Installing Arch Linux Manually on Proxmox VM

medium.com/@pragalva.sapkota/installing-arch-linux-manually-on-proxmox-vm-d01315c6b5df

Pragalva Sapkota October 18, 2025

What is Proxmox?

Proxmox VE is an open-source, type-1 hypervisor that runs directly on bare-metal hardware and lets you create and manage virtual machines or Linux containers through a web interface. It combines KVM for full virtualization and LXC for lightweight containers, making it ideal for homelabs and server virtualization. In this setup, Proxmox acts as the main operating system, eliminating the need for a separate host OS.

Setting Up Your Proxmox VM

Before installing Arch Linux, we need to configure the Proxmox virtual machine correctly. On my first attempt, I encountered an error related to unreadable **UEFI variables**, which happened because of incorrect BIOS and storage settings.

```
[root@archiso /]# mkdir /boot/EFI
[root@archiso /]# mount /dev/sda1 /boot/EFI
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
[root@archiso /]# grub-install --target=x86_64-efi --bootloader-id=grub_uefi --recheck
Installing for x86_64-efi platform.
EFI variables are not supported on this system.
EFI variables are not supported on this system.
grub-install: error: efibootmgr failed to register the boot entry: No such file or directory.
[root@archiso /]#
```

To avoid this, use the following configuration:

- BIOS set to OVMF (UEFI)
- Machine type: default (q35 is fine)
- Bus/Driver: change the disk bus to SATA instead of IDE or SCSI

After creating the VM, open its Options tab inside Proxmox and disable Secure Bootunder the EFI settings.
If Secure Boot stays enabled, you'll likely see this error:

Disabling Secure Boot ensures the Arch ISO can boot properly under UEFI mode.

Step-by-Step Guide

`ip addr show`

Check that your VM is connected to the internet and has a valid IP address.

`lsblk`

View all available disks and current partition info. Identify your main device name (for me it was `/dev/sda`)

Partitioning

`fdisk /dev/sda`

Inside the fdisk prompt:

g #create a new GPT partition table
n #new partition
1
<Enter> #accept default first sector
+1G #EFI partition
n #new partition
2
<Enter> #accept default first sector
+1G #Boot partition
n #new partition
3

<Enter> #accept default first sector
<Enter> #use remaining space for Root/Data partition
t #choose type
3
44 #change type of partition 3 to LUKS encryption (type L if you want to explore all type codes)
w #write changes and exit fdisk

Formatting the EFI partition:
`mkfs.fat -F32 /dev/sda1`

Formatting the Boot partition:
`mkfs.ext4 /dev/sda2`

Setting Up Encryption

Encrypting the third partition is a good idea since it will store the system's root filesystem and user data. This ensures that if the disk or VM image is ever accessed outside Proxmox, the data remains unreadable without the passphrase.

cryptsetup luksFormat /dev/sda3
cryptsetup open — type luks /dev/sda3

We encrypt only the main data partition because the EFI and Boot partitions must remain unencrypted for the system to start. The firmware and bootloader need access to those files before decryption can occur.

Setting Up LVM

LVM (Logical Volume Manager) adds flexibility to disk management by allowing dynamic resizing, snapshots, and easier partition adjustments without manual repartitioning.

pvcreate /dev/mapper/lvm #creating physical volume

vgcreate volgroup0 /dev/mapper/lvm #creating volume group
Now we crate logical volumes for the root and home directories:
lvcreate -L 15G volgroup0 -n lv_root
lvcreate -L 30G volgroup0 -n lv_home

Activating volume group so that the system can use it:
modprobe dm_mod
vgscan
vgchange -ay

Formatting and Mounting the Partitions

Next, we'll format and mount the logical volumes we created earlier. At this stage, we'll focus on **lv_root** and **lv_home**.

Format the logical volumesmkfs.ext4 /dev/volgroup0/lv_root
mkfs.ext4 /dev/volgroup0/lv_home

Mount the root volumemount /dev/volgroup0/lv_root /mnt

Mount the boot partition
Create a directory for boot files, then mount the second partition.
mkdir /mnt/boot
mount /dev/sda2 /mnt/boot

Mount the home volume Create and mount the home directory:
mkdir /mnt/home
mount /dev/volgroup0/lv_home /mnt/home

Preparing Installation Environment

Install base packagespacstrap -i /mnt base

Generate fstab
genfstab -U -p /mnt >> /mnt/etc/fstab
The fstab file defines how and where each partition is mounted at startup. Using `-U` ensures partitions are identified by UUID, making the system more stable across reboots.

Enter the chroot environment

arch-chroot /mnt #changes your root directory

Creating User Accounts

Setting password for root account:
passwd

Creating normal user with administrative (sudo) priviliges:
useradd -m -g users -G wheel pragalva #(change pragalva into username of your choice)
passwd pragalva (use your username)

Install Basic Pakcages

Now we'll install the main packages required for our Arch Linux setup, including the bootloader, desktop environment, and system tools.

pacman -S base-devel dosfstools grub efibootmgr gnome gnome-tweaks lvm2 mtools nano networkmanager openssh os-prober sudo

Installing the Kernel and Firmware

pacman -S linux linux-headers linux-lts linux-lts-headers
pacman -S linux-firmware

Installing GPU Drivers

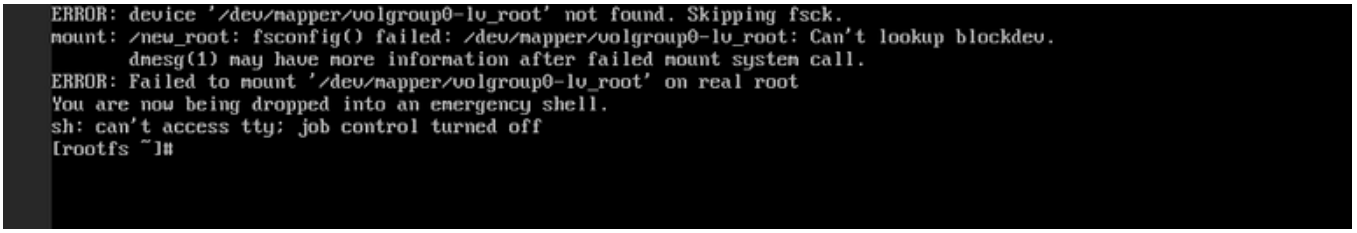
lspci #list your system's PCI devices to identify the GPU
pacman -S mesa (for intel/amd)
pacman -S nvidia nvidia-utils (for nvidia)
pacman -S nvidia-lts (for NVIDIA with LTS kernel)

Editing initramfs Configuration

initramfs (initial RAM filesystem) is a temporary root filesystem loaded into memory at boot time. It contains the essential drivers and tools the kernel needs to mount your real root partition especially important when using encryption or LVM.

nano /etc/mkinitcpio.conf In the `HOOKS` line, insert `encrypt lvm2` **between** `block` and `filesystems`.
Example: HOOKS=(base udev autodetect modconf block encrypt lvm2 filesystems keyboard fsck).

Caution: I got a boot error earlier because I didn't add encrypt and lvm2 in the HOOKS line. The system couldn't unlock the encrypted partition or detect the LVM volumes, so it dropped to an emergency shell saying it couldn't find the root device. This happened because the initramfs lacked the tools to decrypt and mount the root filesystem.



Generate Kernel Ramdisks
For Linux Package: mkinitcpio -p linux
For LTS Package : mkinitcpio -p linux-lts

Edit locale (uncomment en_US.UTF-8)
nano /etc/locale.genlocale-gen

Setting Up GRUB

Edit the GRUB configuration file:
vim /etc/default/grub
Find line starting with {GRUB_CMDLINE_LINUX_DEFAULT=}
and add your encrypted volume info to the end of it
cryptdevice=/dev/sda3:volgroup0

Create and mount the EFI directory:mkdir /boot/EFI
mount /dev/sda1 /boot/EFI

Install GRUB
grub-install — target=x86_64-efi — bootloader-id=grub_uefi — recheck
cp /usr/share/locale/en\@quot/LC_MESSAGES/grub.mo /boot/grub/locale/en.mo

Enable required services for the next boot
systemctl enable gdm
systemctl enable NetworkManager

Exit our chroot environment:

exit

unmount all partitions:
umount -a

Rebot the system:
reboot

After the reboot, the system will ask for the LUKS encryption key you created earlier for /dev/sda3. Entering this passphrase unlocks the encrypted volume so the system can mount your root filesystem and continue booting.

Wrapping Up

This completes the manual Arch Linux installation on my Proxmox server.
It serves as a foundation for experimenting with secure, modular, and customizable Linux environments for future projects and automation setups.