

Installing macOS Sierra 10.12 as a XEN guest

Installing macOS Sierra 10.12 as a XEN HVM guest (Clover / OpenCore)

Field	Value
Author	Anton Afanasyeu
Revision	R0
Creation date	2026-06-26
Last modification date	2026-06-26
Co-authored	Cursor Agent (project assistant)
Severity	low
State	in progress — HVM-only; CloverNG/OpenCore path
Document type	technical guide

Table of contents

1. Prerequisites and host requirements
 - XEN host (managed by Anton)
 - Caller actions (request from Anton)
 - Preparation materials needed
2. Critical constraint — HVM only, not paravirt
3. Architecture overview
4. Bootloader choice — Clover vs OpenCore
 - Clover (r5xxx)
 - OpenCore (0.9+)
 - Recommendation for XEN + OVMF
5. Tools and packages
 - On a Linux workstation (or the XEN dom0 / a Linux VM)
6. Step 1 — Convert .dmg to raw image
 - Option A — Linux with dmg2img
 - Option B — macOS workstation with hdiutil
 - Verify the image
7. Step 2 — Create a bootable macOS installer disk
 - If the image is a full installer (preferred)
 - If only BaseSystem.dmg is present (recovery path)
8. Step 3 — Prepare the target installation disk
9. Step 4 — Obtain and configure Clover or OpenCore EFI
 - Option A — OpenCore (recommended)
 - Option B — Clover (r5xxx)
 - Embed EFI into installer disk
10. Step 5 — XEN HVM domain configuration
11. Step 6 — AppleSMC and the OSK key
12. Step 7 — CPUID masking
13. Step 8 — Run the installer
 - Start the VM
 - Boot sequence
 - Installation steps
14. Step 9 — Post-install: Clover/OC on target disk
 - Mount the EFI partition of the target disk
 - Copy the EFI bootloader
 - Update XL config to boot from target disk only
15. Networking in the guest
16. Known issues and limitations
17. Troubleshooting
 - VM fails to start
 - Bootloader screen never appears (OVMF logo then black)
 - macOS installer kernel panic immediately
 - Installer runs but network is unavailable
 - VM reboots after install but shows installer again
18. Comparison: Clover vs OpenCore for XEN
19. References
20. Changelog

Source: *Installing_MacOS_Sierra_10.12_as_XEN_guest.md* — section links work in PDF viewers that support internal anchors.

Document type: Technical guide

PDF: Installing_MacOS_Sierra_10.12_as_XEN_guest.pdf · Regenerate: `bash ac-docs/build-pdf.sh`

Scope: Running macOS Sierra (10.12) as a XEN HVM guest using a .dmg source, OVMF UEFI firmware, and Clover or OpenCore EFI bootloader

> **Legal notice:** macOS is proprietary software. Apple's macOS EULA permits installation only on

> Apple-branded hardware. Running macOS on non-Apple hardware (including XEN virtual machines)

> violates Apple's EULA. This document is provided for research, testing, and educational purposes

> only. Do not use a macOS XEN guest in production or for commercial purposes without Apple-licensed

> hardware.

1. Prerequisites and host requirements

XEN host (managed by Anton)

Requirement	Value	Notes
XEN version	4.14+ recommended	Older may lack OVMF features
CPU	Intel with VT-x + VT-d	Xeon Gold 5218 ✓ — SSE4.1 present
OVMF package	Installed on XEN host	<code>apt install ovmf / apk add ovmf</code>
QEMU	Installed on XEN host	XEN HVM uses QEMU as device model
VNC or SPICE	Available for console	Needed during installation
VT-d	Optional but helpful	For USB passthrough of physical keyboard/mouse

Caller actions (request from Anton)

VM request:

```
type = hvm          ← mandatory; paravirt will NOT work (see §2)
memory = 4096       ← 4 GB minimum for installer
vcpus = 2
firmware = ovmf      ← UEFI required; traditional BIOS loader will not boot macOS Sierra
VNC console access  ← required during installation
```

Preparation materials needed

Item	Notes
Install macOS Sierra.app or its .dmg	From Apple or Hackintosh image archive
Clover r5xxx .zip or OpenCore .zip	See §4 for which to choose
macOS Sierra base system .dmg or BaseSystem.dmg	Embedded in the installer
An EFI config for Sierra	<code>config.plist</code> — community presets available
AppleSMC OSK string	Required by macOS kernel — see §11

2. Critical constraint — HVM only, not paravirt

macOS cannot run in XEN paravirt mode. This is a hard technical blocker, not a configuration issue:

Mode	Works?	Reason
HVM (type = "hvm")	Yes	Full hardware emulation (VT-x); macOS sees real x86 hardware
Paravirt (type = "pv")	No	Paravirt replaces hardware with Xen hypercalls; macOS has no XEN PV drivers and the boot sequence is incompatible
PVH	No	Hybrid mode still lacks the UEFI environment macOS requires

This means **the macOS guest must be an HVM VM**. If the project moves to paravirt-only nodes for cost/performance reasons, macOS cannot share that pool — it needs a dedicated HVM-capable slot.

Confirm with Anton that the XEN host has hardware VT-x enabled (`xm info | grep hvm` or `xl info | grep virt_caps` should show `hvm`).

3. Architecture overview

XEN Host (Gentoo / any Linux, Intel VT-x)

```
■
■ ■ ■ HVM Guest: macos-sierra
■
■ ■ ■ OVMF (UEFI firmware)      ← provided by XEN host, replaces BIOS
■   ■ ■ ■ boots from EFI partition on disk
■
■ ■ ■ EFI disk / partition
■   ■ ■ ■ Clover or OpenCore   ← macOS-aware UEFI bootloader
■       ■ ■ ■ injects AppleSMC
■       ■ ■ ■ patches CPUID (Hackintosh-style)
```

```

■          ■■■ provides fake SMBios (Mac model)
■          ■■■ chainloads macOS boot.efi
■
■■■ macOS installer disk (hdb)      ← converted from .dmg
■
■■■ Target disk (hda)                ← blank disk, macOS installs here

```

Why OVMF + Clover/OC (not just OVMF alone):

OVMF provides UEFI, but macOS additionally requires Apple-specific firmware tables (SMBios, AppleSMC, NVRAM layout). Clover and OpenCore are bootloaders that inject these tables into the EFI environment before handing off to macOS's `boot.efi`. Without them, macOS halts early with a kernel panic or simply refuses to boot on non-Apple hardware signatures.

4. Bootloader choice — Clover vs OpenCore

Clover (r5xxx)

- The original Hackintosh bootloader
- Loads from EFI partition; injects SMBios, AppleSMC, CPUID patches, kexts
- Mature ecosystem; many pre-made Sierra configs available
- Some quirks with NVRAM on OVMF; workaround: `EmuVariableUefi.efi` emulated NVRAM driver
- Active maintenance: github.com/CloverHackyColor/CloverBootloader

OpenCore (0.9+)

- Modern replacement for Clover; cleaner injection model
- Better NVRAM support with OVMF (uses `OpenVariableRuntimeDxe.efi` or OVMF's native NVRAM)
- Stricter config validation (`ocvalidate` catches errors before boot)
- Supports Sierra with minor adjustments (`MinVersion` quirks)
- Preferred for new setups
- Guide: dortania.github.io/OpenCore-Install-Guide

Recommendation for XEN + OVMF

OpenCore is recommended for new installations with OVMF on XEN — it handles OVMF's emulated NVRAM better than Clover and has active maintenance for recent XEN versions. Clover remains viable if you already have a working Clover config from a KVM/QEMU Hackintosh.

"CloverNG" — if this refers to the next-generation CloverBootloader (post-r5100 series from CloverHackyColor), it is functionally equivalent to standard Clover from a XEN perspective. If it refers to OpenCore, use the OpenCore path below.

5. Tools and packages

On a Linux workstation (or the XEN dom0 / a Linux VM)

```
apt install dmg2img qemu-utils p7zip-full
```

```
apk add dmg2img qemu-img p7zip
```

Tool	Purpose
dmg2img	Convert Apple DMG to raw IMG on Linux
qemu-img	Create/convert virtual disk images (qcow2, raw)
p7zip / 7z	Extract Clover or OpenCore zip archives
hdiutil	macOS native DMG tool (only if macOS workstation available)
fdisk / gdisk	Partition the installer image for EFI setup
mttools	Write files to FAT32 EFI partition without mounting as root

6. Step 1 — Convert .dmg to raw image

The `.dmg` from Apple (or third-party Hackintosh sources) must be converted to a flat raw image that XEN can use as a virtual disk.

Option A — Linux with `dmg2img`

```
dmg2img -i "Install_macOS_Sierra_10.12.6.dmg" -o sierra_installer.img
```

If `dmg2img` fails (some DMG variants use encryption or non-standard compression):

```
7z x "Install_macOS_Sierra_10.12.6.dmg" -o./sierra_extracted/
```

Option B — macOS workstation with `hdiutil`

```
hdiutil convert "Install macOS Sierra.dmg" -format UDRW -o sierra_rw
mv sierra_rw.dmg sierra_installer.img
```

Verify the image

```
qemu-img info sierra_installer.img
file sierra_installer.img
```

> **Note on EFI:** The user correctly identified that converting the DMG produces an EFI-based disk (GPT partition table with an EFI system partition). This is expected and correct for macOS Sierra. The installer image cannot be mounted as a simple CDROM/ISO — it must be presented as a virtual hard disk in HVM mode and booted via UEFI (OVMF). A traditional BIOS/MBR boot path does not exist for Sierra+.

7. Step 2 — Create a bootable macOS installer disk

The raw .img from step 6 may be the full Install macOS Sierra.app disk, or it may be a recovery-only image. Determine which you have:

```
apk add hfsprogs # or: apt install hfsprogs
mkdir /mnt/sierra_check
mount -t hfsplus -o ro sierra_installer.img /mnt/sierra_check
ls /mnt/sierra_check
```

If the image is a full installer (preferred)

Convert to qcow2 for efficient sparse storage:

```
qemu-img convert -f raw -O qcow2 sierra_installer.img sierra_installer.qcow2
```

This image becomes hdb (the installer disk) in the XEN config.

If only BaseSystem.dmg is present (recovery path)

```
cd /Applications/Install\ macOS\ Sierra.app/Contents/SharedSupport/
```

```
7z x sierra_installer.img InstallESD.dmg
dmg2img InstallESD.dmg installeds.img
```

The recovery path produces a minimal boot environment sufficient to run the macOS installer, but internet access or a local macOS packages cache may be needed.

8. Step 3 — Prepare the target installation disk

Create a blank virtual disk that macOS will install onto:

```
qemu-img create -f qcow2 macos_sierra_target.qcow2 60G
```

This disk starts blank. macOS Disk Utility (during installation) will partition and format it as GPT + HFS+.

9. Step 4 — Obtain and configure Clover or OpenCore EFI

Option A — OpenCore (recommended)

Download the latest OpenCore release from:

github.com/acidanthera/OpenCorePkg/releases

```
mkdir opencore_efi && cd opencore_efi
wget https://github.com/acidanthera/OpenCorePkg/releases/download/0.9.x/OpenCore-0.9.x-RELEASE.zip
unzip OpenCore-0.9.x-RELEASE.zip
```

Minimum EFI structure for Sierra on XEN/QEMU:

```
EFI/
  BOOT/
    BOOTx64.efi          ← OpenCore main loader
  OC/
    OpenCore.efi
    config.plist          ← main configuration
    ACPI/
      ← DSDT/SSDT patches (may be empty for VM)
    Drivers/
      OpenRuntime.efi     ← required
      ResetNvramEntry.efi
    Kexts/
      Lilu.kext           ← required base
      VirtualSMC.kext     ← AppleSMC emulation (replaces hardware SMC)
      WhateverGreen.kext  ← GPU fixes (optional for headless)
      IntelMausi.kext     ← for e1000 NIC (or use AppleIGB)
    Tools/
      OpenShell.efi      ← useful for debugging
```

Key config.plist settings for XEN:

```
<!-- Kernel → Quirks -->
<key>DisableIoMapper</key><true/>
<key>XhciPortLimit</key><true/>

<!-- Misc → Security -->
<key>ScanPolicy</key><integer>0</integer>

<!-- NVRAM → Add → 7C436110-... -->
<key>boot-args</key>
<string>-v keepsyms=1 debug=0x100</string> <!-- verbose boot for debugging -->
```

```
<!-- PlatformInfo → Generic -->
<!-- Use MacPro6,1 or iMac17,1 SMBIOS for Sierra compatibility -->
<key>SystemProductName</key><string>iMac17,1</string>
```

Generate MLB, ROM, SystemSerialNumber, SystemUUID using macserial (from OpenCore tools) or GenSMBIOS. Use random/generated values — do not reuse real Apple serial numbers.

Option B — Clover (r5xxx)

Download from:
github.com/CloverHackyColor/CloverBootloader/releases

Minimum EFI structure:

```
EFI/
  BOOT/
    BOOTX64.efi
  CLOVER/
    CLOVERX64.efi
    config.plist
    drivers/
      UEFI/
        ApfsDriverLoader.efi ← for APFS if upgrading later
        AptioMemoryFix.efi ← NVRAM fix for OVMF
        EmuVariableUefi.efi ← emulated NVRAM (important for XEN/OVMF)
        FSInject.efi
        HFSPlus.efi ← HFS+ filesystem driver
    kexts/
      Other/
        Lilu.kext
        VirtualSMC.kext ← or FakeSMC.kext (older alternative)
        IntelMausi.kext
```

Key config.plist settings for XEN/OVMF:

```
<key>RtVariables</key>
<dict>
  <key>LogEveryBoot</key><string>10</string>
  <key>MountEFI</key><string>Yes</string>
</dict>

<key>SMBIOS</key>
<dict>
  <key>ProductName</key><string>iMac17,1</string>
  <!-- Generate BoardSerialNumber, SerialNumber, SmUUID with Clover Configurator -->
</dict>

<!-- Boot → Arguments -->
<key>Arguments</key><string>-v dart=0 debug=0x100 keepsyms=1</string>
```

Embed EFI into installer disk

The Clover/OC EFI files need to be on the **EFI system partition (ESP)** of the installer disk or on a separate small EFI disk image.

Recommended: separate EFI disk (simplest for XEN)

```
dd if=/dev/zero of=efi_boot.img bs=1M count=200
parted efi_boot.img mklabel gpt
parted efi_boot.img mkpart EFI fat32 1MiB 199MiB
parted efi_boot.img set 1 esp on

mkfs.fat -F32 -n EFI \
  --offset 2048 \      # 1MiB in 512-byte sectors
  efi_boot.img         # note: loopback mount preferred
```

```
losetup -P /dev/loop0 efi_boot.img
mkfs.fat -F32 /dev/loop0p1
mount /dev/loop0p1 /mnt/efi/
cp -r EFI/ /mnt/efi/
umount /mnt/efi
losetup -d /dev/loop0
```

This efi_boot.img becomes hdc (or hda — first boot device) in the XEN config.

10. Step 5 — XEN HVM domain configuration

Create /etc/xen/macos-sierra.cfg:

```
type = "hvm"
name = "macos-sierra"
memory = 4096          # 4 GB minimum; 8 GB recommended
vcpus = 2
maxvcpus = 2
firmware = "ovmf"      # UEFI; critical – do NOT use "bios"

disk = [
  'file:/path/to/efi_boot.img,hda,w',
  'file:/path/to/sierra_installer.qcow2,hdb,r',
  'file:/path/to/macos_sierra_target.qcow2,hdc,w',
]

vif = ['type=ioemu,model=e1000']

usbdevice = 'tablet'
vnc = 1
```

```

vnclisten = "127.0.0.1"
vncpasswd = ""
vncunused = 1          # picks next free port starting at 5900

serial    = "pty"

device_model_args = [
    # AppleSMC - macOS kernel checks for this device; OSK must be set (see §11)
    "-device", "applesmc,osk=<REPLACE_WITH_OSK>",
    # Cirrus VGA (simplest; QXL or VirtIO may not work during macOS install)
    "-vga", "cirrus",
    # Sound (optional; macOS may probe for it)
    "-device", "ich9-intel-hda",
    "-device", "hda-duplex",
]

cpuid = [
    "0x80000001:ecx=xxxxxxxxxxxxxxxxxxxxxxxxxxxx0",
]
> On firmware = "ovmf": XEN must have the OVMF binary available. The path is typically
> /usr/share/ovmf/OVMF.fd or /usr/share/qemu/OVMF.fd. If XEN doesn't find it, it falls back
> to the legacy BIOS hvmloder — check with Anton that OVMF is installed on the host.

```

11. Step 6 — AppleSMC and the OSK key

The Apple System Management Controller (SMC) is a coprocessor present in all Apple hardware. macOS checks for its presence during boot. QEMU (used as XEN's HVM device model) can emulate an SMC with:

```
-device applesmc,osk=<128-char-hex-string>
```

The OSK (Output Signal Key) is a 64-byte string specific to Apple hardware. It is technically part of Apple's proprietary firmware but is widely documented in the Hackintosh community.

How to obtain the OSK:

- From a physical Mac: run an AppleSMC reader tool (e.g. `smcutil`)
- From the Hackintosh community: the OSK for common boards is documented; search for "QEMU macOS applesmc osk" — the value is the same across all OSK-based macOS versions
- The OSK is 64 bytes (128 hex characters) passed to `-device applesmc,osk=...`

> **Do not confuse** the OSK with Apple serial numbers. The OSK is a fixed hardware key, not a device identifier. It does not uniquely identify a machine.

VirtualSMC vs hardware applesmc:

If using OpenCore with `VirtualSMC.kext`, the `kext` handles SMC emulation in software and the `applesmc` QEMU device may not be strictly required. However, including both is safe and the hardware device takes precedence. For Clover with `FakeSMC.kext`, the `kext` similarly provides SMC emulation.

12. Step 7 — CPUID masking

macOS detects hypervisors via the `HYPERVISOR` CPUID bit and may behave differently or refuse to boot. Additionally, XEN presents XEN-specific CPU topology that can confuse macOS.

Mask the hypervisor bit (XL config):

```

cpuid = [
    "0x80000001:ecx=xxxxxxxxxxxxxxxxxxxxxxxxxxxx0",
]

```

Set a clean CPU vendor string (optional but recommended):

```

device_model_args = [
    ...,
    "-cpu", "host,vendor=GenuineIntel,+sse4.1,+sse4.2,-hypervisor,kvm=off",
]

```

CPU topology (macOS prefers 1 socket × N cores):

```

vcpus      = 2
cores_per_socket = 2
threads_per_core = 1

```

`-hypervisor` removes the CPUID hypervisor bit at the QEMU level; the `XL cpuid` directive masks it at the XEN level. Apply both for robustness.

13. Step 8 — Run the installer

Start the VM

```

xl create /etc/xen/macOS-sierra.cfg
xl list
vncviewer 127.0.0.1:5900

```

Boot sequence

1. OVMF loads → shows TianoCore logo briefly
2. OVMF detects the EFI partition on hda → loads `BOOTx64.efi` (Clover or OC)

3. Bootloader screen appears — select "Install macOS Sierra" or "Boot macOS Install from ..."
4. macOS boot sequence (verbose if -v set) — expect ~5–10 minutes for kernel load on first boot
5. macOS installer GUI appears (language selection)

Installation steps

1. **Language selection** → proceed
 2. **Disk Utility** → select hdc (target, ~60 GB) → Erase → Format: **Mac OS Extended (Journaled)**, scheme: **GUID Partition Map** → name it (e.g. "Macintosh HD")
 3. Back to installer → Install macOS → select "Macintosh HD"
 4. Installation runs (~30–60 min); VM will restart several times automatically — each time it should boot back to the installer progress screen
 5. Final restart → macOS Setup Assistant (user account, locale, etc.)
- > **If the VM reboots but boots the installer again** instead of the installed disk: the EFI boot order has not changed. In the bootloader menu, manually select the installed disk. After install completes, update the bootloader config to default-boot the target disk.

14. Step 9 — Post-install: Clover/OC on target disk

After installation, the target disk (hdc) has macOS installed but no bootloader. Move the EFI setup to the target disk so it boots independently.

Mount the EFI partition of the target disk

```
losetup -P /dev/loop1 macos_sierra_target.qcow2 # may need qemu-nbd for qcow2
mount /dev/loop1p1 /mnt/target_efi # p1 = EFI partition
```

Copy the EFI bootloader

```
cp -r /mnt/efi/EFI/ /mnt/target_efi/
umount /mnt/target_efi
```

Update XL config to boot from target disk only

```
disk = [
    'file:/path/to/macos_sierra_target.qcow2,hda,w',
]
```

OVMF will now boot from the EFI partition on the target disk → Clover/OC → macOS Sierra.

15. Networking in the guest

XEN HVM with `model=e1000` emulates the Intel 82540EM NIC. macOS Sierra includes built-in drivers for this chipset (`AppleIntelE1000e.kext` or loaded via `IntelMausi.kext`).

Expected: Ethernet appears automatically in System Preferences → Network after install.

DHCP: Works out of the box if the XEN virtual network bridge provides DHCP (standard XEN networking with `xenbr0` and the host running `dnsmasq`).

Static IP: Set in macOS Network preferences if needed.

No internet to iCloud/App Store: macOS will detect non-Apple hardware signatures (unless correct SMBIOS + serial number is configured) and refuse App Store login. For pure testing purposes, network access to local services and `apps.f0xx.org` should work fine without this.

16. Known issues and limitations

Issue	Cause	Workaround / Status
No GPU acceleration	Cirrus VGA is 2D only; Metal requires real GPU or PCIe passthrough	Use headless or accept software rendering; Cirrus sufficient for testing
Kernel panic on boot	Missing SMC, bad CPUID, wrong SMBIOS	Check §11–§12; add -v to boot args for verbose panic
NVRAM not persisting	OVMF emulated NVRAM resets on shutdown	Use <code>EmuVariableUefi.efi</code> (Clover) or <code>OpenVariableRuntimeDxe.efi</code> (OC); or save NVRAM to file
USB mouse/keyboard lost	USB tablet works but real USB may need passthrough	<code>usbdevice = 'tablet'</code> in XL config handles mouse; keyboard via VNC
macOS updates may break boot	OTA updates overwrite <code>boot.efi</code> and may change kext requirements	Snapshot before updating; keep bootloader kexts updated
iCloud / App Store blocked	Missing/wrong Apple serial; Apple server check	Expected for VM use; not needed for testing/dev purposes
Very slow boot (> 15 min)	QEMU Cirrus VGA rendering overhead; slow disk emulation	Use <code>qcow2</code> with preallocation or raw disk; increase memory
Stuck at "2 minutes remaining"	Common Sierra installer bug	Wait; may take 15+ min; do not power off
HVM slot unavailability	Anton's XEN host may limit HVM guest count	Coordinate with Anton; confirm <code>\xl</code> info

Sound/audio	ICH9-HDA device available but macOS may not auto-configure	Install VoodooHDA.kext if audio needed
-------------	--	--

17. Troubleshooting

VM fails to start

```
xl create -c /etc/xen/macOS-sierra.cfg
```

Check: does `xl info | grep hvm` show `hvm=1`? If not, VT-x is not available or enabled.

Bootloader screen never appears (OVMF logo then black)

- OVMF did not find a bootable EFI on hda
- Verify: `efi_boot.img` has a valid GPT ESP partition with `EFI/B00T/B00Tx64.efi`
- In OVMF shell (press F2 or Esc during TianoCore logo): `run Shell → fs0: → dir EFI`

macOS installer kernel panic immediately

- Check AppleSMC OSK is set correctly (non-empty, correct length)
- Add `-v -x` (verbose + safe mode) to boot args
- Check CPUID masking (§12) — remove it and try without; then add back

Installer runs but network is unavailable

- Verify XEN `vif` is correct and `xenbr0` is up on host: `ip link show xenbr0`
- Check macOS System Preferences → Network for the Ethernet adapter
- Install `IntelMausi.kext` if the NIC is not recognized

VM reboots after install but shows installer again

- OVMF boot order still prefers hda (EFI boot disk) over the installed hdc
- In bootloader menu: manually select the installed macOS disk
- After booting successfully once: update bootloader config default entry

18. Comparison: Clover vs OpenCore for XEN

Aspect	Clover (r5xxx)	OpenCore (0.9+)
NVRAM on OVMF	Requires <code>EmuVariableUefi.efi</code>	Native OVMF NVRAM support; more reliable
Sierra support	Excellent; many Sierra configs available	Good; needs <code>MinVersion/MaxVersion</code> quirks
QEMU/XEN compatibility	Mature; known-good with QEMU 5+	Excellent; actively tested on QEMU/KVM/XEN
Config validation	Manual; errors only visible at boot	<code>ocvalidate</code> CLI tool catches errors before boot
Kext injection	Flexible; some ordering quirks	Strict ordering; predictable
Long-term maintenance	Slower; fewer active contributors	Active; new releases regularly
Community resources for XEN	Fewer specific guides	Growing; Docker-OSX project uses OC on KVM (close to XEN)
Recommended for new install	Only if migrating existing config	Yes

19. References

Resource	URL / note
OpenCore Install Guide	https://dortania.github.io/OpenCore-Install-Guide/
OpenCore Legacy Patcher	https://dortania.github.io/OpenCore-Legacy-Patcher/ (Sierra section)
CloverBootloader releases	https://github.com/CloverHackyColor/CloverBootloader
Docker-OSX (KVM/QEMU reference)	https://github.com/sickcodes/Docker-OSX — useful architecture reference even for XEN
VirtualSMC kext	https://github.com/acidanthera/VirtualSMC
Lilu kext	https://github.com/acidanthera/Lilu
IntelMausi NIC kext	https://github.com/acidanthera/IntelMausi
QEMU applesmc documentation	<code>man qemu-system-x86_64</code> ; <code>search -device applesmc</code>
XEN HVM OVMF setup	https://wiki.xenproject.org/wiki/OVMF
GenSMBIOS (serial generation)	https://github.com/corpnewt/GenSMBIOS

20. Changelog

Rev	Date	Change
R0	2026-06-26	Initial guide — HVM+OVMF+OpenCore path; Clover alternative; known issues