Server Post Install script, run the command below in the shell.

```
bash -c "$(wget -qLO - https://github.com/community-scripts/ProxmoxVE/raw/main/misc/post-pbs-install.sh)"
```

Reboot PBS.

If you find this helpful or not please leave a comment.

Ins0mniA

Proxmox customize shell

? One-Line Download & Execute :

```
apt update && apt install -y curl && clear && curl -s
https://docs.greenhome.stream/attachments/52 | bash
```

Option one

Go to the below path

```
cd /etc/profile.d/
```

Create file 00_lxc-details.sh

and then create a file `00_lxc-details.sh` by editing with nano

```
nano /etc/profile.d/00_lxc-details.sh
```

and add the below text and changed according.

```
echo -e ""
echo -e "[1mGreen Home Bookstack LXC Container[m"
echo -e "    □□ [m[33m Provided by: Green Home[m"
echo ""
echo -e "    □□□ [m[33m OS: [1;92mDebian GNU/Linux - Version: 12[m"
echo -e "    □□ [m[33m Hostname: [1;92m$(hostname)[m"
echo -e "    □□ [m[33m IP Address: [1;92m$(hostname -I | awk '{print $1}')[m"
echo -e ""
```

then press **Ctrl-x** then press **Y** and press **Enter**
To save and exit.

Create file

NFS share on OpenMediaVault

If anyone is looking for the way to make this work for an NFS share on OpenMediaVault, use the following share options:

```
subtree_check,insecure,no_root_squash,anonuid=100,anongid=100
```

and make sure the folder you are sharing is owned by group **'users' (gid 100)**

Ins0mniA

Step-by-Step Guide: Running Ollama in Proxmox Debian 12 LXC with Intel iGPU (i5-1240P)

Step-by-Step Guide: Running Ollama in Proxmox Debian 12 LXC with Intel iGPU (i5-1240P)
This guide will show you how to:

1. Enable Intel iGPU passthrough to an LXC container in Proxmox.
2. Set up the latest Intel graphics drivers on both the Proxmox host and the container.
3. Install and use Ollama in the container.

1. Prepare Proxmox Host for Intel iGPU Passthrough

a) Install Essential Packages:

```
apt update && apt install -y intel-opencl-icd
```

b) Enable IOMMU and iGPU Features:
Edit GRUB configuration:

```
nano /etc/default/grub
```

Find the line starting with **GRUB_CMDLINE_LINUX_DEFAULT** and change it to:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on i915.enable_gvt=1"
```

Update GRUB:

```
update-grub
```

c) Load Required Kernel Modules:
Edit /etc/modules

```
nano /etc/modules
```

and add:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
kvmgt
exngt
vfio-mdev
```

Update initramfs:

```
update-initramfs -u -k all
```

d) Enable GUC for i915 (for best iGPU performance, including newer Intel chips):

```
echo "options i915 enable_guc=3" >> /etc/modprobe.d/i915.conf
```

e) Reboot the **Proxmox Host**

```
reboot
```

f) Check iGPU Availability:
After reboot:

```
lspci -nnv | grep VGA
dmesg | grep -e DMAR -e IOMMU
```

You should see the Intel iGPU and confirmation that **IOMMU is enabled**.

2. Configure Proxmox LXC Container for iGPU Passthrough

a) Stop the LXC Container (if running):

```
pct stop <container_id>
```

b) Edit the Container’s Configuration:

```
nano /etc/pve/lxc/<container_id>.conf
```

Add the following lines:

```
lxc.cgroup2.devices.allow: c 226:0 rwm
lxc.cgroup2.devices.allow: c 226:128 rwm
```

```
lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
```

c) Start the Container

```
pct start <container_id>
```

3. Install Latest Intel Graphics Drivers in Debian 12 LXC

Recent Intel drivers for iGPU (including Alder Lake in i5-1240P) are included in the Debian 12 kernel and Mesa packages. Extra steps are only needed if you want bleeding-edge features, but most users do not require them.

a) Update the Container

```
apt update && apt upgrade -y
```

b) Ensure Video Group Membership (for user who'll run Ollama):

```
usermod -aG video <your_username>
usermod -aG render <your_username>
```

c) Test iGPU Access:
Inside the container, run:

```
ls /dev/dri
```

You should see "**renderD128**" (and maybe "**card0**")

Check VAAPI info:

```
apt install vainfo -y
vainfo
```

You should see details about your Intel iGPU.

4. Install Ollama in the LXC Container

a) Install Prerequisites:

```
apt install -y curl
```

b) Install Ollama:

```
curl -fsSL https://ollama.com/install.sh | sh
```

This script sets up the **Ollama binary, user, systemd service, and necessary groups**; Ollama will start running on **localhost:11434**

c) (Optional) Enable API Access from Outside Container:
Edit the service to listen on all IPs:

```
systemctl edit ollama.service
```

Add:

```
[Service]
Environment="OLLAMA_HOST=0.0.0.0"
```

Then restart:

```
systemctl restart ollama.service
```

5. Test Ollama

Example usage:

```
ollama pull llama3:8b
ollama run llama3:8b
```

6. Troubleshooting

- If **/dev/dri** is missing, double-check host config and LXC conf file.
- Ensure container is privileged. Unprivileged LXC passthrough is not generally recommended for iGPU.
- iGPU passthrough is best-supported for media and AI apps in newer Intel chips (11th/12th gen+).
- For advanced iGPU features, stay up-to-date with Proxmox and kernel updates.

You now have Ollama running inside a Debian 12 LXC container on Proxmox with full Intel iGPU (i5-1240P) support and the latest available drivers built into Debian!

Ins0mniA

Generate Certificate for Proxmox.

Generate Certificate for Proxmox.

```
# SSH into your Proxmox server
ssh root@172.17.1.10

# Navigate to the certificates directory
cd /etc/pve/nodes/$(hostname)/

# Generate a new private key
openssl genrsa -out proxmox-custom.key 2048

# Create a certificate signing request
openssl req -new -key proxmox-custom.key -out proxmox-custom.csr \
  -subj "/C=US/ST=State/L=City/O=Organization/CN=pve1.mydomain.com"

# Create a self-signed certificate (valid for 365 days)
openssl x509 -req -in proxmox-custom.csr -signkey proxmox-custom.key \
  -out proxmox-custom.crt -days 365

# Combine certificate and any intermediate certificates
cat proxmox-custom.crt > pveproxy-ssl.pem

# Copy the private key
cp proxmox-custom.key pveproxy-ssl.key

# Set proper permissions
chmod 600 pveproxy-ssl.key

# Restart the Proxmox web service
systemctl restart pveproxy
```

Ins0mniA

Auto upgrade Proxmox VE (Script)

? One-Line Download & Execute:

```
clear && curl -fsSL https://docs.greenhome.stream/attachments/40 -o auto-upgrade-proxmox.sh && chmod +x auto-upgrade-proxmox.sh && clear && ./auto-upgrade-proxmox.sh
```

Here’s a **clear step-by-step guide** explaining what the `auto-upgrade-proxmox.sh` script does, and how you can **set up cron jobs** to:

- Run it **daily at 03:00**
- Run it **weekly on Sundays at 04:00**, and reboot the system if the script completes successfully.

? Script Purpose: What Does `auto-upgrade-proxmox.sh` Do?

This script is designed to **automatically update your Proxmox system** using `apt`.

? How to Add Cron Jobs

To schedule this script, you need to edit the system crontab or the root user's crontab.

> ⚠ Updates and reboots require **root permissions**, so make sure the script is owned by root and executable.

? Make Script Executable

```
sudo chmod +x /path/to/auto-upgrade-proxmox.sh
```

? Add Cron Jobs

Edit the **root crontab**:

```
sudo crontab -e
```

Add the following lines at the bottom:

```
# Daily at 03:00 — Run update only
0 3 * * * /path/to/auto-upgrade-proxmox.sh >> /var/log/auto-upgrade.log 2>&1

# Weekly on Sunday at 04:00 — Run update and reboot if successful
0 4 * * 0 /path/to/auto-upgrade-proxmox.sh && /sbin/reboot
```

? Explanation of Cron Syntax

Field	Meaning
`0 3 * * *`	Every day at **03:00 AM**
`0 4 * * 0`	Every **Sunday at 04:00 AM**
`&& /sbin/reboot`	Only reboot if the script exits with success (code 0)

? Summary

Task	Schedule	Action
Run update script daily	03:00 every day	Just updates and notifies
Weekly update + reboot	04:00 Sundays	Updates, notifies, and reboots on success

---

? Optional: Logging Output

You can monitor the script's output in `/var/log/auto-upgrade.log`.

To ensure logs don't grow indefinitely, consider adding log rotation or using `logrotate`.

Ins0mniA

Ollama with Intel GPU acceleration on Proxmox

Run Ollama with Intel GPU acceleration on Proxmox by exposing the iGPU to a Linux container/VM and enabling Intel's oneAPI/SYCL or OpenVINO backends for model offloading to the Xe iGPU. Below is a clean, step-by-step path using a privileged Ubuntu LXC with GPU device mapping, plus an optional OpenVINO backend route if higher throughput is desired on Intel hardware.

What you'll build

- A Proxmox host with IOMMU and i915 GuC enabled for Alder Lake Xe graphics, exposing /dev/dri to guests for GPU compute access.
- A privileged Ubuntu LXC that gets direct access to the iGPU render node, where Intel oneAPI and Ollama run with GPU offload via SYCL (IPEX-LLM) or OpenVINO.
- Optional: a full VM with PCI passthrough if containers are not preferred, though Intel iGPU passthrough is fussier on Alder Lake, especially on Windows.

Step 1: Prep Proxmox host (IOMMU + i915 GuC)

- In BIOS/UEFI, enable VT-d (Intel IOMMU) for device assignment; this is required for clean device mapping/passthrough.
- Enable IOMMU on Proxmox and update boot config:
  - Edit GRUB:

```
sudo nano /etc/default/grub
# Ensure this includes:
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt"
```

Then apply and reboot:

```
sudo update-grub
sudo reboot
```

Validate after reboot:

```
dmesg | grep -e DMAR -e IOMMU
```

(You should see IOMMU/DMAR enabled messages.)[^1][^4]

- Enable GuC for Intel i915 (better scheduling/perf on Xe iGPU):

```
echo "options i915 enable_guc=3" | sudo tee /etc/modprobe.d/i915.conf
sudo update-initramfs -u -k all
sudo reboot
```

After reboot, confirm the render node exists:

```
ls -l /dev/dri
# Expect renderD128 at minimum
```

Step 2: Create a privileged Ubuntu LXC and map the GPU

- Create a privileged Ubuntu 22.04/24.04 LXC in Proxmox (privileged avoids extra uid/gid mapping hurdles with GPU devices).
- Add these lines to the container's config at /etc/pve/lxc/.conf to pass the render node and allow access:

```
lxc.cgroup2.devices.allow: c 226:0 rwm
lxc.cgroup2.devices.allow: c 226:128 rwm
lxc.mount.entry: /dev/dri/renderD128 dev/dri/renderD128 none bind,optional,create=file
```

Start the container and verify /dev/dri/renderD128 exists inside it.

- Note: This also works for unprivileged containers with the correct device rules, but privileged is simpler; if staying unprivileged, consult a known working pattern for iGPU into unprivileged LXC.
- Alternative (optional): Use a full VM with PCI passthrough of the iGPU; this can work well on Linux guests, but Windows guests on Alder Lake can throw Code 43 errors without careful tuning.

Step 3: Install Intel runtimes and Ollama (inside the LXC)

- Install Intel oneAPI Base runtime and prerequisites, then use Intel's IPEX-LLM integration to prime Ollama for Intel GPU offload via SYCL/Level Zero.
- In Ubuntu LXC:

```
sudo apt update
sudo apt install -y python3.11-venv
python3.11 -m venv ~/llm_env
source ~/llm_env/bin/activate
pip install --pre --upgrade ipex-llm[cpp]
mkdir -p ~/llama-cpp && cd ~/llama-cpp
# Initialize Ollama with Intel GPU support via IPEX-LLM helper
init-ollama
```

Intel’s guide uses a Python venv, installs ipex-llm[cpp], then initializes an Ollama build configured for Intel GPUs.

- Optional sanity check: ensure the container can see the Intel GPU compute stack by confirming OpenCL exposure if installed; many users validate with clinfo after installing Intel’s OpenCL runtime.

Step 4: Run Ollama accelerated on the Intel iGPU

- Set recommended environment variables for full layer offload and Level Zero behavior, then start the service:

```
# From inside the LXC (activate venv if needed)
export OLLAMA_NUM_GPU=999
export no_proxy=localhost,127.0.0.1
export ZES_ENABLE_SYSMAN=1
export SYCL_CACHE_PERSISTENT=1
# If oneAPI setvars is installed/system-wide, source it:
source /opt/intel/oneapi/setvars.sh || true
# For certain kernels/GPUs this can help:
export SYCL_PI_LEVEL_ZERO_USE_IMMEDIATE_COMMANDLISTS=1
# Optional: listen on all interfaces if exposing to LAN
# export OLLAMA_HOST=0.0.0.0
ollama serve
```

OLLAMA_NUM_GPU=999 forces all layers that can run on the GPU to offload, while the Level Zero variables improve device telemetry and command submission for Intel GPUs.

- In a second shell, pull and run a model, for example:

```
ollama run llama3.1:8b
```

The IPEX-LLM integration steers the underlying llama.cpp execution toward Intel’s SYCL/Level Zero path when possible.

Option B: OpenVINO backend for Ollama (higher throughput path)

- Intel’s OpenVINO integration can accelerate inference on Intel CPU/iGPU/NPU and offers a dedicated backend for Ollama via OpenVINO GenAI, which can outperform generic SYCL paths in many cases.
- High-level flow in the LXC or VM:

1) Download and initialize the OpenVINO GenAI runtime; set GODEBUG=cgocheck=0 for the Ollama executable using this backend. 2) Obtain an OpenVINO IR model (e.g., quantized DeepSeek-R1-Distill-Qwen-7B int4), then package it. 3) Write a Modelfile declaring ModelType

“OpenVINO” and InferDevice “GPU,” and create the Ollama model image.

- Practical commands (example flow shown by the OpenVINO team and contributors):

```
# 1) Prepare OpenVINO GenAI runtime (env example)
export GODEBUG=cgocheck=0
# Source the OpenVINO/GenAI setup if provided by your runtime package
# source setupvars.sh

# 2) Download an OpenVINO IR model (example uses ModelScope tools)
pip install modelscope
modelscope download --model zhaohb/DeepSeek-R1-Distill-Qwen-7B-int4-ov --local_dir ./DeepSeek-R1-Distill-Qwen-7B-int4-ov
tar -zcvf DeepSeek-R1-Distill-Qwen-7B-int4-ov.tar.gz DeepSeek-R1-Distill-Qwen-7B-int4-ov

# 3) Modelfile (OpenVINO backend)
cat > Modelfile << 'EOF'
FROM DeepSeek-R1-Distill-Qwen-7B-int4-ov.tar.gz
ModelType "OpenVINO"
InferDevice "GPU"
PARAMETER stop ""
PARAMETER stop "```
PARAMETER stop "</User|>"
PARAMETER stop "<|end_of_sentence|>"
PARAMETER stop "</□"
PARAMETER max_new_token 4096
PARAMETER stop_id 151643
PARAMETER stop_id 151647
PARAMETER repeat_penalty 1.5
PARAMETER top_p 0.95
PARAMETER top_k 50
PARAMETER temperature 0.8
EOF

# Create and run the model with the OpenVINO backend
ollama create DeepSeek-R1-Distill-Qwen-7B-int4-ov:v1 -f Modelfile
ollama run DeepSeek-R1-Distill-Qwen-7B-int4-ov:v1
```

Troubleshooting and tips

- If /dev/dri/renderD128 is missing inside the container, recheck the LXC config lines and that i915 has created the render node on the host after enabling GuC.
- If choosing a full VM instead of LXC, Linux guests are typically smoother than Windows on Alder Lake; Windows guests often hit Code 43 unless carefully configured.
- Expect meaningful gains from OpenVINO on Intel iGPU versus generic SYCL; community benchmarks have reported several-fold speedups on some 7B models, though results vary by model and kernel.
- Vulkan on Intel iGPU often underperforms versus CPU or Intel’s native stacks; prefer oneAPI/SYCL or OpenVINO backends on Intel hardware.

Why this layout

- LXC with /dev/dri mapping is the simplest, most stable way to put the iGPU to work without full PCI passthrough, and is widely used for GPU-accelerated services on Proxmox.
- Intel’s documented IPEX-LLM environment for Ollama is straightforward and stays close to upstream Ollama usage, while OpenVINO offers an alternative backend with strong hardware-specific optimizations on Intel platforms.

Ins0mniA

Backup Proxmox Script

? One-Line Download & Execute :

```
apt update && apt install -y curl && clear && curl -s
https://docs.greenhome.stream/attachments/47 | bash
```

? One-Line Download & Execute :

```
apt update && apt install -y curl && clear && curl -s
https://docs.greenhome.stream/attachments/74 | bash
```

Ins0mniA

Proxmox Upgrade Pve8 to Pve9 Script

? One-Line Download & Execute :

```
apt update && apt install -y curl && clear && curl -s
https://docs.greenhome.stream/attachments/53 | bash
```

Ins0mniA

Proxmox Kernel Manager

Script

Proxmox script to delete old and unused Kernels from the system.

? One-Line Retrieve & Execute :

```
bash <(curl -fsSL https://docs.greenhome.stream/attachments/62)
```

Done.