Xen Project Software Overview - Xen

34-43 minutes

What is the Xen Project Hypervisor?

The Xen Project hypervisor is an open-source type-1 or baremetal hypervisor, which makes it possible to run many instances of an operating system or indeed different operating systems in parallel on a single machine (or host). The Xen Project hypervisor is the only type-1 hypervisor that is available as open source. It is used as the basis for a number of different commercial and open source applications, such as: server virtualization, Infrastructure as a Service (IaaS), desktop virtualization, security applications, embedded and hardware appliances. The Xen Project hypervisor is powering the largest clouds in production today.

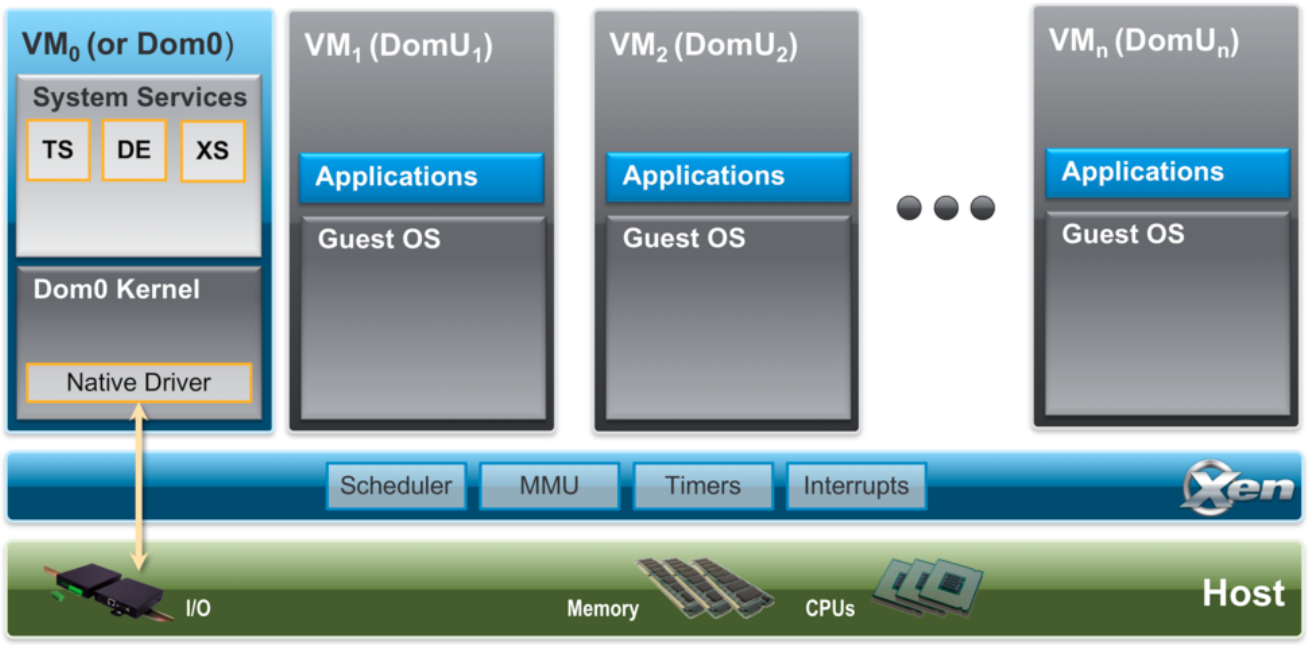
Here are some of the Xen Project hypervisor's key features:

- Small footprint and interface (is around 1MB in size). Because it uses a microkernel design, with a small memory footprint and limited interface to the guest, it is more robust and secure than other hypervisors.
- Operating system agnostic: Most installations run with Linux as the main control stack (aka "domain 0"). But a number of other operating systems can be used instead, including NetBSD and OpenSolaris.
- Driver Isolation: The Xen Project hypervisor has the capability to allow the main device driver for a system to run inside of a virtual machine. If the driver crashes, or is compromised, the VM containing the driver can be rebooted and the driver restarted without affecting the rest of the system.
- Paravirtualization: Fully paravirtualized guests have been optimized to run as a virtual machine. This allows the guests to run much faster than with hardware extensions (HVM). Additionally, the hypervisor can run on hardware that doesn't support virtualization extensions.

This page will explore the key aspects of the Xen Project architecture that a user needs to understand in order to make the best choices.

Introduction to Xen Project Architecture

Below is a diagram of the Xen Project architecture. The Xen Project hypervisor runs directly on the hardware and is responsible for handling CPU, Memory, timers and interrupts. It is the first program running after exiting the bootloader. On top of the hypervisor run a number of virtual machines. A running instance of a virtual machine is called a **domain** or **guest**. A special domain, called domain 0 contains the drivers for all the devices in the system. Domain 0 also contains a control stack and other system services to manage a Xen based system. Note that through Dom0 Disaggregation it is possible to run some of these services and device drivers in a dedicated VM: this is however not the normal system set-up.

- 

Components in detail:

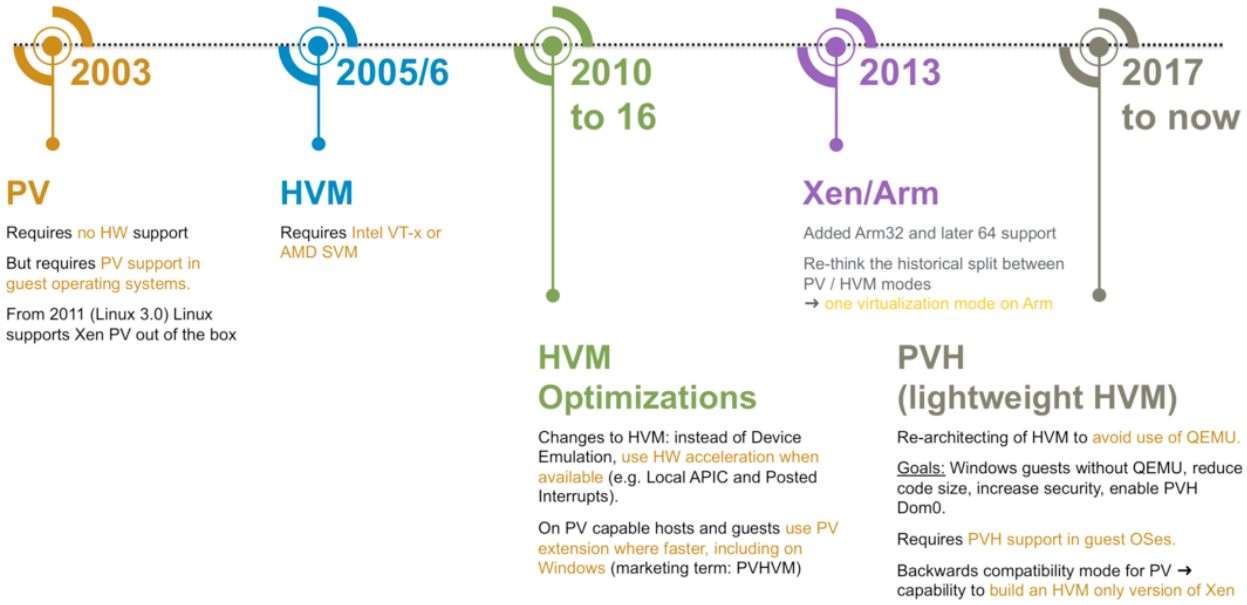
- **The Xen Project Hypervisor** is an exceptionally lean (<65KSLOC on Arm and <300KSLOC on x86) software layer that runs directly on the hardware and is responsible for managing CPU, memory, and interrupts. It is the first program running after the bootloader exits. The hypervisor itself has no knowledge of I/O functions such as networking and storage.
- **Guest Domains/Virtual Machines** are virtualized environments, each running their own operating system and applications. The hypervisor supports several different virtualization modes, which are described in more detail below. Guest VMs are totally isolated from the hardware: in other words, they have no privilege to access hardware or I/O functionality. Thus, they are also called unprivileged domain (or DomU).
- **The Control Domain (or Domain 0)** is a specialized Virtual Machine that has special privileges like the capability to access the hardware directly, handles all access to the system's I/O functions and interacts with the other Virtual Machines. he Xen Project hypervisor is not usable without Domain 0, which is the first VM started by the system. In a standard set-up, Dom0 contains the following functions:
  - **System Services**: such as XenStore/XenBus (XS) for managing settings, the Toolstack (TS) exposing a user interface to a Xen based system, Device Emulation (DE) which is based on QEMU in Xen based systems
  - **Native Device Drivers**: Dom0 is the source of physical device drivers and thus native hardware support for a Xen system
  - **Virtual Device Drivers**: Dom0 contains virtual device drivers (also called backends).
  - **Toolstack**: allows a user to manage virtual machine creation, destruction, and configuration. The toolstack exposes an interface that is either driven by a command line console, by a graphical interface or by a cloud orchestration stack such as OpenStack or CloudStack. Note that several different toolstacks can be used with Xen
- **Xen Project-enabled operating systems**: Domain 0 requires a Xen Project-enabled kernel. Paravirtualized guests require a PV-enabled guest. Linux distributions that are based on Linux kernels newer than Linux 3.0 are Xen Project-enabled and usually include packages that contain the hypervisor and Tools (the default Toolstack and Console). All but legacy Linux kernels older than Linux 2.6.24 are PV-enabled, capable of running PV guests.

Also see:

- Xen Project Release Features
- Xen Project-Enabled operating systems

Guest Types

The following diagrams show how guest types have evolved for Xen.



On ARM hosts, there is only one guest type, while on x86 hosts the hypervisor supports the following three types of guests:

- **Paravirtualized Guests or PV Guests:** PV is a software virtualization technique originally introduced by the Xen Project and was later adopted by other virtualization platforms. PV does not require virtualization extensions from the host CPU, but requires Xen-aware guest operating systems. PV guests are primarily of use for legacy HW and legacy guest images and in special scenarios, e.g. special guest types, special workloads (e.g. Unikernels), running Xen within another hypervisor without using nested hardware virtualization support, as container host, …
- **HVM Guests:** HVM guests use virtualization extensions from the host CPU to virtualize guests. HVM requires Intel VT or AMD-V hardware extensions. The Xen Project software uses QEMU device models to emulate PC hardware, including BIOS, IDE disk controller, VGA graphic adapter, USB controller, network adapter, etc. HVM Guests use PV interfaces and drivers when they are available in the guest (which is usually the case on Linux and BSD guests). On Windows, drivers are available to download via our download page. When available, HVM will use Hardware and Software Acceleration, such as Local APIC, Posted Interrupts, Viridian (Hyper-V) enlightenments and make use of guest PV interfaces where they are faster. Typically HVM is the best performing option on for Linux, Windows, *BSDs.
- **PVH Guests:** PVH guests are lightweight HVM-like guests that use virtualization extensions from the host CPU to virtualize guests. Unlike HVM guests, PVH guests do not require QEMU to

emulate devices, but use PV drivers for I/O and native operating system interfaces for virtualized timers, virtualized interrupt and boot. PVH guests require PVH enabled guest operating system. This approach is similar to how Xen virtualizes ARM guests, with the exception that ARM CPUs provide hardware support for virtualized timers and interrupts.

	IMPORTANT: Guest types are selected through `builder` configuration file option for Xen 4.9 or before and the `type` configuration file option from Xen 4.10 onwards in the (also see man pages).

PV (x86)

Paravirtualization (PV) is a virtualization technique originally introduced by Xen Project, later adopted by other virtualization platforms. PV does not require virtualization extensions from the host CPU and is thus ideally suited to run on older hardware. However, paravirtualized guests require a PV-enabled kernel and PV drivers, so the guests are aware of the hypervisor and can run efficiently without emulation or virtual emulated hardware. PV-enabled kernels exist for Linux, NetBSD and FreeBSD. Linux kernels have been PV-enabled from 2.6.24 using the Linux pvops framework. In practice this means that PV will work with most Linux distributions (with the exception of very old versions of distros).

Also see:

- More Information...
- PV Specific Config Options

HVM and its variants (x86)

Full Virtualization or Hardware-assisted virtualization (HVM) uses virtualization extensions from the host CPU to virtualize guests. HVM requires Intel VT or AMD-V hardware extensions. The Xen Project software uses QEMU to emulate PC hardware, including BIOS, IDE disk controller, VGA graphic adapter, USB controller, network adapter etc. Virtualization hardware extensions are used to boost performance of the emulation. Fully virtualized guests do not require any kernel support. This means that Windows operating systems can be used as a Xen Project HVM guest. For older host operating systems, fully virtualized guests are usually slower than paravirtualized guests, because of the required emulation.

To address this, the Xen Project community has upstreamed PV drivers and interfaces to Linux and other open source operating systems. On operating systems with Xen Support, these drivers and software interfaces will be automatically used when you select the HVM virtualization mode. On Windows this requires that appropriate PV drivers are installed. You can find more information at

- Windows PV Driver Downloads
- Windows PV Drivers Portal
- 3rd Party GPL PV Drivers (signed drivers are available)

HVM mode, even with PV drivers, has a number of things that are unnecessarily inefficient. One example are the interrupt controllers: HVM mode provides the guest kernel with emulated interrupt controllers (APICs and IOAPICs). Each instruction that interacts with the APIC requires a call into Xen and a software instruction decode; and each interrupt delivered requires several of these emulations.

Many of the paravirtualized interfaces for interrupts, timers, and so on are available for guests running in HVM mode: when available in the guest - which is true in most modern versions of Linux, *BSD and Windows - HVM will use these interfaces. This includes Viridian (i.e. Hyper-V) enlightenments which ensure that Windows guests are aware they are virtualized, which speeds up Windows workloads running on Xen.

When HVM improvements were introduced, we used marketing labels to describe HVM improvements. This seemed like a good strategy at the time, but has since created confusion amongst users. For example, we talked about PVHVM guests to describe the capability of HVM guests to use PV interfaces, even though PVHVM guests are just HVM guests. The following table gives an overview of marketing terms that were used to describe stages in the evolution of HVM, which you will find occasionally on the Xen wiki and in other documentation:

Label	Xen	Mode	With
HVM / Fully Virtualized	3.0+	HVM	No PV I/O Drivers
HVM + PV drivers	3.0+	HVM	PV I/O Drivers
PVHVM	4.0+	HVM	Guest operating system support (e.g. Linux 2.6.36 or newer, FreeBSD 10 or newer)

Compared to PV based virtualization, HVM is generally faster.

Also see:

- HVM specific Config options

PVH (x86)

A key motivation behind PVH is to combine the best of PV and HVM mode and to simplify the interface between operating systems with Xen Support and the Xen Hypervisor. To do this, we had two options: start with a PV guest and implement a "lightweight" HVM wrapper around it (as we have done for ARM) or start with a HVM guest and remove functionality that is not needed. The first option looked more promising based on our experience with the Xen ARM port, than the second. This is why we started developing an experimental virtualization mode called PVH (now called PVHv1) which was delivered in Xen Project 4.4 and 4.5. Unfortunately, the initial design did not simplify the operating system - hypervisor interface to the degree we hoped: thus, we started a project to evaluate a second option, which was significantly simpler. This led to PVHv2 (which in the early days was also called HVMLite). PVHv2 guests are lightweight HVM guests which use Hardware virtualization support for memory and privileged instructions, PV drivers for I/O and native operating system interfaces for everything else. PVHv2 also does not use QEMU for device emulation, but it can still be used for user-space backends (see PV I/O Support)..

PVHv1 has been replaced with PVHv2 in Xen 4.9, and has been made fully supported in Xen 4.10. PVH (v2) requires guests with Linux 4.11 or newer kernel.

Also see:

- PVH specific Config options (Xen 4.10+)

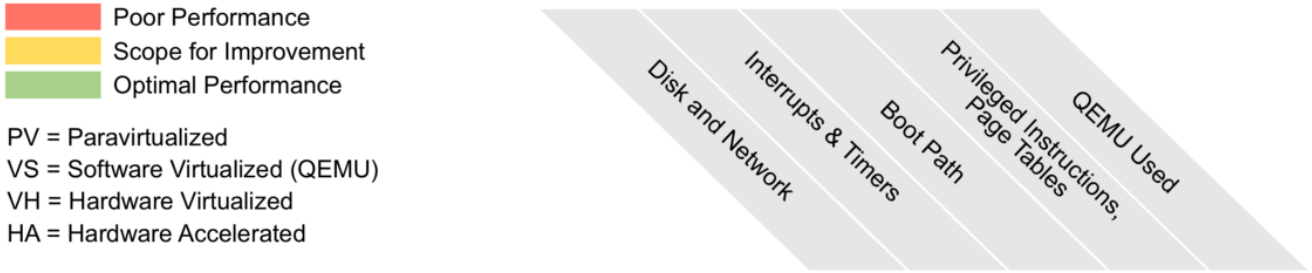
- Currently PVH only supports Direct Kernel Boot. EFI support is currently being developed.

ARM Hosts

On ARM hosts, there is only one virtualization mode, which does not use QEMU.

Summary

The following diagram gives an overview of the various virtualization modes implemented in Xen. It also shows what underlying virtualization technique is used for each virtualization mode.



x86 Shortcut	Mode	With					
HVM / Fully Virtualized	HVM		VS	VS[1]	VS	VH	Yes
HVM + PV drivers	HVM	PV Drivers Installed	PV	VS[1]	VS	VH	Yes
PVHVM	HVM	PVHVM Capable Guest	PV	PV[2]	VS	VH	Yes
PVH	PVH	PVH Capable Guest	PV	HA[3]	PV[4]	VH	No
PV	PV		PV	PV	PV[5]	PV	No
ARM							
N/A	N/A		PV	VH	PV[6]	VH	No

Footnotes:

1. Uses QEMU on older hardware and hardware acceleration on newer hardware – see 3)
2. Always uses Event Channels
3. Implemented in software with hardware accelerator support from IO APIC and posted interrupts
4. PVH uses Direct Kernel Boot or PyGrub. EFI support is currently being developed.
5. PV uses PvGrub for boot
6. ARM guests use EFI boot or Device Tree for embedded applications

From a users perspective, the virtualization mode primarily has the following effects:

- Performance and memory consumption will differ depending on virtualization mode
- A number of command line and config options will depend on the virtualization mode
- The boot path and guest install on HVM and PV/PVH are different: the workflow of installing guest operating systems in HVM guests is identical to installing real hardware, whereas installing guest OSes in PV/PVH guests differs. Please refer to the boot/install section of this document

Toolstack and Management APIs

Xen Project software employs a number of different toolstacks. Each toolstack exposes an API, against which a different set of tools or user-interface can be run. The figure below gives a very brief overview of the choices you have, which commercial products use which stack and examples of hosting vendors using specific APIs.

- Boxes marked in blue are developed by the Xen Project

The Xen Project software can be run with the default toolstack, with Libvirt and with XAPI. The pairing of the Xen Project hypervisor and XAPI became known as XCP which has been superseded by open source XenServer and XCP-ng. The diagram above shows the various options: all of them have different trade-offs and are optimized for different use-cases. However in general, the more on the right of the picture you are, the more functionality will be on offer.

Which to Choose?
The article Choice of ToolStacks gives you an overview of the various options, with further links to tooling and stacks for a specific API exposed by that toolstack.

xl

For the remainder of this document, we will however assume that you are using the default toolstack with its command line tool XL. These are described in the project's Man Pages. xl has two main parts:

- The **xl command line tool**, which can be used to create, pause, and shutdown domains, to list current domains, enable or pin VCPUs, and attach or detach virtual block devices. It is normally run as root in Dom0
- Domain **configuration files**, which describe per domain/VM configurations and are stored in the Dom0 filesystem

I/O Virtualization in Xen

The Xen Project Hypervisor supports the following techniques for I/O Virtualization:

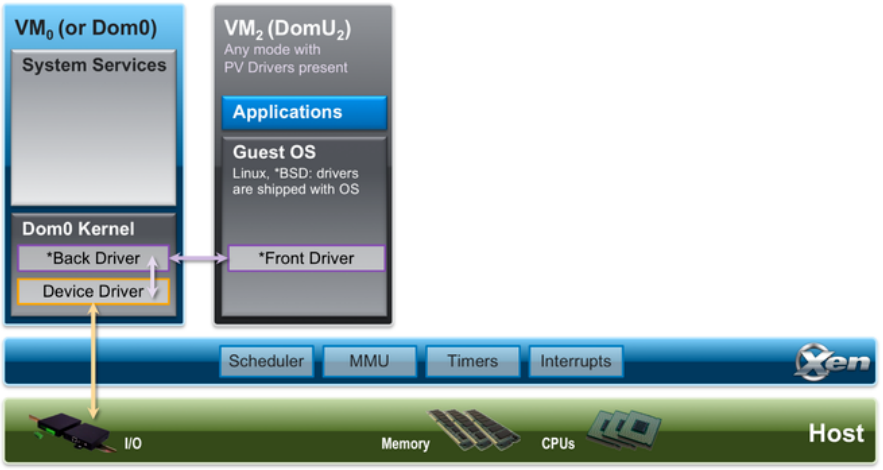
- **The PV split driver model**: in this model, a virtual front-end device driver talks to a virtual back-end device driver which in turn talks to the physical device via the (native) device driver. This enables multiple VMs to use the same hardware resource while being able to re-use native hardware support. In a standard Xen configuration, (native) device drivers and the virtual back-end device drivers reside in Dom0. Xen does allow running device drivers in so-called driver domains. PV based I/O virtualization is the primary I/O virtualization method for disk and network, but there are a host of PV drivers for DRM, Touchscreen, Audio, … that have been developed for non-server use of Xen. This model is independent of the virtualization mode used by Xen and

merely depends on the presence of the relevant drivers. These are shipped with Linux and *BSD out-of-the box. For Windows drivers have to be downloaded and installed into the guest OS.

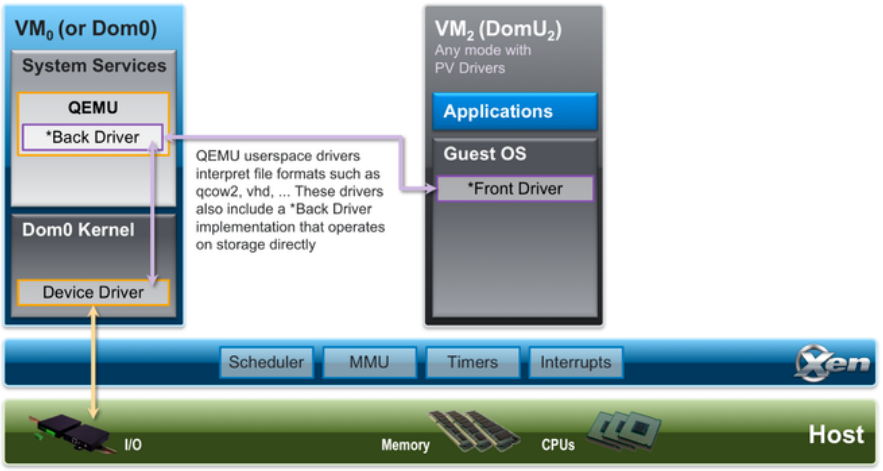
- **Device Emulation Based I/O**: HVM guests emulate hardware devices in software. In Xen, QEMU is used as Device Emulator. As the performance overhead is high, Device Based emulation is normally only used during system boot or installation and for low-bandwidth devices.
- **Passthrough**: allows you to give control of physical devices to guests. In other words, you can use PCI passthrough to assign a PCI device (NIC, disk controller, HBA, USB controller, FireWire controller, sound card, etc) to a virtual machine guest, giving it full and direct access to the PCI device. Xen supports a number of flavours of PCI passthrough, including VT-d passthrough and SR-IOV. However note that using passthrough has security implications, which are well documented here.

PV I/O Support

The following two diagrams shows two variants of the PV split driver model as implemented in Xen:



- I/O Virtualization using the split driver model



- I/O Virtualization using QEMU user space back-end drivers

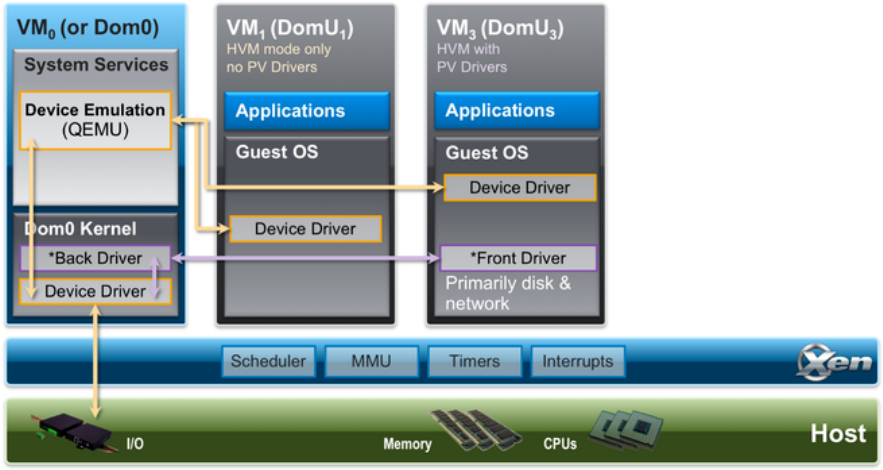
In the first model, a PV front-end driver will talk directly to a PV back-end driver in the Dom0 kernel. This model is primarily used for plain networking and storage virtualization with LVM, iSCSI, DRBD, etc. Note that the above figures are simplified representations of what is happening in a Xen stack, as even in the simplest cases there will be the Linux/BSD network/block stack in between the back-end driver and the real hardware device.

In the second model, a QEMU user-space backend will interpret formatted file data (such as qcow2, vmdk, vdi, etc.) and presents a raw disk interface to its own PV back-end implementation.

From a user's or guest's perspective, there is no visible difference to whether a back-end driver runs in user or kernel space. Xen will automatically choose the appropriate combination of front and back-end drivers based on the configuration option used.

HVM I/O Support

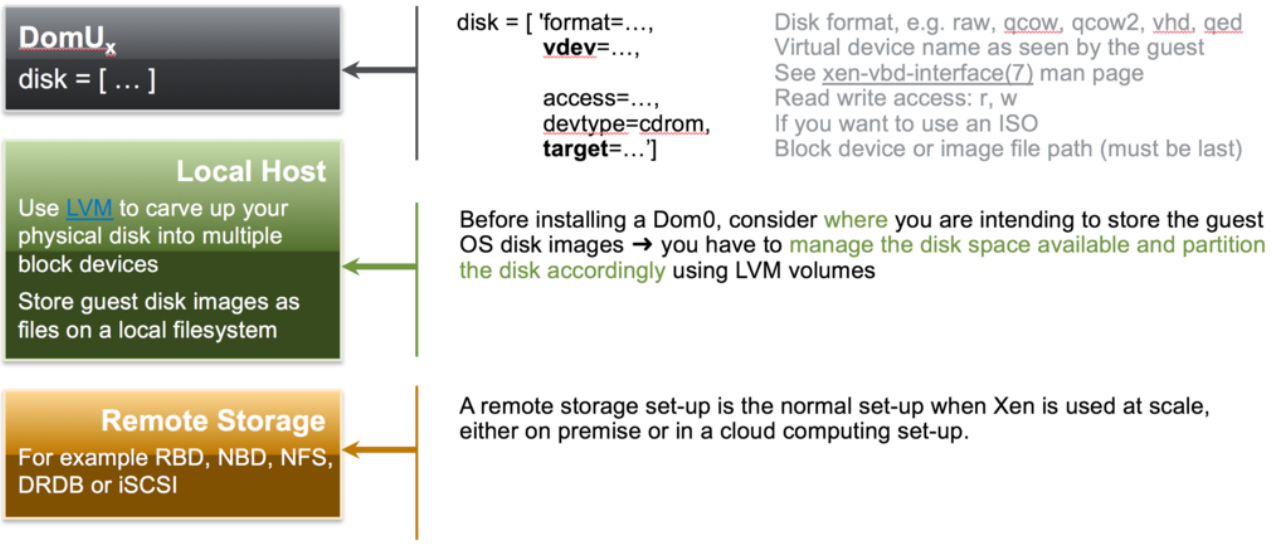
The following diagram shows how Device Emulation is used in isolation and used together with PV I/O Support.



This support is only available for HVM Guests and primarily used to emulate legacy devices that are needed during the boot process of a guest. It is also used for low bandwidth devices, such as the serial console for HVM guests.

Storage

The following picture gives a brief overview of Storage Options with Xen

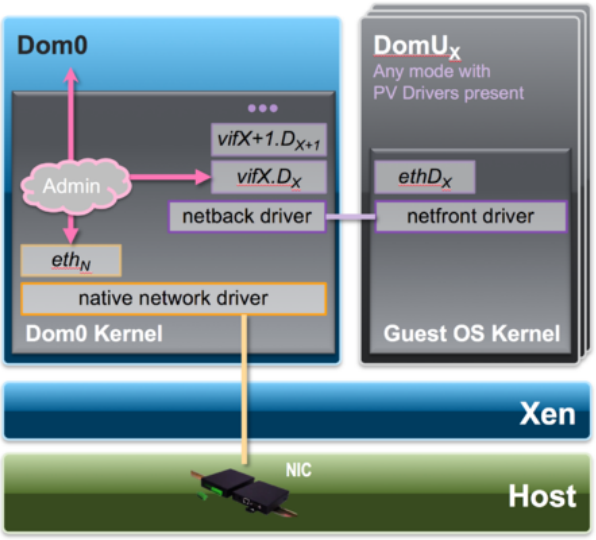


Defining storage is relatively straightforward, but requires some planning when used at scale. This applies to Xen Project Software as well as other virtualization solutions. For more information see:

- xl-disk-configuration(5)
- Storage options

Networking

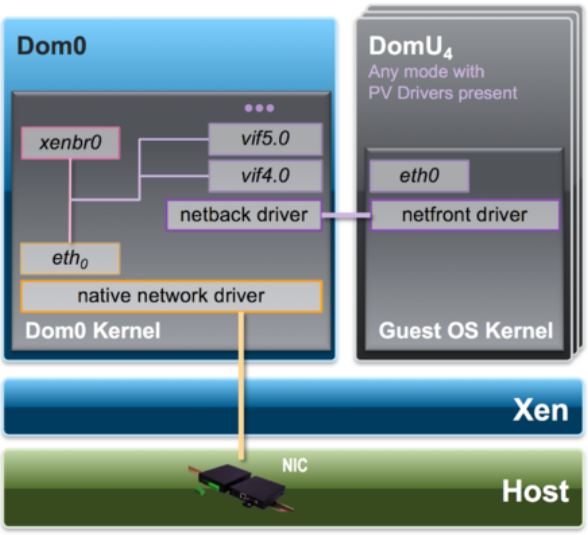
With xl, the host networking configuration is not configured by the toolstack. In general, the xl toolstack follows the philosophy of not implementing functionality that is available in the Host OS: setting up networking as well as managing system services are examples. Thus, the host administrator needs to setup an appropriate network configuration in Dom0 using native Linux/BSD tools using one of the following common networking styles: Bridging (most common), Open vSwitch, Routing, NAT. This is usually done immediately **after** Xen has been installed. See picture below:



To do this you may have to:

- **Step 1:** install bridging software packages, if not present
- **Step 2:** set up a network bridge (xenbr0) in Dom0. This is distro specific: you can find a number of examples on how to do this here.

As we outlined earlier, a paravirtualized network device consists of a pair of network devices. The first of these (the frontend) will reside in the guest domain while the second (the backend) will reside in the backend domain (typically Dom0).

- The frontend devices appear much like any other physical Ethernet NIC in the guest domain. Typically under Linux it is bound to the xen-netfront driver and creates a device called **ethN**. Under NetBSD and FreeBSD the frontend devices are named xennetN and xnN respectively.
- The backend device is typically named such that it contains both the guest domain ID and the index of the device. Under Linux such devices are by default named **vifDOMID.DEVID** while under NetBSD xvifDOMID.DEVID is used.
- 

- **Step 3:** To connect these virtual network devices to the network, a **vif** entry is added for each backend device in the respective domain configuration file.

This will look like this:

```
vif = ['mac=…, bridge=xenbr0' ]
```

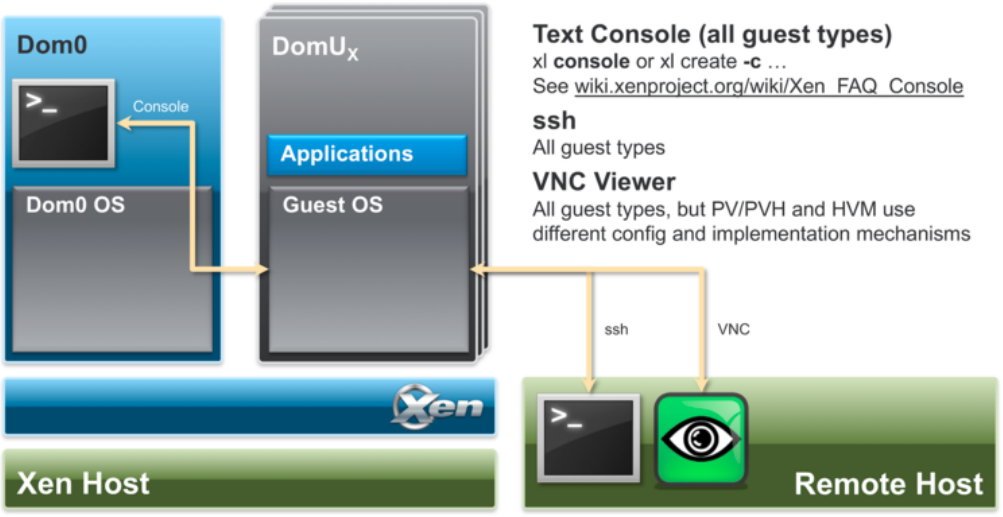
By default, most Xen toolstacks will select a random MAC address. Depending on the toolstack this will either be static for the entire life time of the guest (e.g. Libvirt, XAPI) or will change each time the guest is started (e.g. XL). For the latter, it is best practice to assign a defined MAC address, such that IP addresses remain static when used with a DHCP server.

Although the networking set-up in Xen may seem daunting, it is fairly straightforward. For more information see:

- xl-network-configuration(5)
- Xen Networking Guide
- Script to assign unique MAC addresses to backend devices
- Script to set up bridges on CentOS 7

Connecting to Guests: Console, ssh, VNC

The following diagram gives an overview of the different methods of connecting to Xen guests



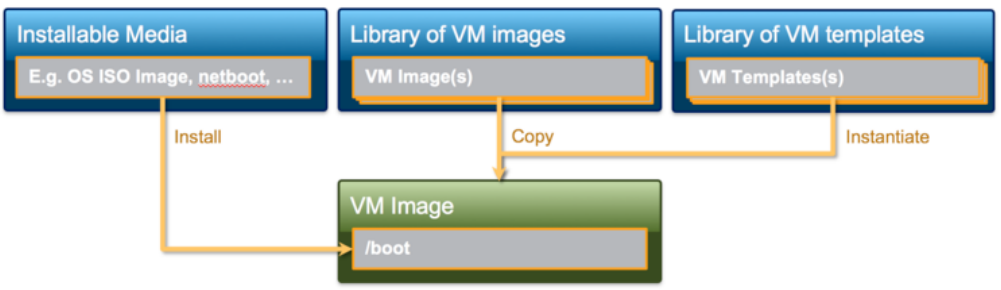
Also See

- FAQ: Console
- Set up a VNC console for PV guests
- Running and Connecting to VNC Servers on a Xen Guest
- Beginners Guide - includes VNC setup

Boot options for Xen

When a VM is created, it does not contain a bootable operating system. A user in principle has the following primary options

- Install the OS using normal operating system installation procedures: aka using an ISO based installation medium, network installation (e.g. PXE) or similar.
- Create clones of a previously created VM instance. Note that pre-built Guest VM Images for Xen are available from a number of sources. Clones can be used to set up a network of identical virtual machines, and they can also be distributed to other destinations. Some Xen project based products and distributions provide the capability to export and import VM images (e.g. any libvirt based Xen variant, XenServer and XCP-ng): xl does not provide such functionality: however, saving the master disk image and configuration file and creating a clone using a file copy of the disk image and configuration file (which will need to be adapted) is sufficient.
- Some Xen based products (e.g. XenServer and XCP-ng) as well as the libvirt toolstack provide a mechanism called templates to streamline the process of creating VM clones. Templates are instances of a virtual machine that are designed to be used as a source for cloning. You can create multiple clones from a template and make minor modifications to each clone using the provided template tooling.
- In addition, there are provisioning tools such as Xen tools



Also See

- Booting Overview
- PV Netboot
- PvGrub2
- PyGrub

xl

Minimal Config file

The following code snipped shows a minimal xl configuration file. Note that there are config file templates in /etc/xen

```
# Guest name and type, Memory Size and VCPUs
name = “myguestname”
type = “TYPE”
memory = MMM
vcpus = VVV

# Boot related information, unless type='hvm’ … one of the following
# See https://wiki.xenproject.org/wiki/Booting_Overview
# for an explanation

# Netboot/Direct Kernel Boot/PV GRUB
kernel = "/…/vmlinuz”
ramdisk = "/…/initrd.gz”
extra = …
# To use PVGrub (if installed)
firmware="pvgrub32|pvgrub64
# Boot from disk
bootloader=“pygrub”

# Disk specifications
disk = [' ']
# Network specifications
vif = [' ']
```

Common xl commands

VM control

- xl create [configfile] [OPTIONS]
- xl shutdown [OPTIONS] -a|domain-id
- xl destroy [OPTIONS] domain-id
- xl pause domain-id
- xl unpause domain-id Information

Information

- xl info [OPTIONS]
- xl list [OPTIONS] [domain-id ...]
- xl top

Debug

- xl dmesg [OPTIONS]
- xl -v … logs from /var/log/xen/xl-${DOMNAME}.log, /var/log/xen/qemu-dm-${DOMNAME}.log, …

Xen filesystem locations

- /etc/xen : scripts, config file examples, your config files
- /var/log/xen : log files
- /usr/lib64/xen/bin : xen binaries
- /usr/lib64/xen/boot : xen firmware and boot related binaries
- /boot : boot and install images

Getting Xen Project, Host and Guest Install

Choice of Control Domain (Dom0)

As stated earlier, the Xen Project hypervisor requires a kernel as control domain. Most Xen Project-enabled kernels are very similar from the perspective of the hypervisor itself. Choosing the right Dom0 for you comes down to:

- How familiar you are with a specific distro (e.g. packaging system, etc.)
- Xen Project Hypervisor version that ships with the distro
- Whether you can get commercial support (if you need it)

If you use XenServer or XCP-ng, you typically will not be interfacing much with Dom0 unless you are a power user.

Also See

- Dom 0 Kernels
- Getting Started

Getting Xen Project software

The Xen Project hypervisor is available as source distribution from XenProject.org. However, you can get recent binaries as packages from many Linux and Unix distributions, both open source and commercial.

Xen Project Source Distributions The Xen Project community delivers the hypervisor as a source distribution, following the delivery model of the Linux kernel. The software is released approximately once every 6-9 months, with several update releases per year containing security fixes and critical bug fixes. To build Xen Project software from source, you can either download a source release or you can fetch the source tree from the source repository. Each source release and the source tree contain a

README file in the root directory, with detailed build instructions for the hypervisor. The release notes for each release also contain build instructions and so does the Compiling Xen Project software page.

Xen Project software in Linux/Unix Distributions Most Linux and many Unix distributions contain built binaries of the Xen Project hypervisor that can be downloaded and installed through the native package management system. If your Linux/Unix distribution includes the hypervisor and a Xen Project-enabled kernel, we recommend to use them as you will benefit from ease of install, good integration with the distribution, support from the distribution, provision of security updates etc. Installing the hypervisor in a distribution typically requires the following basic steps: a) Install your favourite distribution, b) Install Xen Project package(s) or meta-package, c) check boot settings and d) reboot. After the reboot, your system will run your favourite Linux/Unix distribution as Control Domain on top of the hypervisor.

Host and Guest Install

The following documents

- Category:Host Install contains guides on how to install a control domain for various Linux/Unix distros
- Category:Guest Install contains guides on how to install a variety of guests for various Linux/Unix distros

This table contains a list of Xen Project resources for various Linux and Unix distributions.	[edit]

Distro	Main website	Description	Resources
Arch Linux	**archlinux.org**	**Arch Linux is a lightweight and flexible Linux® distribution that tries to "keep it simple".**	• **Arch as Xen Host** • **Arch as Xen Guest (PV)** • **All Arch_Linux pages**
Alpine Linux	**alpinelinux.org**	**A security-oriented, lightweight Linux distribution based on musl libc and busybox.**	• **Xen Dom0 on a HDD** • **Xen Dom0 on USB / SDD** • **Xen DomU install** • **Alpine Linux Xen LiveCD** • **All Alpine Linux wiki pages**
CentOS 5	**centos.org**	**CentOS is an Enterprise-class Linux Distribution derived from**	• **CentOS 5.x includes the Xen**

		sources freely provided to the public by a prominent North American Enterprise Linux vendor. CentOS conforms fully with the upstream vendor's redistribution policy and aims to be 100% binary compatible. (CentOS mainly changes packages to remove upstream vendor branding and artwork.) CentOS is free.	Project 3.4 Hypervisor, and can be used as a dom0 and domU out-of-the-box • CentOS / RHEL 6 as Xen Project Host • All CentOS wiki pages
CentOS 6	centos.org	CentOS is an Enterprise-class Linux Distribution derived from sources freely provided to the public by a prominent North American Enterprise Linux vendor. CentOS conforms fully with the upstream vendor's redistribution policy and aims to be 100% binary compatible. (CentOS mainly changes packages to remove upstream vendor branding and artwork.) CentOS is free.	CentOS 6.0 - 6.3 does not include Xen Project software, but you can get support from various sources. The following articles may be useful • CentOS / RHEL 6 as Xen Project Host • CentOS 6 as Xen Project Guest (32 bit) • CentOS 6 as Xen Project Guest (64 bit) • CentOS 6 as XCP Guest • All CentOS wiki pages CentOS 6.4+ does include Xen Project support and can be used as a dom0 and domU out-of-the-box, thanks to the Xen4CentOS project • Xen4CentOS • Xen4CentOS6 Release Notes • Xen4CentOS6 Quickstart Guide • All CentOS wiki pages Xen packages in CentOS

			6 and commercial support are also available from "Xen made easy!" • Xen packages from "Xen made easy!"
Debian	debian.org	The Debian project produces an entirely free operating system that empowers its users to be in control of the software running their computers.	• Debian as Xen Host • Debian Guest Installation Using Debian Installer • XCP toolstack on a Debian-based distribution • All Debian wiki pages
Fedora	fedoraproject.org	Fedora is a RPM-based distribution with a 6-month release cycle, and is the community-supported base of RHEL releases.	• Fedora 16 and newer provides full Xen support out of the box. • Fedora Host Installation • Fedora Test Days • All Fedora wiki pages
FreeBSD	freebsd.org	FreeBSD® is an advanced operating system for modern server, desktop, and embedded computer platforms.	• How to Install a FreeBSD domU on a Linux Host • All FreeBSD Wiki Pages
Finnix	finnix.org	Finnix is a sysadmin utility Linux LiveCD, and includes out-of-the-box Xen Project guest support.	• Finnix for VPS providers • All Finnix wiki pages

Gentoo Linux	gentoo.org	Gentoo Linux is a special flavor of Linux that can be automatically optimized and customized for just about any application or need. Extreme performance, configurability and a top-notch user and developer community are all hallmarks of the Gentoo experience.	• Gentoo Xen page • All Gentoo wiki pages
NetBSD	netbsd.org	NetBSD is a free, fast, secure, and highly portable Unix-like open source operating system.	• NetBSD Xen Overview • NetBSD Xen HowTo • Compiling Xen From Source on NetBSD • All NetBSD wiki pages
Oracle Linux	oracle.com	Oracle Corporation distributes Oracle Linux the Unbreakable Enterprise Kernel. Oracle states that the Unbreakable Enterprise Kernel is compatible with RHEL, Oracle middleware and 3rd-party RHEL-certified applications. Oracle Linux supports KVM, Xen Project, and Oracle VM Server for x86, which is based on Xen.	• All Oracle Linux wiki pages
openSuSE	opensuse.org	openSuSE is a free and Linux-based operating system for your PC, Laptop or Server.	• All OpenSuSE wiki pages
Red Hat Enterprise Linux (RHEL) 5.x	redhat.com	RHEL 5.x includes the Xen Project 3.4 Hypervisor as well as a Xen Project-enabled kernel, and can be used as a dom0 and domU	• CentOS / RHEL 6 as Xen Project Host • All RHEL wiki pages

Red Hat Enterprise Linux (RHEL) 6.x	**redhat.com**	**RHEL 6.x does not include the Xen Project Hypervisor. But, a Dom0 capable kernel, Xen Project hypervisor, and libvirt packages for use with RedHat Enterprise Linux 6 and its derivatives are available from either the Xen4CentOS project or the "Xen made easy!" effort.**	• **Xen4CentOS** • **Xen Project packages from "Xen made easy!"**
Ubuntu	**ubuntu.com**	**Fast, secure and stylishly simple, the Ubuntu operating system is used by 20 million people worldwide every day.**	• **Ubuntu Xen Project Wiki page (includes both host and guest installation)** • **Virtualization Docs** • **XCP Ubuntu Install** • **All Ubuntu wiki pages**

Getting Started Tutorial (running Xen within VirtualBox)

The following tutorial allows you to experiment with Xen running within VirtualBox.

- System Requirements and Setup
- Exercise Script
- Accompanying Presentation
- Images and Files to download (7.7G)

Getting Help!

The Xen Project community contains many helpful and friendly people. We are here for you. There are several ways to get help and keep on top of what is going on!

- Read News!
- Read Documentation!
- Contact other users, to ask the questions and discuss the hypervisor or other Xen Project-related projects

News Sources

- Xen Project Home page aggregates selected news stories and blog posts
- The Xen Project Blog covers technical and community related stories
- xen-announce mailing list is a low frequency mailing list for important announcements

Documentation

Documentation for projects hosted on XenProject.org is available on the Xen Project Wiki. Our wiki is active and community maintained. It contains a lot of useful information and uses categories extensively to make it easy to find information. You may also want to check:

- Category:Manual
- Category:HowTo
- Category:FAQ
- For more applicable categories, see the front page

Mailing Lists

Search Mailing Lists All XenProject.org mailing lists are archived using the MarkMail system at xen.markmail.org. Before you ask a question, it is worth checking whether somebody else has asked the question before

Main Mailing Lists XenProject.org maintains a number of mailing lists for users of the hypervisor and other projects. English is used by readers on this list.

- xen-users is the list for technical support and discussions for the Xen Project hypervisor. If you are not sure where your question belongs start here!

IRC

Internet Relay Chat (IRC) is a great way to connect with Xen Project community members in real time chat and for support.

- **#xen** is the channel for technical support and discussions for the Xen Project hypervisor. If you are not sure where your question belongs start here!
- Check out our IRC page if you are not familiar with IRC.

Other places

There are a number of other places, where you can get help on Xen Project software. For example:

- Xen Project Questions on serverfault
- XCP & XenServer Questions on serverfault
- Many Linux Distributions that support Xen Project software have a dedicated list - see the table in this link

Raising Bugs

If you find a bug, you can report bugs against the software. Before you raise a bug, please read Reporting Bugs!

Roadmaps, Release Cadence, Maintenance Releases

The Xen Project community releases the Xen Project Hypervisor with a release cadence of 6 months (in

June and December of each year). Roadmap information is tracked at Xen Roadmap. You can find information on the maintenance release cycle at Xen Project Maintenance Releases.

Also See

Installation

- Getting Started guides a new user through key decisions to be made
- Category:Host Install contains guides on how to install a control domain
- Category:Guest Install contains guides on how to install a variety of guests
- Category:Host Configuration contains documents related to bootloader, console and network configuration
- Guest VM Images provides pointers to various preinstalled guest images.

Release Information

- Category:Manual contains Xen Project manual documents
- Category:Release Notes contain Xen Project release notes
- Xen Release Features contains a matrix of features against Xen Project versions

Specialist Topics: Networking, Performance, Security, NUMA, VGA, ...

Specialized Xen Project topics:

- Category:Networking contains articles related to networking
- Category:NUMA contains all articles related to the running (or to improving the support for doing so) of the Xen Project Hypervisor on NUMA architectures
- Category:Performance contains documents, tuning instructions and benchmarks related to the performance of Xen Project software
- Category:Security contains documents related to Xen Project security
- Category:VGA contains documents related to VGA, VTd, GPY passthrough, etc.

FAQs, HowTos, ...

- Category:FAQ contains Xen Project FAQs
- Category:HowTo contains various HowTo's
- Category:Tutorial contains various Tutorials