

Proxmox Pages

- [Konfiguration Proxys für Proxmox](#)
- [Ultimate Beginner's Guide to Proxmox GPU Passthrough](#)
- [VM Setup](#)
- [Proxmox Software Defined Networking \(SDN\)](#)
- [VLAN Setup Example](#)

Konfiguration Proxys für Proxmox

Einleitung: Diese Anleitung erläutert den Prozess zur Konfiguration von HTTP- und apt-Proxys für Ihren Proxmox-Host, um den gesamten Datenverkehr, einschließlich administrativer Schnittstellen und Paketaktualisierungen, über bestimmte Proxys zu leiten. Durch die Implementierung von HTTP- und apt-Proxys erhalten Sie die Kontrolle über den Datenverkehrsfluss und verbessern die Sicherheit und Effizienz Ihres Proxmox-Systems. **Schritt-für-Schritt-Anleitung:**

1. **Zugriff auf die Netzwerkkonfiguration:** Öffnen Sie eine SSH-Verbindung zu Ihrem Proxmox-Host und melden Sie sich als Administrator an.
2. **Bearbeiten der Umgebungsvariablen für HTTP-Proxy:** Öffnen Sie die Datei `/etc/environment`, um Proxy-Einstellungen hinzuzufügen:

```
sudo nano /etc/environment
```

Fügen Sie am Ende der Datei die folgenden Zeilen hinzu:

```
http://"www-proxy.example.com:3128"  
https_proxy="www-proxy.example.com:3128"  
ftp_proxy="www-proxy.example.com:3128"  
no_proxy="localhost,127.0.0.1,
```

Optional geht auch port 80

3. **Bearbeiten der apt-Proxy-Konfiguration:** Öffnen Sie die Datei `/etc/apt/apt.conf.d`, um apt-Proxy-Einstellungen hinzuzufügen:

```
sudo nano /etc/apt/apt.conf.d
```

Fügen Sie die folgende Zeile hinzu:

```
Acquire::http::Proxy "www-proxy.example.com:3128";
```

Optional geht auch port 80

4. **Speichern und Schließen der Dateien:** Speichern Sie die Änderungen in beiden Dateien und schließen Sie die Texteditoren.
5. **Aktualisierung der Umgebungsvariablen:** Aktualisieren Sie die Umgebungsvariablen, um die Proxy-Einstellungen anzuwenden:

```
source /etc/environment
```

6. **Testen der Konnektivität:** Überprüfen Sie, ob die Proxy-Einstellungen funktionieren, indem Sie versuchen, eine Verbindung zu einer externen Website herzustellen:

```
curl https://www.google.com
```

Stellen Sie sicher, dass die Verbindung über den HTTP-Proxy hergestellt wird.

7. Proxmox GUI Proxy

Zum eintragen von GUI Proxy

Um einen Proxy für diesen Zweck einzurichten, müssen Sie einfach auf Datacenter → Options → HTTP proxy gehen und den folgenden Proxy eintragen, dieser dient zum herunter laden von isos und lxc container:

```
http://www-proxy.example.com:3128
```

[Bildschirmfoto 2024-03-06 um 14.21.17.png](#)

Fazit: Durch die Konfiguration von HTTP- und apt-Proxys für Ihren Proxmox-Host können Sie den Datenverkehr effizient steuern und filtern, was die Sicherheit erhöht und den Internetzugriff optimiert. Durch Befolgung der oben genannten Schritte können Sie nahtlos HTTP- und apt-Proxys für Ihren Proxmox-Host konfigurieren und die damit verbundenen Vorteile nutzen.

Ultimate Beginner's Guide to Proxmox GPU Passthrough

mirror of [The Ultimate Beginner's Guide to GPU Passthrough \(Proxmox, Windows 10\)](#) by [/u/cj alas](#)

“ Welcome all, to the first installment of my **Idiot Friendly** tutorial series! I'll be guiding you through the process of configuring GPU Passthrough for your Proxmox Virtual Machine Guests. This guide is aimed at beginners to virtualization, particularly for Proxmox users. It is intended as an overall guide for passing through a GPU (or multiple GPUs) to your Virtual Machine(s). It is not intended as an all-exhaustive how-to guide; however, I will do my best to provide you with all the necessary resources and sources for the passthrough process, from start to finish. If something doesn't work properly, please check [/r/Proxmox](#), [/r/Homelab](#), [/r/VFIO](#), or [/r/linux4noobs](#) for further assistance from the community.

Before We Begin (Credits)

“ This guide wouldn't be possible without the fantastic online Proxmox community; both here on Reddit, on the official forums, as well as other individual user guides (which helped me along the way, in order to help you!). If I've missed a credit source, please let me know! Your work is appreciated.

Disclaimer: In no way, shape, or form does this guide claim to work for all instances of Proxmox/GPU configurations. Use at your own risk. I am not responsible if you blow up your server, your home, or yourself. Surgeon General Warning: do not operate this guide while under the influence of intoxicating substances. Do not let your cat operate this guide. You have been warned.

Let's Get Started (Pre-configuration Checklist)

“ It's important to make note of all your hardware/software setup before we begin the GPU passthrough. For reference, I will list what I am using for hardware and software. This guide may or may not work the same on any given hardware/software configuration, and it is intended to help give you an overall understanding and basic setup of GPU passthrough for Proxmox *only*.

Your hardware should, at the very least, support: VT-d, interrupt mapping, and UEFI BIOS.

My Hardware Configuration:

“ Motherboard: *Supermicro X9SCM-F (Rev 1.1 Board + Latest BIOS)*

CPU: LGA1150 Socket, *Xeon E3-1220 (version 2) ^**(1)***

Memory: *16GB DDR3 (ECC, Unregistered)*

GPU: *2x GTX 1050 Ti 4gb, 2x GTX 1060 6gb ^**(2)***

My Software Configuration:

“ [Latest Proxmox Build](#) (5.3 as of this writing)

Windows 10 LTSC Enterprise (Virtual Machine) ^**(3)**

Notes:

¹On most Xeon E3 CPUs, IOMMU grouping is a mess, so some extra configuration is needed. More on this later.

²It is not recommended to use multiple GPUs of the same exact brand/model type. More on this later.

³Any Windows 10 installation ISO should work, however, try to stick to the latest available ISO from Microsoft.

Configuring Proxmox

“ This guide assumes you already have at the very least, installed Proxmox on your server and are able to login to the WebGUI and have access to the server node's Shell terminal. If you need help with installing base Proxmox, I highly recommend the [official "Getting Started" guide](#) and their [official YouTube guides](#).

Step 1: Configuring the Grub

“ Assuming you are using an Intel CPU, either SSH directly into your Proxmox server, or utilizing the noVNC Shell terminal under "Node", open up the **/etc/default/grub** file. I prefer to use **nano**, but you can use whatever text editor you prefer.

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

“ Look for this line:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet"
```

“ Then change it to look like this:

For Intel CPUs:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on"
```

“ For AMD CPUs:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet amd_iommu=on"
```

“ IMPORTANT ADDITIONAL COMMANDS

You might need to add additional commands to this line, if the passthrough ends up failing. For example, if you're using a similar CPU as I am (Xeon E3-12xx series), which has horrible IOMMU grouping capabilities, and/or you are trying to passthrough a single GPU.

These additional commands essentially tell Proxmox not to utilize the GPUs present for itself, as well as helping to split each PCI device into its own IOMMU group. This is important because, if you try to use a GPU in say, IOMMU group 1, and group 1 also has your CPU grouped together for example, then your GPU passthrough will fail.

Here are my grub command line settings:

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt  
pcie_acs_override=downstream,multifunction nofb nomodeset video=vesafb:off,efifb:off"
```

“ For more information on what these commands do and how they help:

A. Disabling the Framebuffer: [video=vesafb:off,efifb:off](#)

B. ACS Override for IOMMU groups:

[pcie_acs_override=downstream,multifunction](#)

When you finished editing **/etc/default/grub** run this command:

```
update-grub
```

Step 2: VFIO Modules

“ You'll need to add a few VFIO modules to your Proxmox system. Again, using nano (or whatever), edit the file **/etc/modules**

```
nano /etc/modules
```

“ Add the following (copy/paste) to the /etc/modules file:

```
vfio
vfio_iommu_type1
vfio_pci
vfio_virqfd
```

“ Then save and exit.

Step 3: IOMMU interrupt remapping

“ I'm not going to get too much into this; all you really need to do is run the following commands in your Shell:

```
echo "options vfio_iommu_type1 allow_unsafe_interrupts=1" > /etc/modprobe.d/iommu_unsafe_interrupts.conf
echo "options kvm ignore_msrs=1" > /etc/modprobe.d/kvm.conf
```

Step 4: Blacklisting Drivers

“ We don't want the Proxmox host system utilizing our GPU(s), so we need to blacklist the drivers. Run these commands in your Shell:

```
echo "blacklist radeon" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nouveau" >> /etc/modprobe.d/blacklist.conf
echo "blacklist nvidia" >> /etc/modprobe.d/blacklist.conf
```


Step 5: Adding GPU to VFIO

“ Run this command:

```
lspci -v
```

“ Your shell window should output a bunch of stuff. Look for the line(s) that show your video card. It'll look something like this:

01:00.0 VGA compatible controller: NVIDIA Corporation GP104 [GeForce GTX 1070] (rev a1) (prog-if 00 [VGA controller])

01:00.1 Audio device: NVIDIA Corporation GP104 High Definition Audio Controller (rev a1)

Make note of the first set of numbers (e.g. **01:00.0** and **01:00.1**). We'll need them for the next step.

Run the command below. Replace **01:00** with whatever number was next to your GPU when you ran the previous command:

```
lspci -n -s 01:00
```

“ Doing this should output your GPU card's *Vendor IDs*, usually one ID for the GPU and one ID for the Audio bus. It'll look a little something like this:

01:00.0 0000: 10de:1b81 (rev a1)

01:00.1 0000: 10de:10f0 (rev a1)

What we want to keep, are these vendor id codes: **10de:1b81** and **10de:10f0**.

Now we add the GPU's vendor id's to the VFIO (***remember to replace the id's with your own!***):

```
echo "options vfio-pci ids=10de:1b81,10de:10f0 disable_vga=1"> /etc/modprobe.d/vfio.conf
```

Finally, we run this command:

```
update-initramfs -u
```

“ And restart:

```
reset
```

“ Now your Proxmox host should be ready to passthrough GPUs!

Configuring the VM (Windows 10)

“ Now comes the 'fun' part. It took me many, many different configuration attempts to get things *just right*. Hopefully my pain will be your gain, and help you get things done right, the first time around.

Step 1: Create a VM

“ Making a Virtual Machine is pretty easy and self-explanatory, but if you are having issues, I suggest looking up the official Proxmox Wiki and How-To guides.

For this guide, you'll need a Windows ISO for your Virtual Machine. Here's a handy guide on [how to download an ISO file directly into Proxmox](#). You'll want to copy ALL your .ISO files to the proper repository folder under Proxmox (including the VirtIO driver ISO file mentioned below).

Example Menu Screens

[General](#) => [OS](#) => [Hard disk](#) => [CPU](#) => [Memory](#) => [Network](#) => [Confirm](#)

IMPORTANT: DO NOT START YOUR VM (yet)

Step 1a (Optional, but RECOMMENDED): [Download VirtIO drivers](#)

“ If you follow this guide and are using VirtIO, then you'll need this ISO file of the VirtIO drivers to mount as a CD-ROM in order to install Windows 10 using VirtIO (SCSI).

For the CD-Rom, it's fine if you use IDE or SATA. Make sure CD-ROM is selected as the primary boot device under the *Options* tab, when you're done creating the VM. Also, you'll want to make sure you select VirtIO (SCSI, *not* VirtIO Block) for your Hard disk and Network Adapter.

Step 2: Enable OMVF (UEFI) for the VM

“ Under your VM's **Options** Tab/Window, set the following up like so:

Boot Order: CD-ROM, Disk (scsi0)

SCSI Controller: VirtIO SCSI Single

BIOS: OMVF (UEFI)

“ **Don't Forget:** When you change the BIOS from *SeaBIOS (Default)* to *OMVF (UEFI)*, Proxmox will say something about adding an EFI disk. So you'll go to your **Hardware** Tab/Window and do that. Add > EFI Disk.

Step 3: Edit the VM Config File

“ Going back to the Shell window, we need to edit `/etc/pve/qemu-server/.conf`, where is the VM ID Number you used during the VM creation (General Tab).

```
nano /etc/pve/qemu-server/<vmid>.conf
```

In the editor, let's add these command lines (doesn't matter where you add them, so long as they are on new lines. Proxmox will move things around for you after you save):

```
machine: q35
cpu: host,hidden=1,flags=+pcid
args: -cpu 'host,+kvm_pv_unhalt,+kvm_pv_eoi,hv_vendor_id=NV43FIX,kvm=off'
```

“ Save and exit the editor.

Step 4: Add PCI Devices (Your GPU) to VM

[Look at all those GPUs](#)

“ Under the VM's **Hardware** Tab/Window, click on the *Add* button towards the top. Then under the drop-down menu, click *PCI Device*.

[Look for your GPU in the list](#), and select it. On the PCI options screen, you should only need to [configure it like so](#):

```
All Functions: YES
Rom-Bar: YES
Primary GPU: NO
PCI-Express: YES (requires 'machine: q35' in vm config file)
```

“ [Here's an example image](#) of what your Hardware Tab/Window should look like when you're done creating the VM.

[Oopsies, make sure “All Functions” is CHECKED.](#)

Step 4a (Optional): ROM File Issues

In the off chance that things don't work properly at the end, you MIGHT need to come back to this step and specify the ROM file for your GPU. This is a process unto itself, and requires some extra steps, as outlined below.

Step 4a1:

[Download](#) your GPU's ROM file

OR

Dump your GPU's ROM File:

```
cd /sys/bus/pci/devices/0000:01:00.0/  
echo 1 > rom  
cat rom > /usr/share/kvm/<GPURomFileName>.bin  
echo 0 > rom
```

“ Alternative Methods to Dump ROM File:

a. [Using GPU-Z \(recommended\)](#)

b. [Using NVFlash](#)

Step 4a2: Copy the ROM file (if you downloaded it) to the */usr/share/kvm/* directory.

You can use SFTP for this, or directly through Windows' Command Prompt:

```
scp /path/to/<romfilename>.rom myusername@proxmoxserveraddress:/usr/share/kvm/<romfilename>.rom
```

“ Step 4a3: Add the ROM file to your VM Config (EXAMPLE):

```
hostpci0: 01:00,pcie=1,romfile=<GTX1050ti>.rom
```

“ NVIDIA USERS: If you're still experiencing issues, or the ROM file is causing issues on its own, you might need to patch the ROM file (particularly for NVIDIA

cards). There's a great tool for patching GTX 10XX series cards here: <https://github.com/sk1080/nvidia-kvm-patcher> and here <https://github.com/Matoking/NVIDIA-vBIOS-VFIO-Patcher>. It only works for 10XX series though. If you have something older, you'll have to patch the ROM file manually using a hex editor, which is beyond the scope of this tutorial guide.

[Example of the Hardware Tab/Window, Before Windows 10 Installation.](#)

Step 5: START THE VM!

“ We're almost at the home stretch! Once you start your VM, open your noVNC / Shell Tab/Window (under the VM Tab), and you should see the Windows installer booting up. Let's quickly go through the process, since it can be easy to mess things up at this junction.

Final Setup: Installing / Configuring Windows 10

Copyright(c) Jon Spraggins (<https://jonspraggins.com>)

“ If you followed the guide so far and are using VirtIO SCSI, you'll run into an issue during the Windows 10 installation, when it [tries to find your hard drive](#). Don't worry!

Step 1: VirtIO Driver Installation

“ Simply go to your VM's **Hardware** Tab/Window (again), double click the CD-ROM drive file (it should currently have the Windows 10 ISO loaded), and switch the **ISO image** to the **VirtIO ISO** file.

Tabbing back to your noVNC Shell window, click *Browse*, find your newly loaded VirtIO CD-ROM drive, and go to the ***vioscsi > w10 > amd64*** sub-directory. Click OK.

Now the Windows installer should do its thing and load the Red Hat VirtIO SCSI driver for your hard drive. Before you start installing to the drive, go back again to the VirtIO CD-Rom, and also install your Network Adapter VirtIO drivers from ***NetKVM > w10 > amd64*** sub-directory.

“ **IMPORTANT #1: Don't forget to switch back the ISO file from the VirtIO ISO image to your Windows installer ISO image under the VM Hardware > CD-Rom.**

When you're done changing the CD-ROM drive back to your Windows installer ISO, go back to your Shell window and click *Refresh*. The installer should then have your VM's hard disk appear and have windows ready to be installed. Finish your Windows installation.

IMPORTANT #2: When Windows asks you to restart, right click your VM and hit 'Stop'. Then go to your VM's Hardware Tab/Window, and Unmount the Windows ISO from your CD-Rom drive. Now 'Start' your VM again.

Step 2: Enable Windows Remote Desktop

“ If all went well, you should now be seeing your Windows 10 VM screen! It's important for us to enable some sort of remote desktop access, since we will be disabling Proxmox's noVNC / Shell access to the VM shortly. I prefer to use Windows' built-in Remote Desktop Client. [Here's a great, simple tutorial on enabling RDP access.](#)

NOTE: While you're in the Windows VM, make sure to make note of your VM's Username, internal IP address and/or computer name.

Step 3: Disabling Proxmox noVNC / Shell Access

“ To make sure everything is properly configured before we get the GPU drivers installed, we want to disable the built-in video display adapter that shows up in

the Windows VM. To do this, we simply go to the VM's **Hardware** Tab/Window, and under the *Display* entry, we select *None (none)* from the drop-down list. Easy. Now 'Stop' and then 'Start' your Virtual Machine.

NOTE: If you are not able to (re)connect to your VM via Remote Desktop (using the given internal IP address or computer name / hostname), go back to the VM's Hardware Tab/Window, and under the PCI Device Settings for your GPU, checkmark *Primary GPU*.** Save it, then 'Stop' and 'Start' your VM again.**

Step 4: Installing GPU Drivers

“ At long last, we are almost done. The final step is to get your GPU's video card drivers installed. Since I'm using NVIDIA for this tutorial, we simply go to <http://nvidia.com> and browse for our specific GPU model's driver (in this case, GTX 10XX series). While doing this, I like to check Windows' **Device Manager** (under Control Panel) to see if there are any missing VirtIO drivers, and/or if the GPU is giving me a *Code 43 Error*. You'll most likely see the Code 43 error on your GPU, which is why we are installing the drivers. If you're missing any VirtIO (usually shows up as 'PCI Device' in Device Manager, with a yellow exclamation), just go back to your VM's **Hardware** Tab/Window, repeat the steps to mount your VirtIO ISO file on the CD-Rom drive, then point the Device Manager in Windows to the CD-Rom drive when it asks you to add/update drivers for the Unknown device.

Sometimes just installing the plain NVIDIA drivers will throw an error (something about being unable to install the drivers). In this case, you'll have to install using NVIDIA's crappy **GeForce Experience(tm)** installer. It sucks because you have to create an account and all that, but your driver installation should work after that.

Congratulations!

“ After a reboot or two, you should now be able to see NVIDIA Control Panel installed in your Windows VM, as well as Device Manager showing no Code 43 Errors on your GPU(s). Pat yourself on the back, do some jumping jacks, order a cake! You've done it!

Credits / Resources / Citations

1. https://pve.proxmox.com/wiki/Pci_passthrough
2. <https://forum.proxmox.com/threads/gpu-passthrough-tutorial-reference.34303/>
3. <https://vfio.blogspot.com/2014/08/iommu-groups-inside-and-out.html>
4. <https://forum.proxmox.com/threads/nvidia-single-gpu-passthrough-with-ryzen.38798/>
5. <https://heiko-sieger.info/iommu-groups-what-you-need-to-consider/>
6. <https://heiko-sieger.info/running-windows-10-on-linux-using-kvm-with-vga-passthrough/>
7. <http://vfio.blogspot.com/2014/08/vfiovga-faq.html>
8. <https://passthroughpo.st/explaining-csm-efiboff-setting-boot-gpu-manually/>
9. <http://bart.vanhauwaert.org/hints/installing-win10-on-KVM.html>
10. <https://jonspraggins.com/the-idiot-installs-windows-10-on-proxmox/>
11. https://pve.proxmox.com/wiki/Windows_10_guest_best_practices
12. <https://docs.fedoraproject.org/en-US/quick-docs/creating-windows-virtual-machines-using-virtio-drivers/index.html>
13. https://nvidia.custhelp.com/app/answers/detail/a_id/4188/~/extracting-the-geforce-video-bios-rom-file
14. <https://www.overclock.net/forum/69-nvidia/1523391-easy-nvflash-guide-pictures-gtx-970-980-a.html>
15. <https://medium.com/@konpat/kvm-gpu-pass-through-finding-the-right-bios-for-your-nvidia-pascal-gpu-dd97084b0313>
16. <https://www.groovypost.com/howto/setup-use-remote-desktop-windows-10/>



“ Thank you everyone!

VM Setup

When setting up a Windows VM, whether it's a standard Windows 10/11 or Windows Server, you have to take a few extra steps, to enable the best possible performance for your VM.

1. Setup Steps

Follow the steps below, when creating the VM:

	<p>In the first step, simply give your VM a name and click <code>Next</code>.</p>
	<p>Next, select your Windows ISO and click <code>Next</code>.</p> <p>Also Since we are using VirtIO devices, we will need a driver CD, for Windows to load all necessary drivers later.</p> <p>Simply download the ISO here. In Proxmox under <code>Hardware</code> click on <code>Add</code> <code>CD/DVD Drive</code> and select the downloaded VirtIO ISO.</p>

Bildschirmfoto 2024-03-14 um 09.36.08.png	After that, make sure to enable `Qemu Agent`. You can leave the other settings in their default settings, or change the Graphics Card and storage controller according to your hardware setup
Bildschirmfoto 2024-03-14 um 09.37.04.png	Create a new SCSI drive (check the windows requirements for your version for the minimum required storage size) and select `Write back` as Cache, aswell as `Discard` to improve performance. If your storage consists of SSDs, you can enable `SSD emulation`.
Bildschirmfoto 2024-03-14 um 09.37.39.png	For your CPU, select the amount of cores you need and change the Type to `Host`, to significantly boost performance. I'd also recommend enabling `NUMA`
Bildschirmfoto 2024-03-14 um 09.38.02.png	Choose the amount of RAM, you need
Bildschirmfoto 2024-03-14 um 09.38.19.png	Change the network settings according to your Proxmox settings, and select `VirtIO` as Model
	Finish the initial Setup

Proxmox Software Defined Networking (SDN)

The Proxmox Software Defined Networking (SDN) configuration provides a comprehensive overview of network management within the Proxmox cluster. This configuration is accessible through the web-based user interface (UI) at the datacenter level, offering several sections for different aspects of SDN management:

SDN Overview

- **SDN:** This section provides an overview of the current active SDN state across the entire cluster. It allows users to view the status of SDN configurations and apply any pending changes to the network setup cluster-wide.

Network Zones Management

- **Zones:** In this section, users can create and manage virtually separated network zones. Zones enable the segmentation of the network into distinct areas, each with its own set of rules and configurations. This segmentation helps in organizing and securing network traffic based on specific requirements or criteria.

Virtual Network Bridges and Subnets Management

- **VNets:** Within this section, users can create virtual network bridges and manage subnets. Virtual Network Bridges facilitate the connection of virtual machines (VMs) to the network, while subnets allow for the organization and management of IP address ranges within those bridges.

Additional Services Configuration

- **Options:** This category allows users to add and manage additional services to enhance their SDN setup.

- **Controllers:** For controlling layer 3 routing in complex setups, which involves directing traffic between different networks based on IP addresses.
- **DHCP:** Users can define a DHCP server for a zone, which automatically allocates IP addresses for guests (VMs) from a predefined pool and leases them to the guests via DHCP. This simplifies the management of IP addresses within the network.
- **IPAM:** Enables external IP address management for guests, providing a centralized system for tracking and assigning IP addresses to VMs.
- **DNS:** Users can define a DNS server integration for registering virtual guests' hostname and IP addresses, ensuring that DNS resolution works seamlessly within the SDN environment.

Summary

The Proxmox SDN configuration offers a comprehensive suite of tools and features for managing network resources within the cluster. From creating and managing network zones to configuring DHCP, IPAM, and DNS services, users can efficiently set up and maintain their network infrastructure to meet their specific requirements. The web-based UI provides an intuitive interface for accessing and configuring these settings, making network management tasks more accessible and streamlined.

VLAN Setup Example

In this example, we will demonstrate how to set up a Virtual LAN (VLAN) zone to facilitate network-level isolation using VLAN tags. This setup will allow virtual machines (VMs) on different nodes to communicate through an isolated network.

Creating VLAN Zone and VNet

1. Create VLAN Zone:

- **ID:** myvlanzone
- **Bridge:** vmbr0

2. Create VNet:

- **Name:** myvnet1
- **Zone:** myvlanzone
- **Tag:** 10

Apply the configuration through the main Software-Defined Networking (SDN) panel to create VNets locally on each node.

Setting Up Virtual Machines

VM1 (node1):

- **VM Name:** vm1
- **Network Interface:** vNIC on myvnet1
- **Network Configuration:**

```
auto eth0
iface eth0 inet static
    address 10.0.3.100/24
```

VM2 (node2):

- **VM Name:** vm2

- **Network Interface:** vNIC on myvnet1
- **Network Configuration:**

```
auto eth0
iface eth0 inet static
    address 10.0.3.101/24
```

Verifying Connectivity

After configuring VM1 and VM2, you should be able to ping between them using the configured network settings. This indicates successful communication over the isolated network established by the VLAN setup.

By following these steps, you've effectively created an isolated network using VLANs, allowing VMs on different nodes to communicate securely through the specified VLAN zone and VNet.