

# How To Run macOS on Proxmox VE

By [Klinsmann Öteyo](#) - May 16, 2023

The concept of virtualization is now everywhere with popular tools such as Docker, Virtualbox, Hyper-V, and Vmware playing a huge part in its popularity. Virtualization can be defined as the process of creating a virtual version of something, such as a computer hardware platform, operating system, storage device, or network resources. The created virtual sessions can then be used to run multiple operating systems or applications on a single physical machine, allowing greater flexibility and efficiency in resource utilization.

One of the recent virtualization tools to be introduced is **Proxmox**. This is free and open-source software, licensed under the GNU Affero General Public License. However, there is also a paid enterprise version called **Proxmox VE Subscription**, that offers additional features and professional support.

Today, we will learn how to run macOS on Proxmox VE. This is made possible via the KVM-OpenCore which is a fork of Leoyzen's OpenCore image for QEMU/KVM, which has been extended to add a build system for automatically building all of the required files from source code and to keep up with the latest OpenCore changes. This has been tested to boot macOS Catalina, Big Sur, and Monterey, but will likely also boot older versions of macOS. Although the images provided are for QEMU/KVM distributions, they can still be built to work for Promox.

Now let's dive in!

## Prerequisites

For this guide, you need to have Proxmox 7 installed. This can be done by following the below guides:

- [How To Install Proxmox VE 7 on Debian 11 \(Bullseye\)](#)
- [Install Proxmox VE 7 on Hetzner root server](#)
- [How To Install Proxmox VE 7 on OVH Dedicated Server](#)

The Proxmox host should have:

- CPU with support SSE 4.2
- First CPU generation bearing the "Core" i5/i7 branding

## Step 1 – Create macOS Installation ISO

We will build an ISO for installation from the OSX-KVM repository. First, download the copy onto your machine:

```
git clone https://github.com/thenickdude/OSX-KVM
```

If you are building the ISO on a Linux host, you need to install the below packages:

```
##On Debian/Ubuntu
sudo apt update
sudo apt install qemu-utils make
```

Now navigate into the directory:

```
cd OSX-KVM/scripts/monterey
```

Now create the image:

```
##For Minimal_Image
make Monterey-recovery.img

##For Full_Image(on MacOS only)
make Monterey-full.img
```

The above command will download the Monterey installer from Apple's software distribution servers and create the **Monterey-recovery.img** file. This image will be used later to run the installation on Proxmox. With the **minimal image**, you need internet during the installation to download any required files. For the full image, no Internet is required.

Upload the image file to the Proxmox ISO store, normally at **/var/lib/vz/template/iso**. You can use the command below:

```
sudo cp Monterey-recovery.img /var/lib/vz/template/iso/
```

## Step 2 – Prepare an OpenCore image

For this step, you need to download the OpenCore image. Although it has a **.iso** extension, this is a hard disk image.

Download the file from the [GitHub release](#) page. You can also use the commands below:

```
##Export the Version
URL=$( curl -s https://api.github.com/repos/thenickdude/KVM-
Opencore/releases/latest|grep browser_download_url|cut -d '"' -
f 4|grep .iso.gz)

##Pull the file
wget $URL
```

Now extract the file:

```
gunzip -v OpenCore-*.iso.gz
```

Now upload this ISO to the Proxmox ISO store, normally at **/var/lib/vz/template/iso**.

```
sudo cp OpenCore-*.iso /var/lib/vz/template/iso
```

## Step 3 – Fetch the OSK authentication key

During the installation, macOS checks that it is running on real Mac hardware, **it refuses to boot** if it's running on third-party hardware. You can find a way around this by using obtaining an authentication key out of your real Mac hardware.

You can obtain the OSK key on your **Mac** machine using the below steps:

Create the below file:

```
$ vim smc_read.c
/*
 * smc_read.c: Written for Mac OS X 10.5. Compile as follows:
 *
 * gcc -Wall -o smc_read smc_read.c -framework IOKit
 */

#include <stdio.h>
#include <IOKit/IOKitLib.h>

typedef struct {
    uint32_t key;
    uint8_t __d0[22];
    uint32_t datasize;
    uint8_t __d1[10];
    uint8_t cmd;
```

```
    uint32_t __d2;
    uint8_t  data[32];
} AppleSMCBuffer_t;

int
main(void)
{
    io_service_t service =
IOServiceGetMatchingService(kIOMasterPortDefault,
                            IOServiceMatching("AppleSMC"));

    if (!service)
        return -1;

    io_connect_t port = (io_connect_t)0;
    kern_return_t kr = IOServiceOpen(service, mach_task_self(),
0, &port);
    IOObjectRelease(service);
    if (kr != kIOReturnSuccess)
        return kr;

    AppleSMCBuffer_t inputStruct = { 'OSK0', {0}, 32, {0}, 5,
}, outputStruct;
    size_t outputStructCnt = sizeof(outputStruct);

    kr = IOConnectCallStructMethod((mach_port_t)port,
(uint32_t)2,
                                (const void*)&inputStruct, sizeof(inputStruct),
                                (void*)&outputStruct, &outputStructCnt);
    if (kr != kIOReturnSuccess)
        return kr;

    int i = 0;
    for (i = 0; i < 32; i++)
        printf("%c", outputStruct.data[i]);

    inputStruct.key = 'OSK1';
    kr = IOConnectCallStructMethod((mach_port_t)port,
(uint32_t)2,
                                (const void*)&inputStruct, sizeof(inputStruct),
                                (void*)&outputStruct, &outputStructCnt);
    if (kr == kIOReturnSuccess)
        for (i = 0; i < 32; i++)
```

```
printf("%c", outputStruct.data[i]);

printf("\n");

return IOServiceClose(port);
}
```

Now in the directory, run the below command:

```
xcode-select --install # If you don't already have gcc
gcc -o smc_read smc_read.c -framework IOKit
./smc_read
```

The above commands will output a 64-character string is your OSK. Take note of it.

## Step 4 – Create the macOS VM on Proxmox

Now we will create the macOS virtual machine from the Proxmox web UI as shown. Provide the name of the VM. Take note of the macOS **VM ID** as it will be used later.

Create: Virtual Machine

General OS System Disks CPU Memory Network Confirm

Node: pve01 Resource Pool: (empty)

VM ID: 100

Name: macOS

Start at boot:

Start/Shutdown order: any

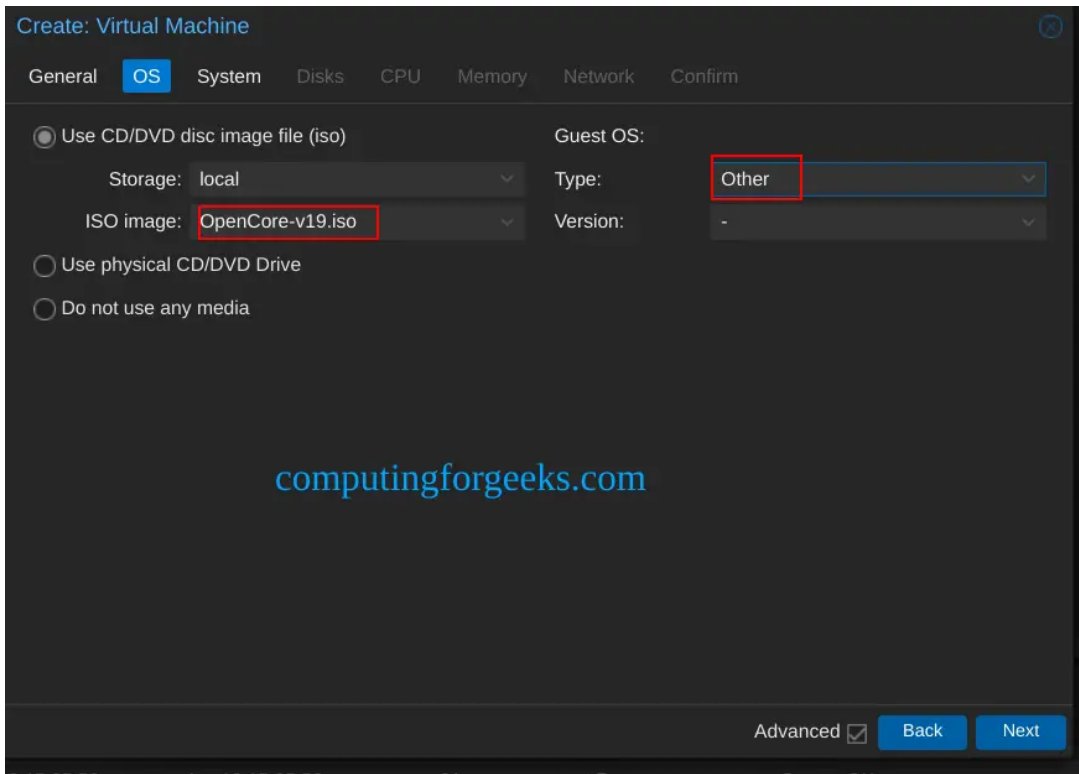
Startup delay: default

Shutdown timeout: default

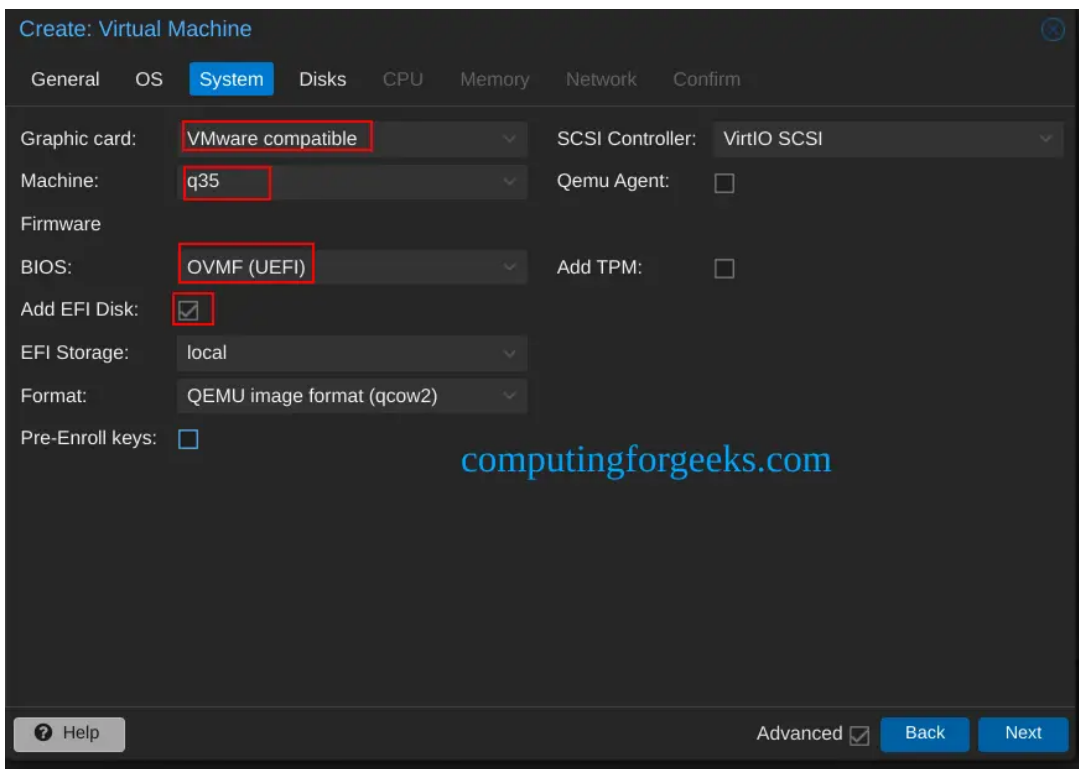
computingforgeeks.com

Help Advanced  Back Next

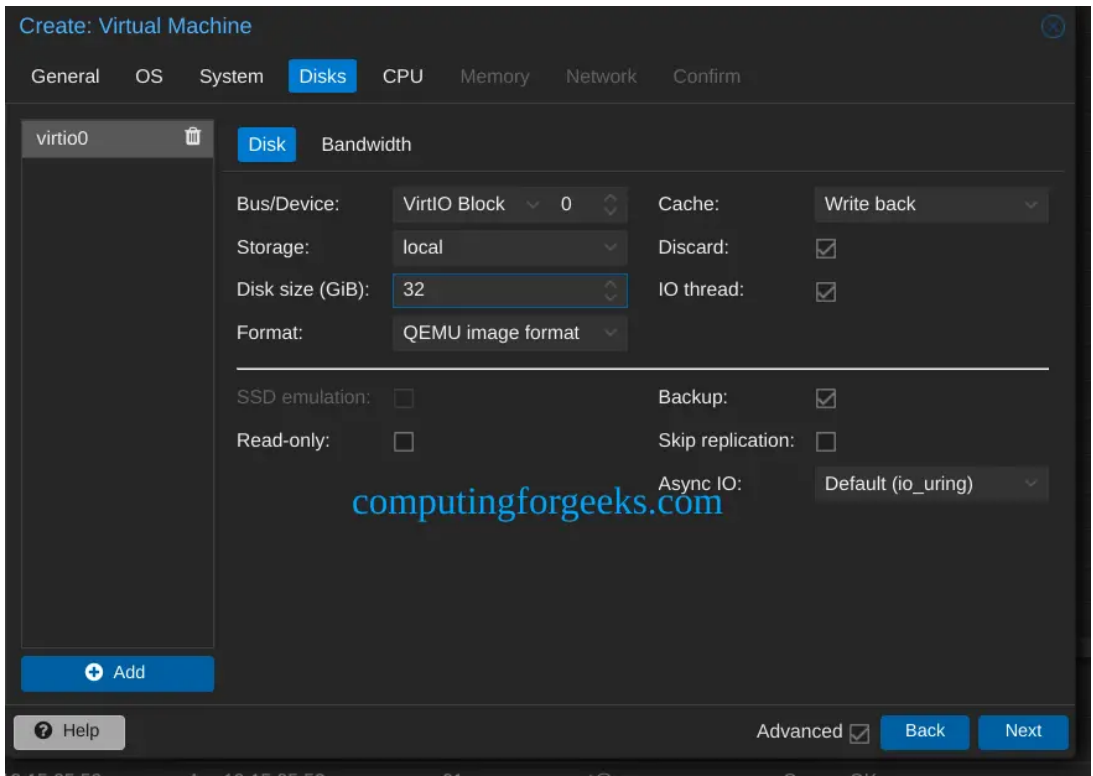
For the ISO, select the **OpenCore ISO** file.



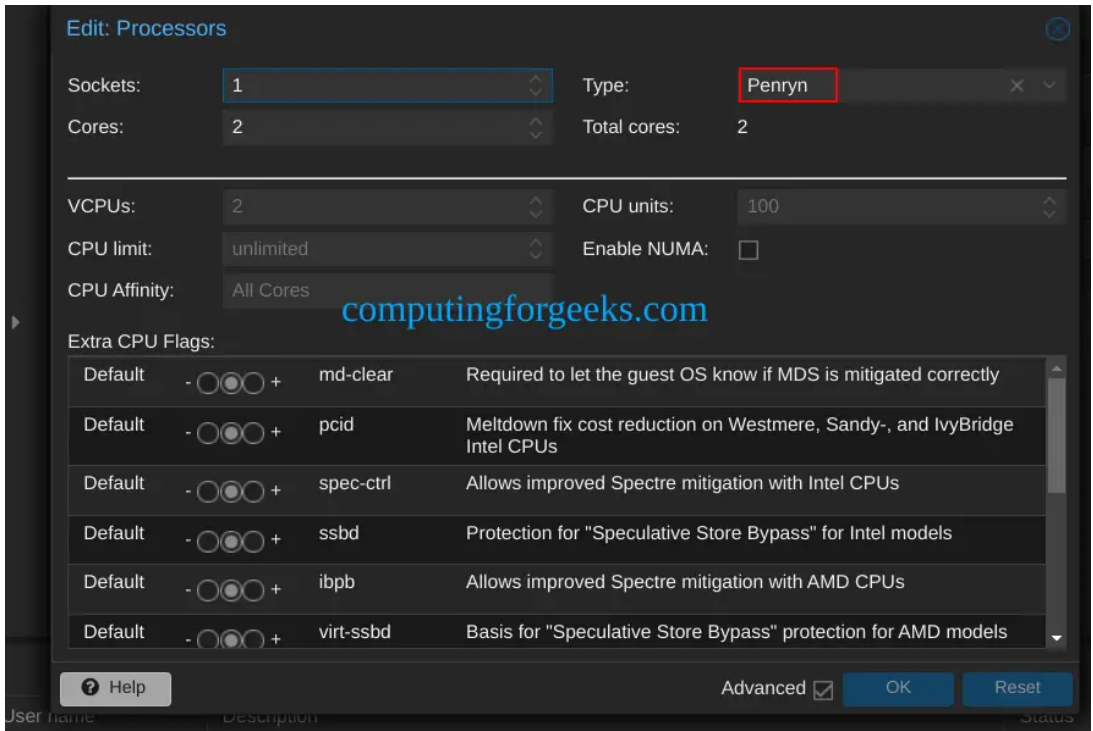
Set the system as shown. ensure that you set the graphics to VMware compatible, set the machine to **q35** and select the **Qemu agent** and add an **EFI disk** and set storage for it



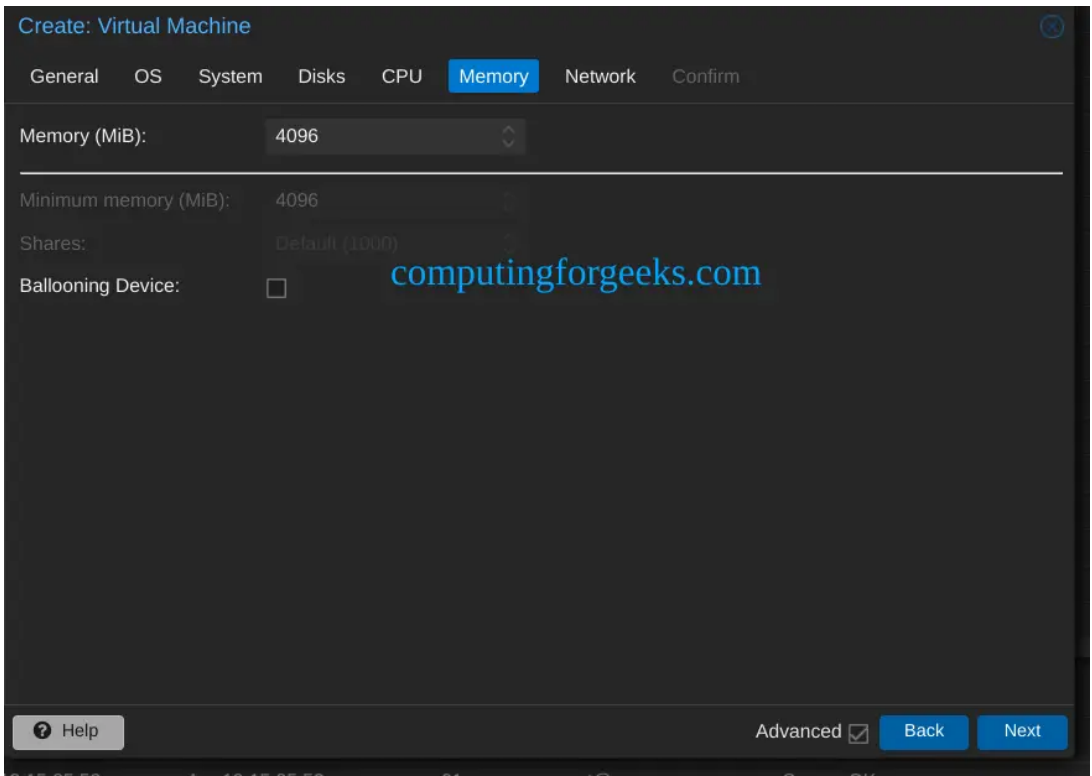
Configure the hard disk. The hard disk should be greater than **32 GB**



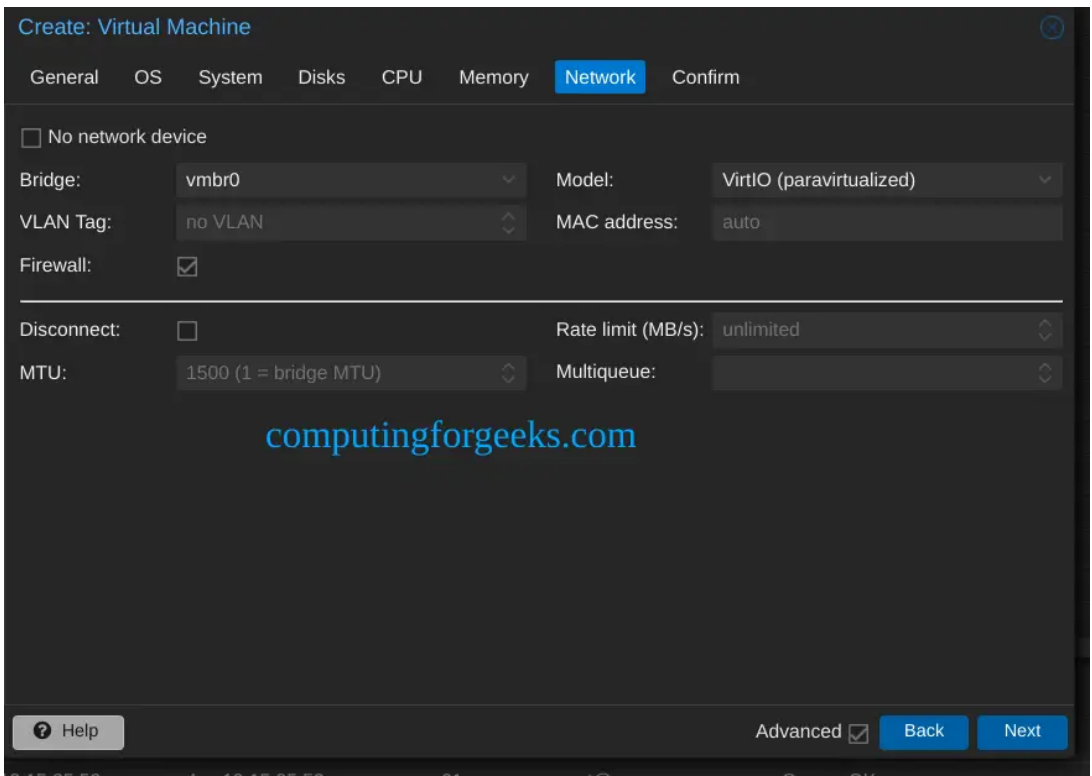
Set the CPU for the VM. Set the type as **Penryn**



Set the memory:

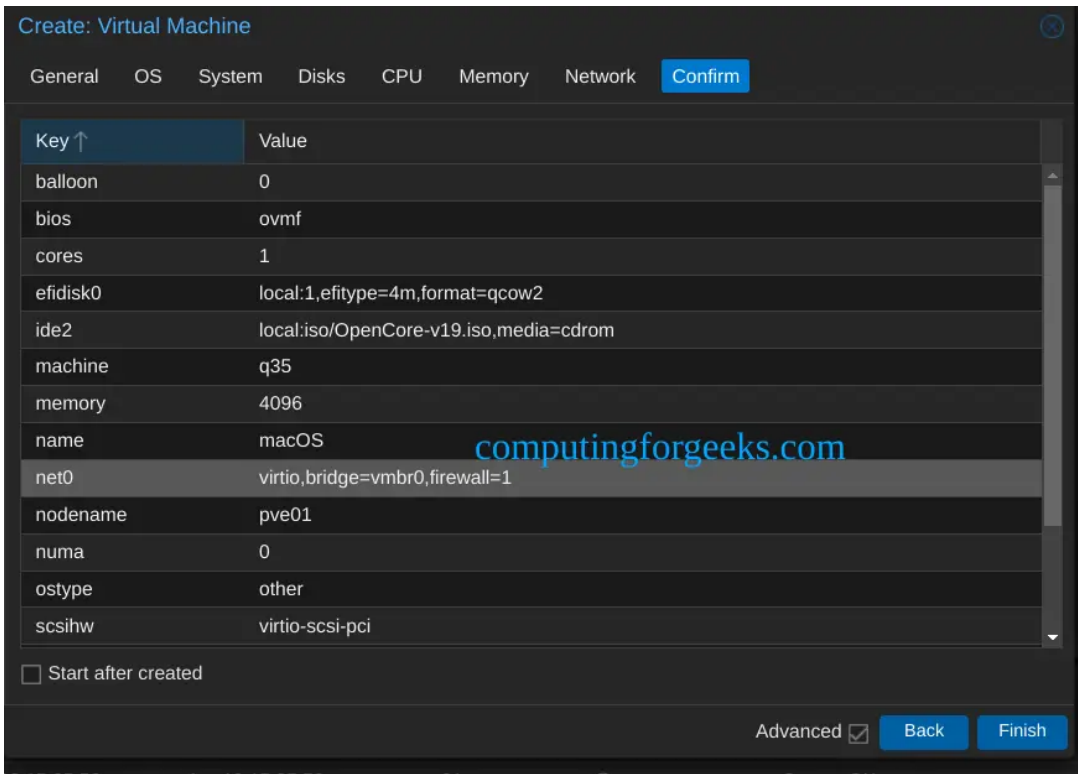


Make the network configurations. Set the model as **VirtIO(paravirtualized)**

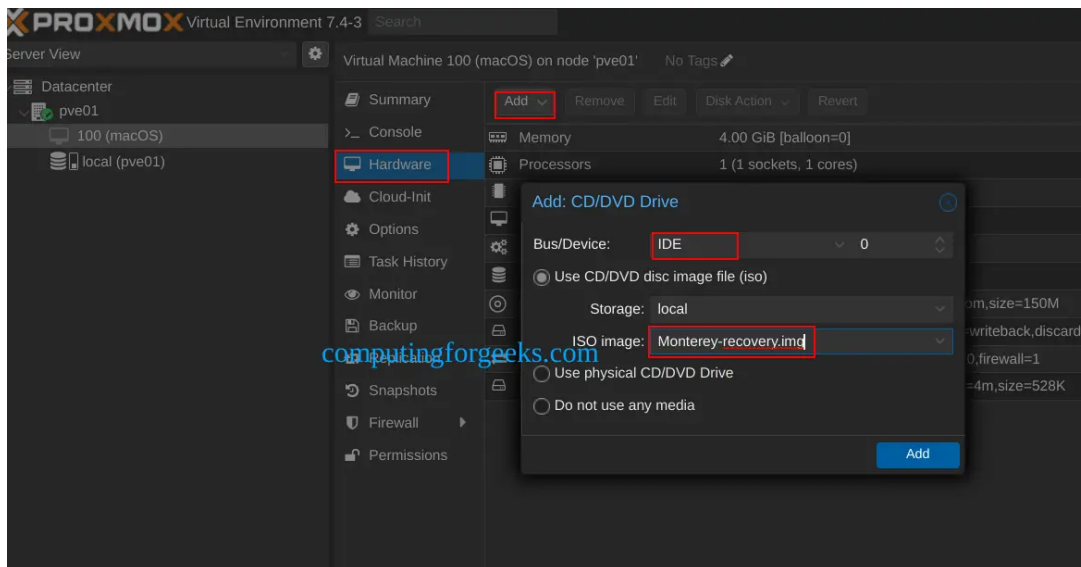


Now confirm the changes. **Do not start** the VM yet, we need to make configurations first.





Navigate to the Hardware tab and add a second DVD drive at **IDE0**. Here, select the **Monterey-full.img** or **Monterey-recovery.img** created earlier.



Then access your Proxmox host and make the below configs:

```
sudo vim /etc/pve/qemu-server/YOUR-VM-ID-HERE.conf
```

In the file, make the below changes and provide your OSK created earlier:

```
args: -device isa-applesmc,osk="THE-OSK-YOU-EXTRACTED-GOES-HERE" -smbios type=2 -device usb-kbd,bus=ehci.0,port=2 -global nec-usb-xhci.msi=off -global ICH9-LPC.acpi-pci-hotplug-with-bridge-support=off
```

We have added a USB keyboard since macOS doesn't support QEMU's default PS/2 keyboard. Remember, if you fail to provide the OSK key, the system will fail to boot.

We also need to add the below CPU argument in the **above args area**:

```
##For Intel
-cpu
host,kvm=on,vendor=GenuineIntel,+kvm_pv_unhalt,+kvm_pv_eoi,+hypervisor,+invts

##For AMD
-cpu
Penryn,kvm=on,vendor=GenuineIntel,+kvm_pv_unhalt,+kvm_pv_eoi,+hypervisor,+invts,+pcid,+ssse3,+sse4.2,+popcnt,+avx,+avx2,+aes,+fma,+fma4,+bmi1,+bmi2,+xsaves,+xsavesopt,+rdrand,check
```

This config fools the system into believing that the CPU is Penryn. This will make the MacOS VM happy even if the host CPU is AMD. You can remove the "+invts" feature from the list if your CPU doesn't support it.

Finally, find the lines that define "ISOs" (ide0 and ide2) and remove the ",media=cdrom" part and replace it with ",**cache=unsafe**". This will treat the two ISOs as hard disks and not DVDs.

```
ide0: isos:iso/Monterey-full.img,cache=unsafe,size=14G
ide2: isos:iso/OpenCore-v15.img,cache=unsafe,size=150M
```

Now you will have a config appear as below:

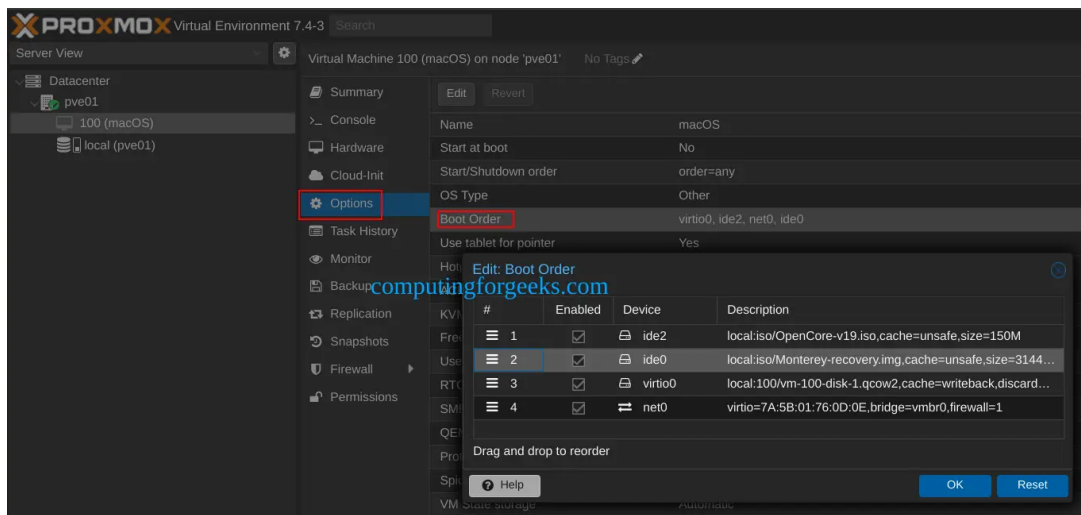
```
args: -device isa-applesmc,osk="ourhardworkbythesewordsguardedpleasedontsteal(c)AppleComputerInc" -smbios type=2 -device usb-kbd,bus=ehci.0,port=2 -global nec-usb-xhci.msi=off -global ICH9-LPC.acpi-pci-hotplug-with-bridge-support=off -cpu host,kvm=on,vendor=GenuineIntel,+kvm_pv_unhalt,+kvm_pv_eoi,+hypervisor,+invts
balloon: 0
bios: ovmf
boot: order=ide2;ide0;net0
cores: 2
cpu: Penryn
efidisk0: local:100/base-100-disk-0.qcow2,efitype=4m,size=528K
```

```

ide0: storage1:vm-100-disk-1,cache=unsafe,size=3G
ide2: storage1:vm-100-disk-2,cache=unsafe,size=152M
machine: q35
memory: 6096
meta: creation-qemu=7.2.0,ctime=1681399367
name: macOS
net0: virtio=7A:5B:01:76:0D:0E,bridge=vbr0,firewall=1
numa: 0
ostype: other
scsihw: virtio-scsi-pci
smbios1: uuid=c3cf5b1c-2dd3-46cd-ad7b-4b8f9566056c
sockets: 1
template: 1
unused0: local:100/base-100-disk-1.qcow2
unused1: local:iso/Monterey-recovery.img
unused2: local:iso/OpenCore-v19.iso
vga: vmware
virtio0: storage1:vm-100-disk-0,iothread=1,size=37G
vmgenid: 9b5673d9-0bb6-48ef-b24f-4c9771c65577

```

Now on the **options** tab, put IDE2 first on the boot order.



Now run the below command to avoid a boot loop during macOS boot and make the changes persist below reboots:

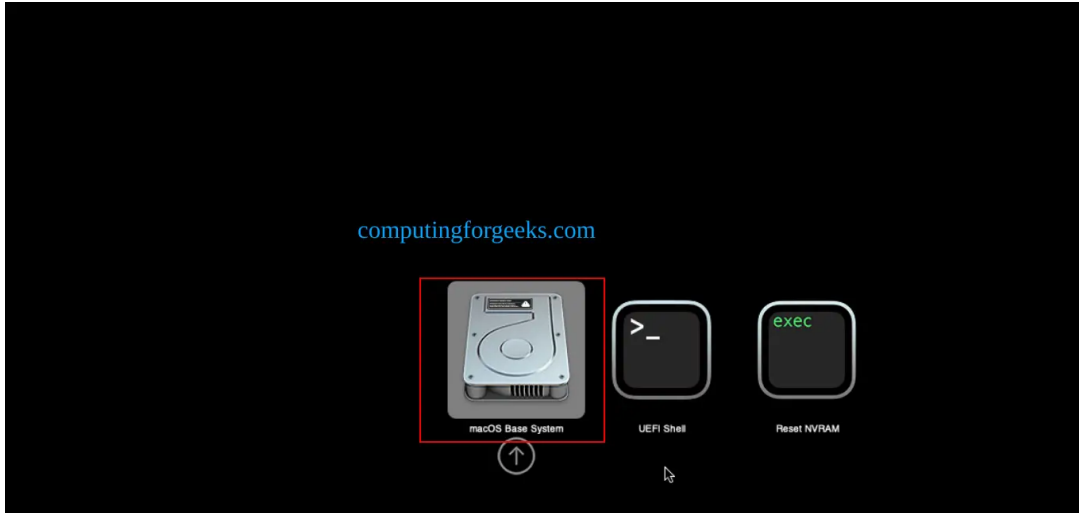
```

echo 1 > /sys/module/kvm/parameters/ignore_msrs"
echo "options kvm ignore_msrs=Y" >> /etc/modprobe.d/kvm.conf &&
update-initramfs -k all -u

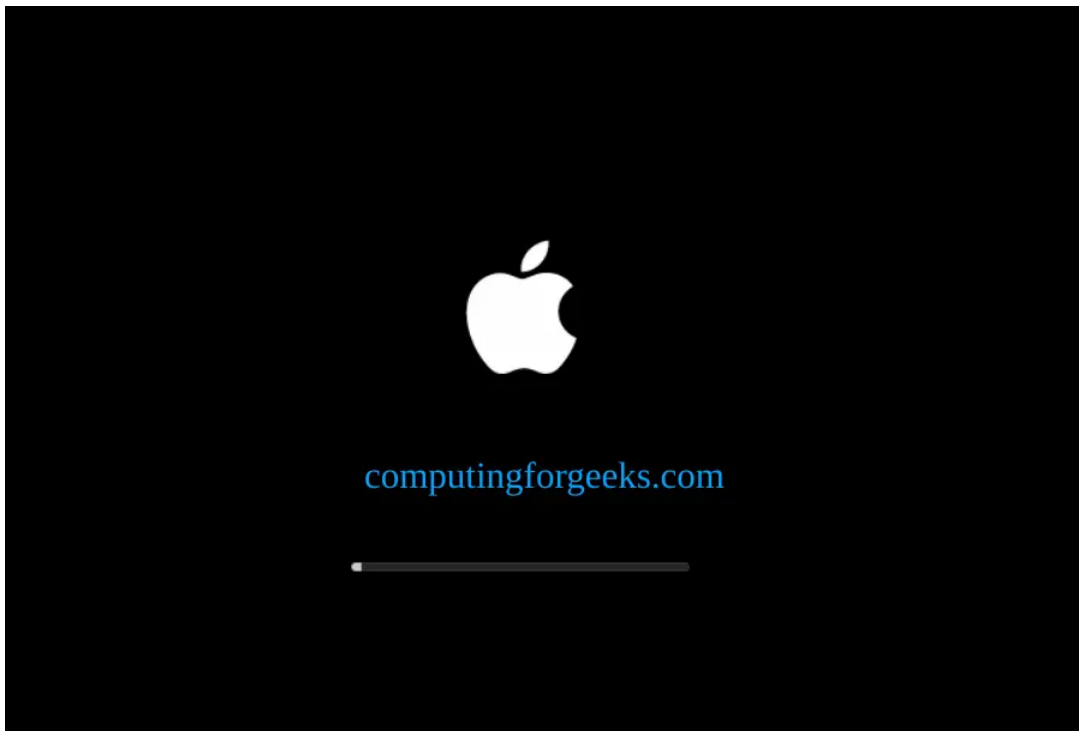
```

## Step 5 – Install macOS on Proxmox

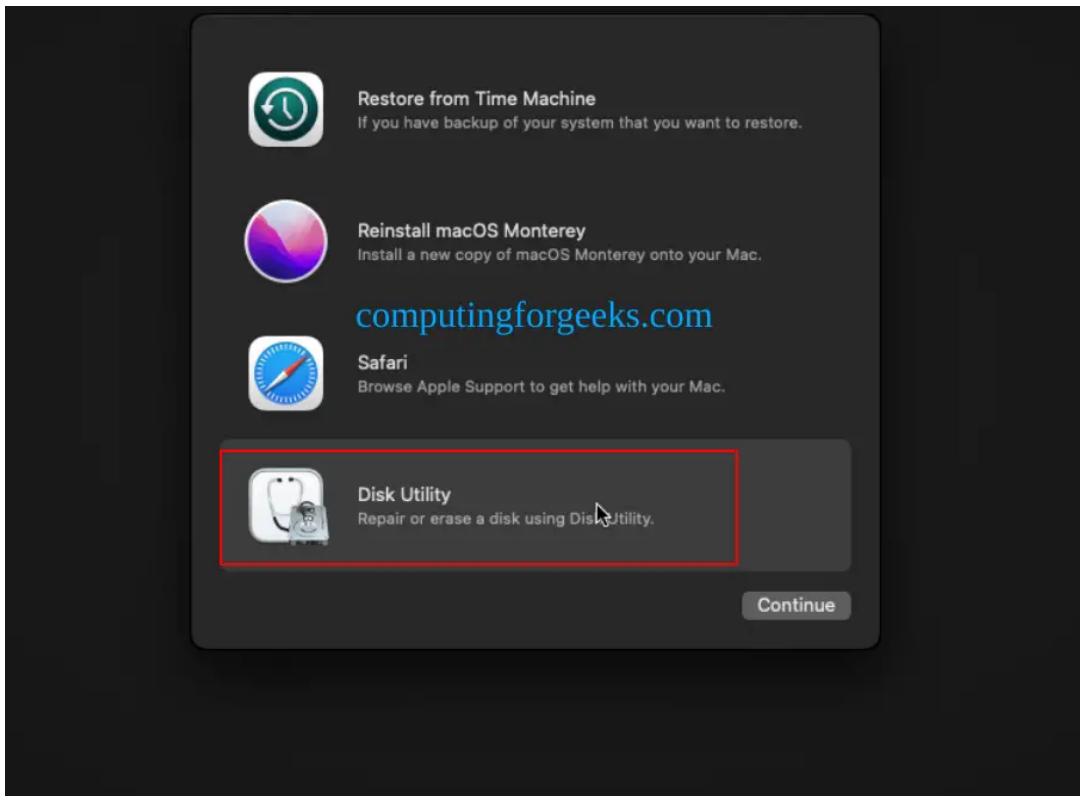
Now start the VM and it should boot into the OpenCore boot picker. Make the below selection **“MacOS Base System”**



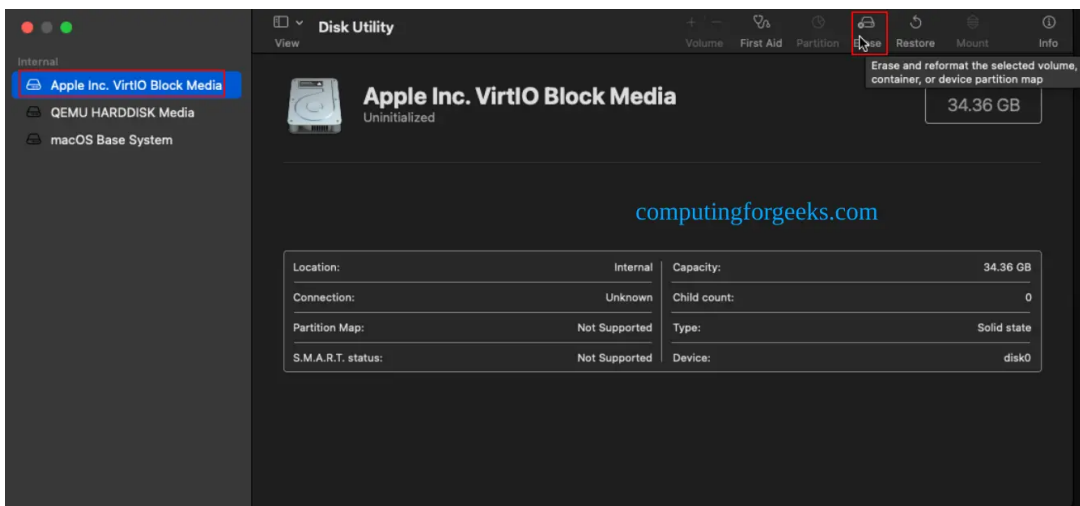
For those who built a full image, you will see **“Install macOS Monterey”** as the label. The boot will load:



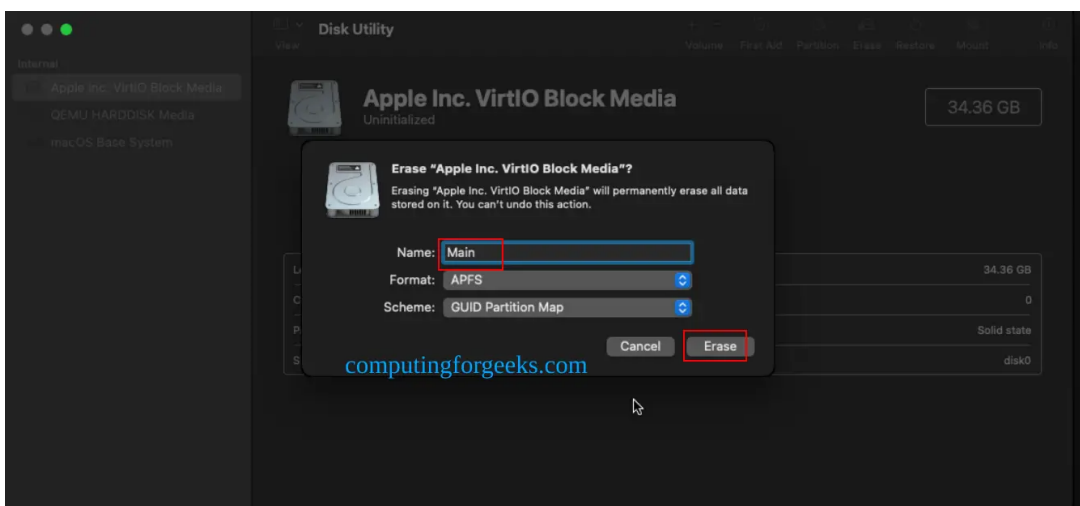
Proceed to the disk utility window:



While here, locate your hard disk(*named Apple Inc. VirtIO*) and erase it:



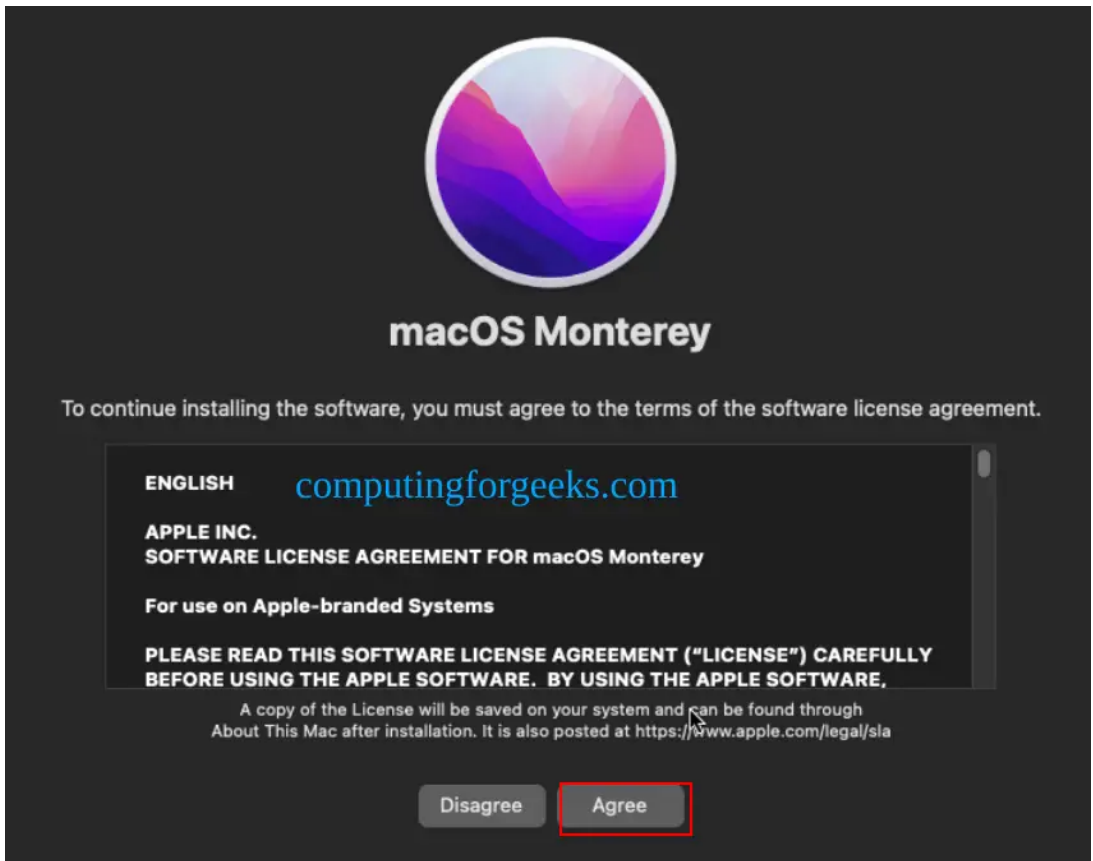
Provide a name for it. For this guide, we will call it **Main**



Once the erasing is complete, close the disk utility window and select **Reinstall macOS**



Agree to the License terms:



Select the disk on which you want to make the installation:



Sit back and wait for the installation to happen. Remember, this can take sometime:





During this stage, your system will reboot severally asking you to manually select the "macOS Installer" option:



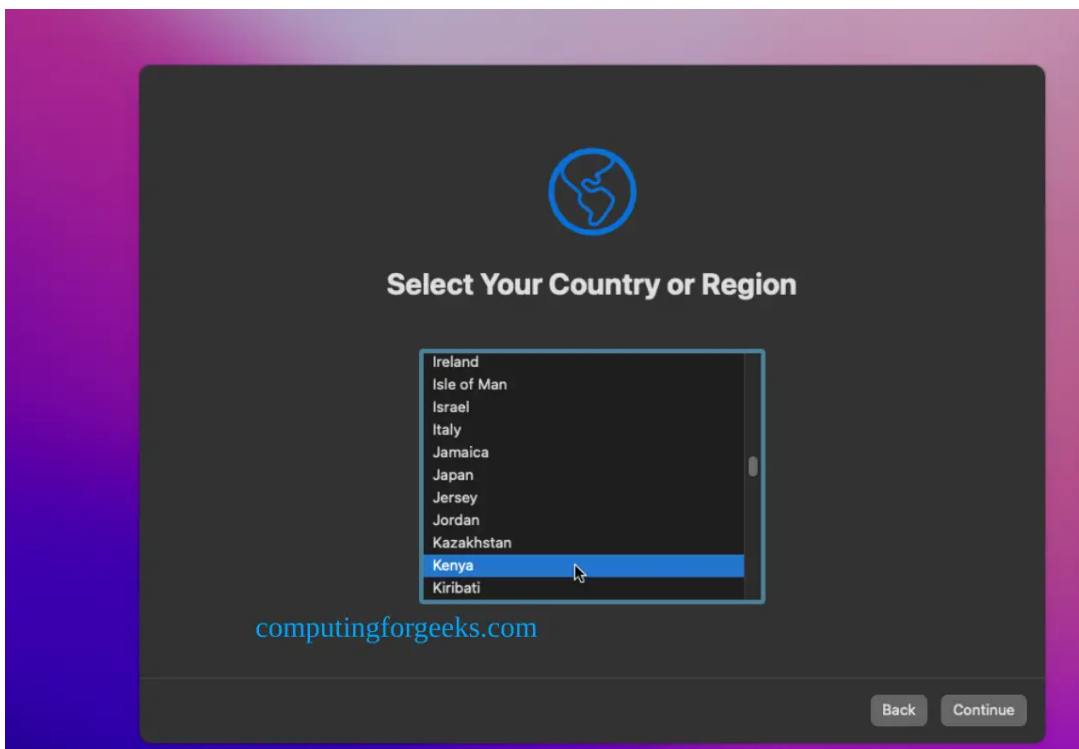
Once complete, you will see your disk appear here( ours is labelled **Main**)



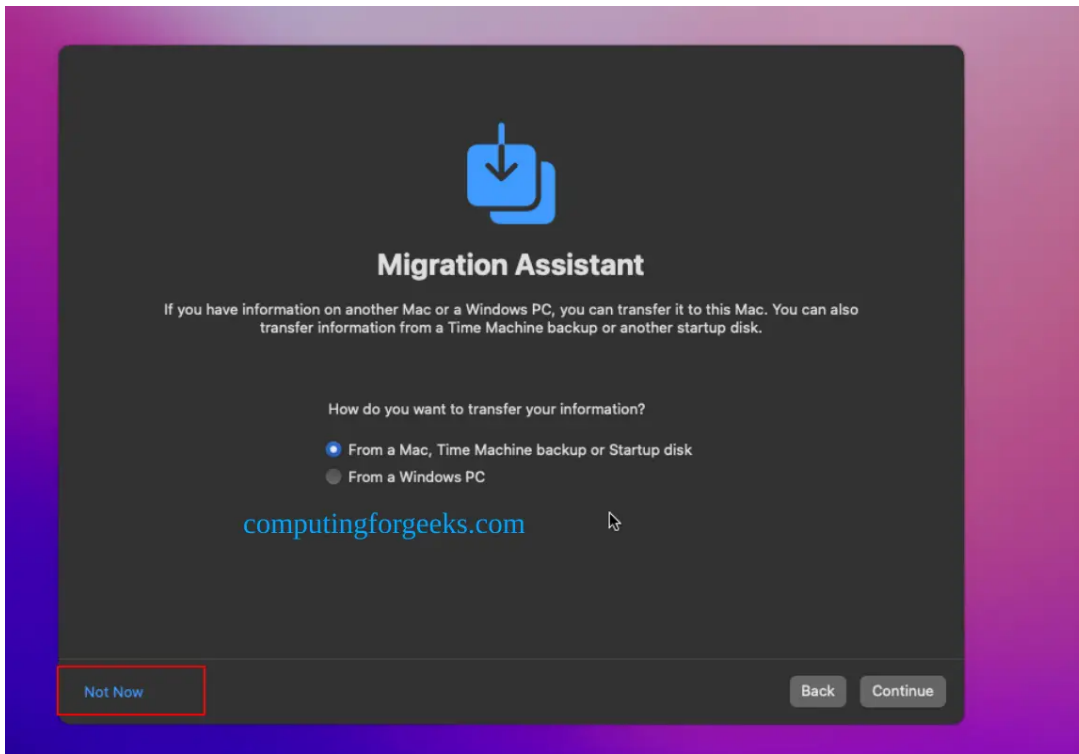


This shows that the installation is complete and we should be able to boot into our hard disk. Proceed with the initial configurations.

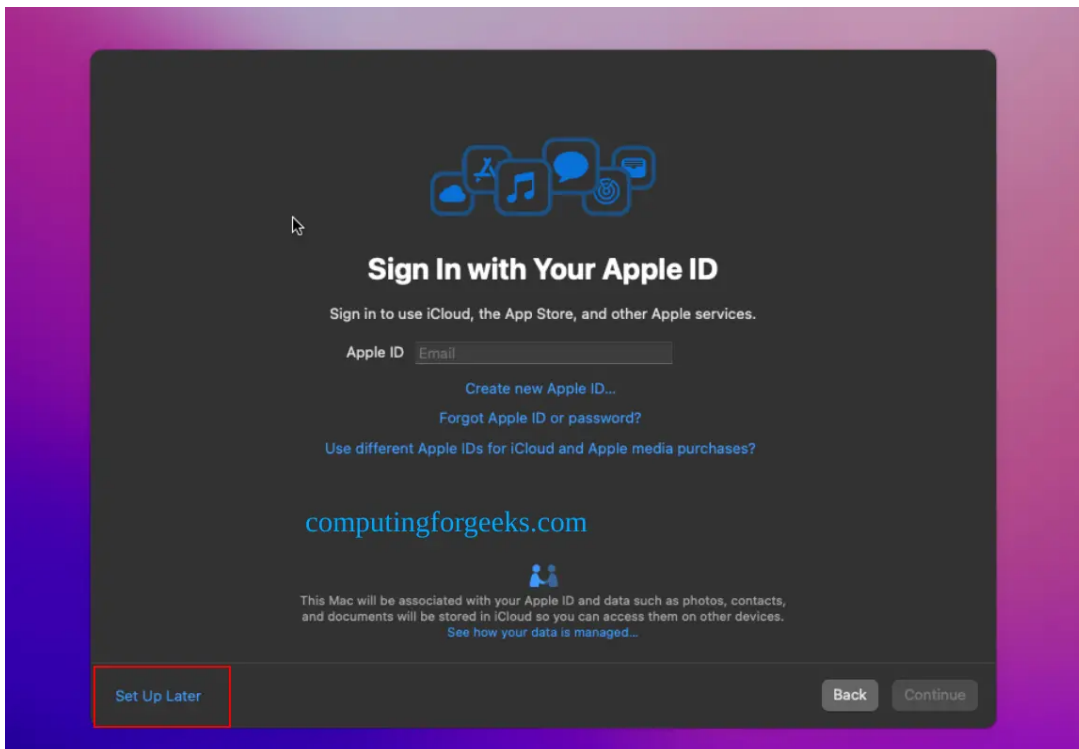
Select your country:



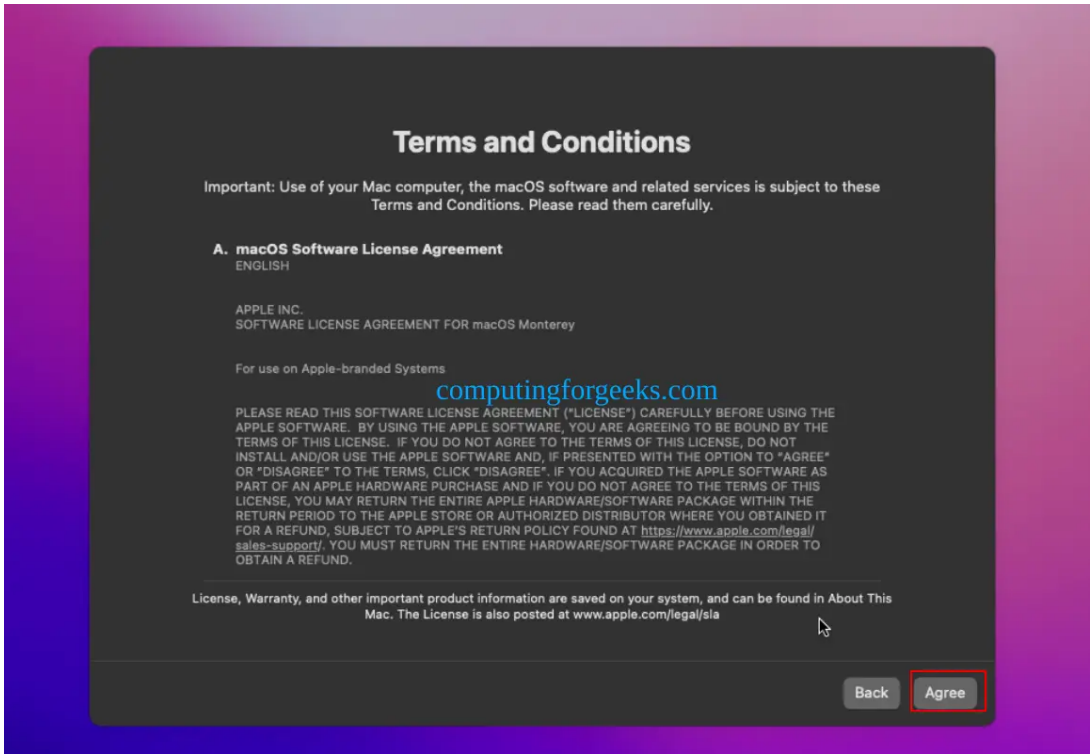
Configure your migration assistant:



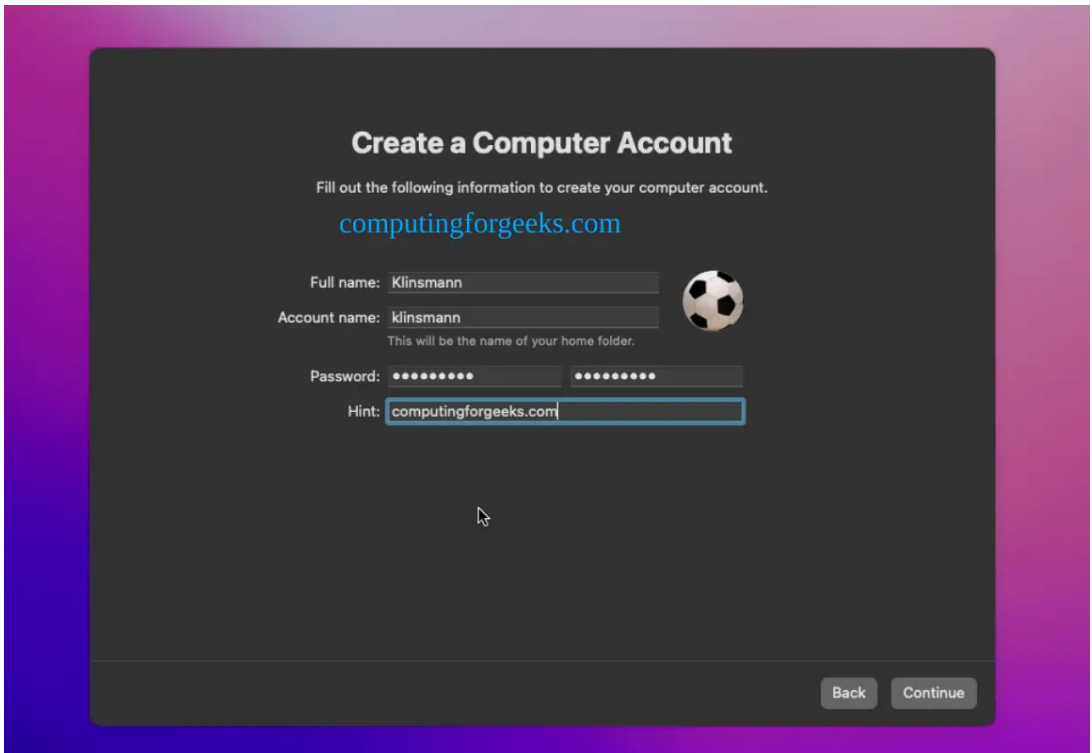
**Skip** this step of signing into your Apple ID until you've configured your Mac's serial number in OpenCore. Otherwise, a Mac device with the default shared serial numbering the OpenCore image will be added to your Apple ID.



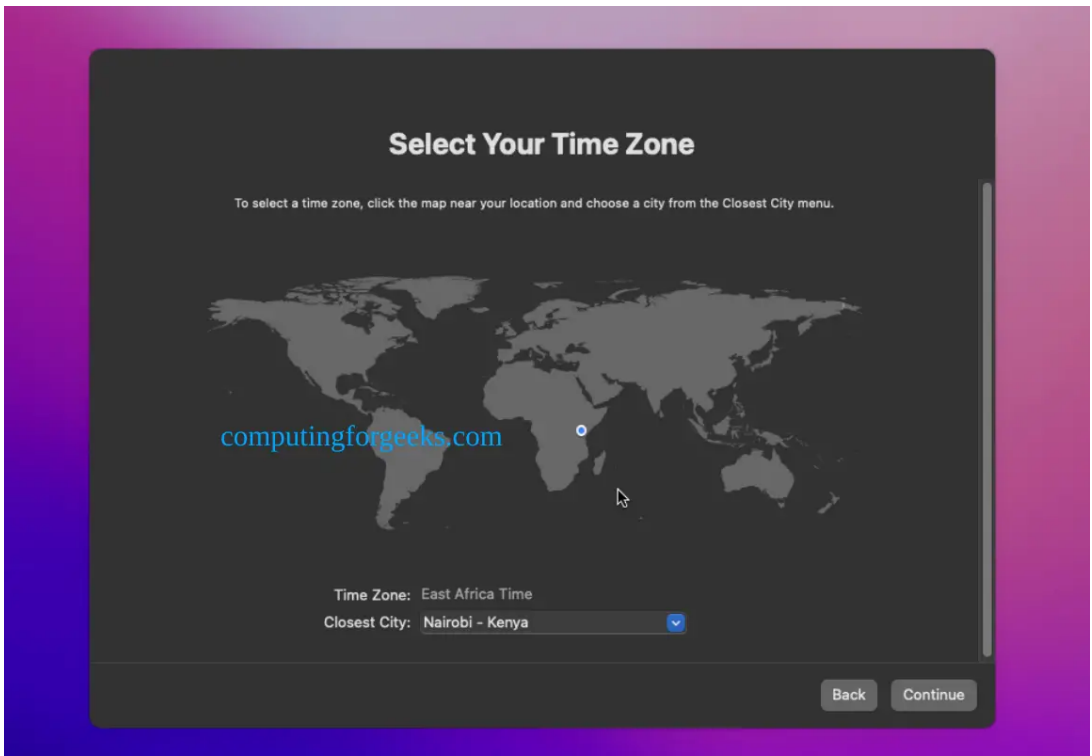
Agree to the license terms:



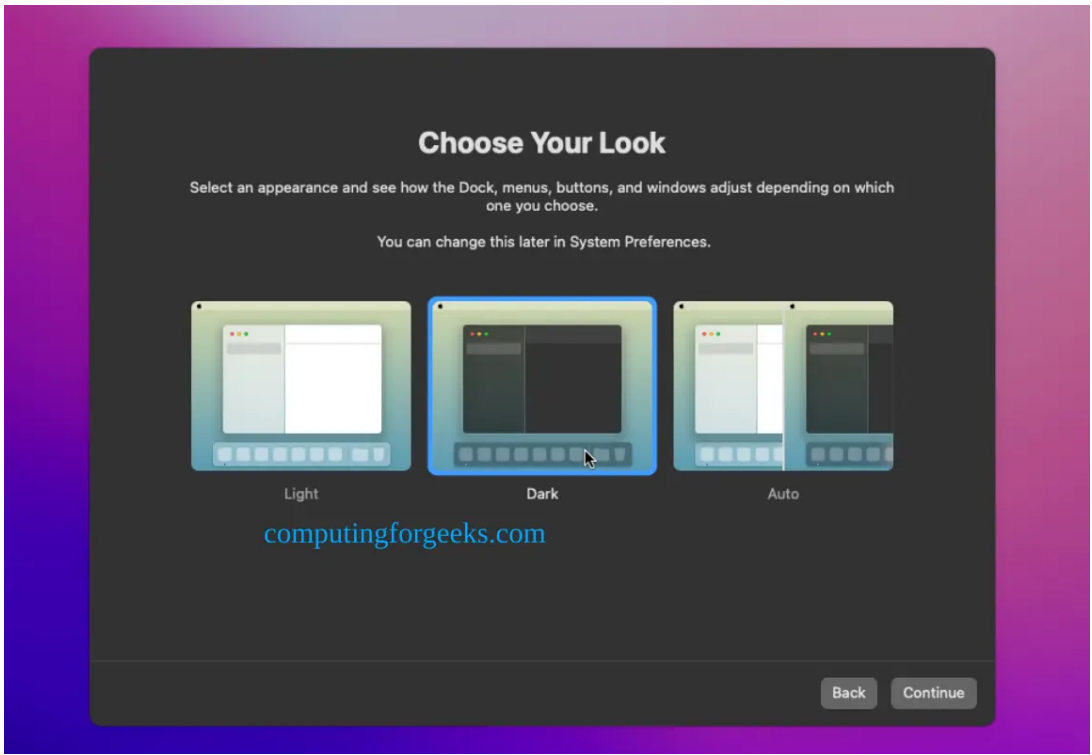
Create a system user:



Set your TimeZone:



Configure your Theme:



Voila! You now have the Mac desktop launched!



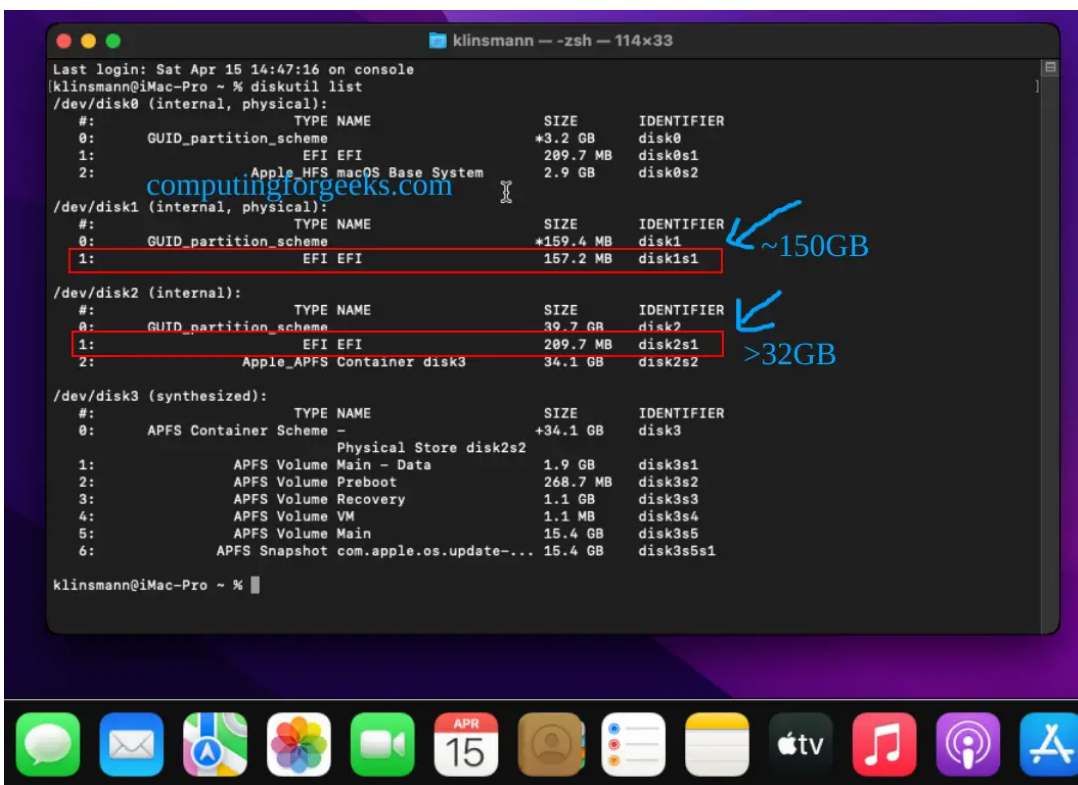
## Step 6 – Make the MacOS Installation Persistent

We have seen the MacOS desktop but that does not mean that we are done. You need to make the installation permanent. This involves copying the contents of the EFI partition on OpenCore to the hard disk.

First, launch the terminal on this Mac installation and view the available partitions using the command:

```
diskutil list
```

Sample Output:



Now we will copy the EFI partition from OpenCore to the hard disk using the command with the below syntax:

```
sudo dd if=<source> of=<dest>
```

The OpenCore EFI partition exists on the small disk(**approximately 150MB**), and the main hard disk is the one with the large(**greater than 30GB**) **Apple\_APFS** "Container" partition on it.

In this case, the EFI partitions are called **disk1s1** a and **disk2s1** respectively, this **may not be similar** to yours. Now that makes our command to be:

```
sudo dd if=/dev/disk1s1 of=/dev/disk2s1
```

**Remember to be very careful at this stage because if you get the names wrong, you can overwrite the wrong disk and you'll have to start the installation over again!**

Sample Output:

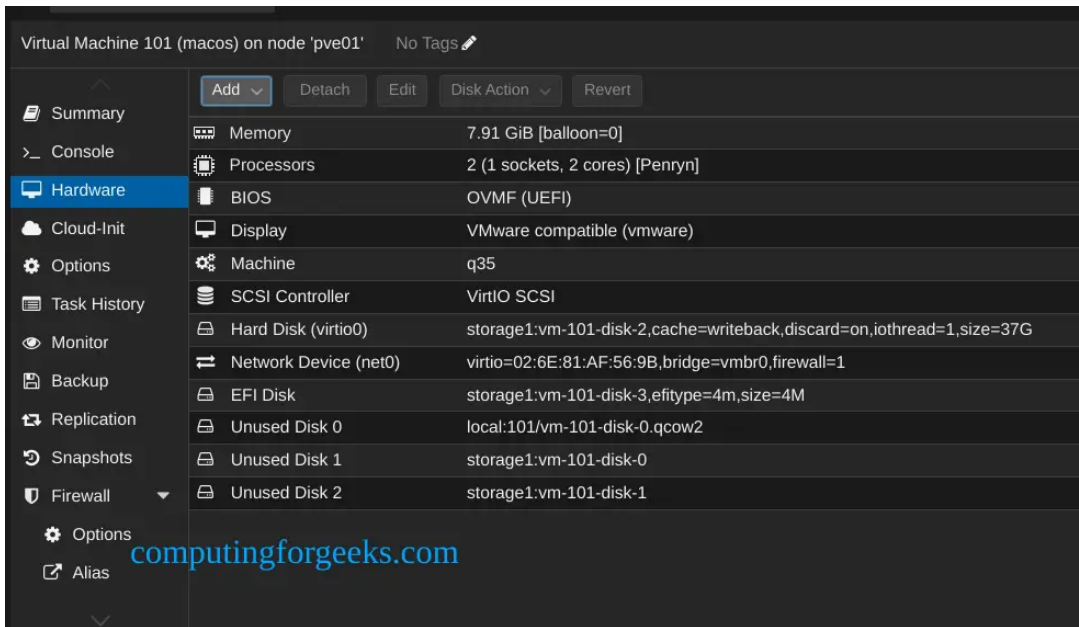


```
klinsmann@iMac-Pro ~ % sudo dd if=/dev/disk1s1 of=/dev/disk2s1
>Password:
307120+0 records in
307120+0 records out
157245440 bytes transferred in 67.471307 secs (2330553 bytes/sec)
klinsmann@iMac-Pro ~ %
klinsmann@iMac-Pro ~ %
```

computingforgeeks.com

## Step 7 – Boot into macOS installation

Now shut down the VM and remove both the OpenCore and the Monterey installer drives in the **Hardware** tab.



Virtual Machine 101 (macos) on node 'pve01' No Tags

Summary Console Hardware Cloud-Init Options Task History Monitor Backup Replication Snapshots Firewall Options Alias

computingforgeeks.com

Component	Value
Memory	7.91 GiB [balloon=0]
Processors	2 (1 sockets, 2 cores) [Penryn]
BIOS	OVMF (UEFI)
Display	VMware compatible (vmware)
Machine	q35
SCSI Controller	VirtIO SCSI
Hard Disk (virtio0)	storage1:vm-101-disk-2,cache=writeback,discard=on,iothread=1,size=37G
Network Device (net0)	virtio=02:6E:81:AF:56:9B,bridge=vbr0,firewall=1
EFI Disk	storage1:vm-101-disk-3,efitype=4m,size=4M
Unused Disk 0	local:101/vm-101-disk-0.qcow2
Unused Disk 1	storage1:vm-101-disk-0
Unused Disk 2	storage1:vm-101-disk-1

In the options tab, edit your boot order and place your hard disk as the first boot option.

Now start the VM, and if everything is okay, you should see this:

Login to the VM.

Once authenticated, you can proceed and make the desired configurations for your Mac system.



If you are unable to wake Monterey from sleep, using your mouse or keyboard, you can disable the system sleep in **Monterey's Energy Saver settings** to avoid the issue.

You can also wake the system manually using the command:

```
##From your Proxmox host
qm monitor YOUR-VM-ID-HERE
system_wakeup
quit
```

## Verdict

With that, we conclude this guide on how to run macOS on Proxmox VE. I hope you too managed to spin MacOS on Proxmox.

See more guides on this page:

- [How To Install Flatcar Container Linux in Proxmox VE](#)
- [Export Proxmox Virtual Machine and Run on KVM \(Libvirt\)](#)
- [Secure Proxmox VE Server With Let's Encrypt SSL](#)

**YOU CAN SUPPORT OUR WORK WITH A CUP OF COFFEE**

As we continue to grow, we would wish to reach and impact more people who visit and take advantage of the guides we have on our blog. This is a big task for us and we are so far extremely grateful for the kind people who have shown amazing support for our work over the time we have been online.

Thank You for your support as we work to give you the best of guides and articles. Click below to buy us a coffee.



**Klinsmann Öteyo**

A systems engineer with excellent skills in systems administration, cloud computing, systems deployment, virtualization, containers, and a certified ethical hacker.

