

Introduction to the Open Source Xen Hypervisor

Jeanna Matthews and Zach Shepherd
Clarkson University

About Us

Zach Shepherd

- Undergraduate at Clarkson University
- Director of Clarkson Open Source Institute
- Worked with Xen in Infrastructure
- Virtualization Benchmarking Experience
- Technical Reviewer for *Running Xen*

Jeanna Matthews

- Associate professor at Clarkson University
- Research focus on operating systems, virtualization, computer networks and computer security
- Co-author of *Running Xen* and a variety of Xen related research publications

Overview (Four Sessions)

Session 1 Xen Introduction

Session 2 Xen Setup

Session 3 Xen Trouble-shooting and Advanced Topics

Session 4 Xen Security and Examples

Session 1

Xen Introduction

Outline of Xen Introduction

Virtualization Basics Ways to use Virtualization

Categories of Virtualization Technologies

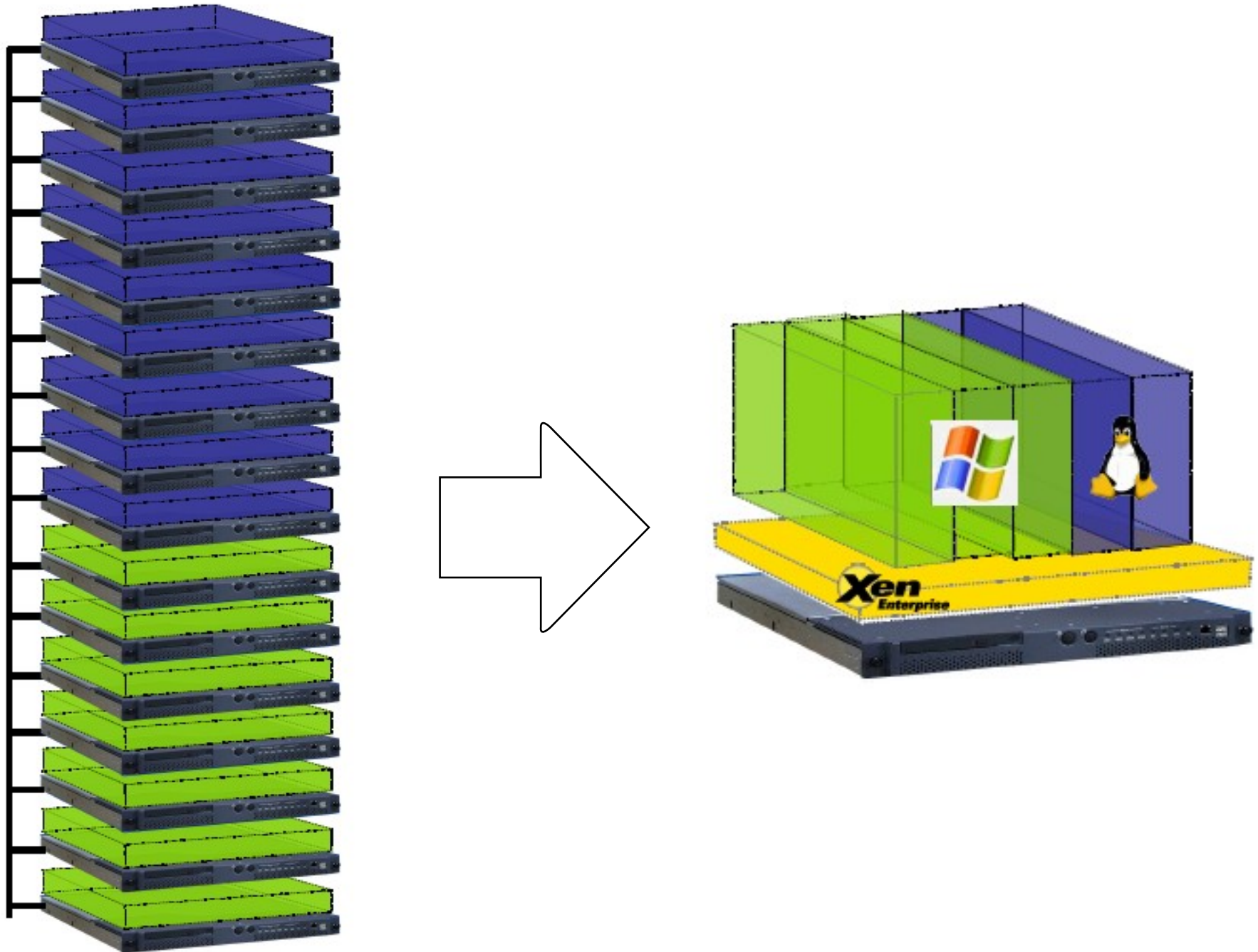
What Xen is

Reasons to use Xen

A look at Xen (terminology and architecture)

Introductory Demos

Hardware vs. Virtualization



Virtualization Basics

A physical machine runs a *program* to manage virtual machines (*Virtual Machine Monitor* or *hypervisor*)

On the physical machine, there are one or more *virtual machines* (*domains*) running.

A virtual machine is an encapsulated operating system which can run applications as a physical machine.

The primary virtual machine (*base machine*) is responsible for interacting with the hypervisor.

Other virtual machines are called *guests*.

Ways to use Virtualization

- **Fully utilize hardware resources** - consolidation of workloads onto a single machine, exploitation of multi-core machines
- **Running heterogeneous and conflicting environments** on the same machine - different operating systems, different libraries
- **Isolation** - separate workloads that have different requirement and/or to avoid attacks on one from compromising another
- **Manageability** - rapid deployment and provisioning, backup/disaster recovery, portability

Categories of Virtualization Technologies

Emulation

- Fully-emulate the underlying hardware architecture

Full-virtualization

- Simulate the base hardware architecture

Paravirtualization

- Abstract the base architecture

OS-level Virtualization

- Shared kernel (and architecture)
- Separate user spaces

What Xen is

Xen is a virtualization system supporting both paravirtualization and hardware-assistant full virtualization

Name from ne**X**t g**E**neration virtualization

"Xen and the Art of Virtualization"

Initially created by University of Cambridge Computer Laboratory

Open source (Licensed under GPL2)

Virtualization in Xen

Paravirtualization:

- Uses a modified Linux Kernel (e.g. Linux-2.6.18-xen)
- Guest loads dom0's pygrub or dom0's kernel
- Front-end and back-end virtual device model
- Cannot run unmodified OSes (e.g. Windows XP)
- Guest "knows" it's a VM and tells the hypervisor

Hardware-assisted full virtualization:

- Uses the same, normal, OS Kernel
- Guest contains grub and Kernel
- Normal device drivers
- Guest doesn't "know" it's a VM, so the hardware manages it

Reasons to use Xen

Paravirtualization (PV)

- High performance (claim to fame)
- High scalability
- Uses a *modified* Operating System

Hardware-assisted full virtualization (HVM)

- Leading hardware vendors to enhance virtualization in x86 architecture
- Uses an *unmodified* Operating System

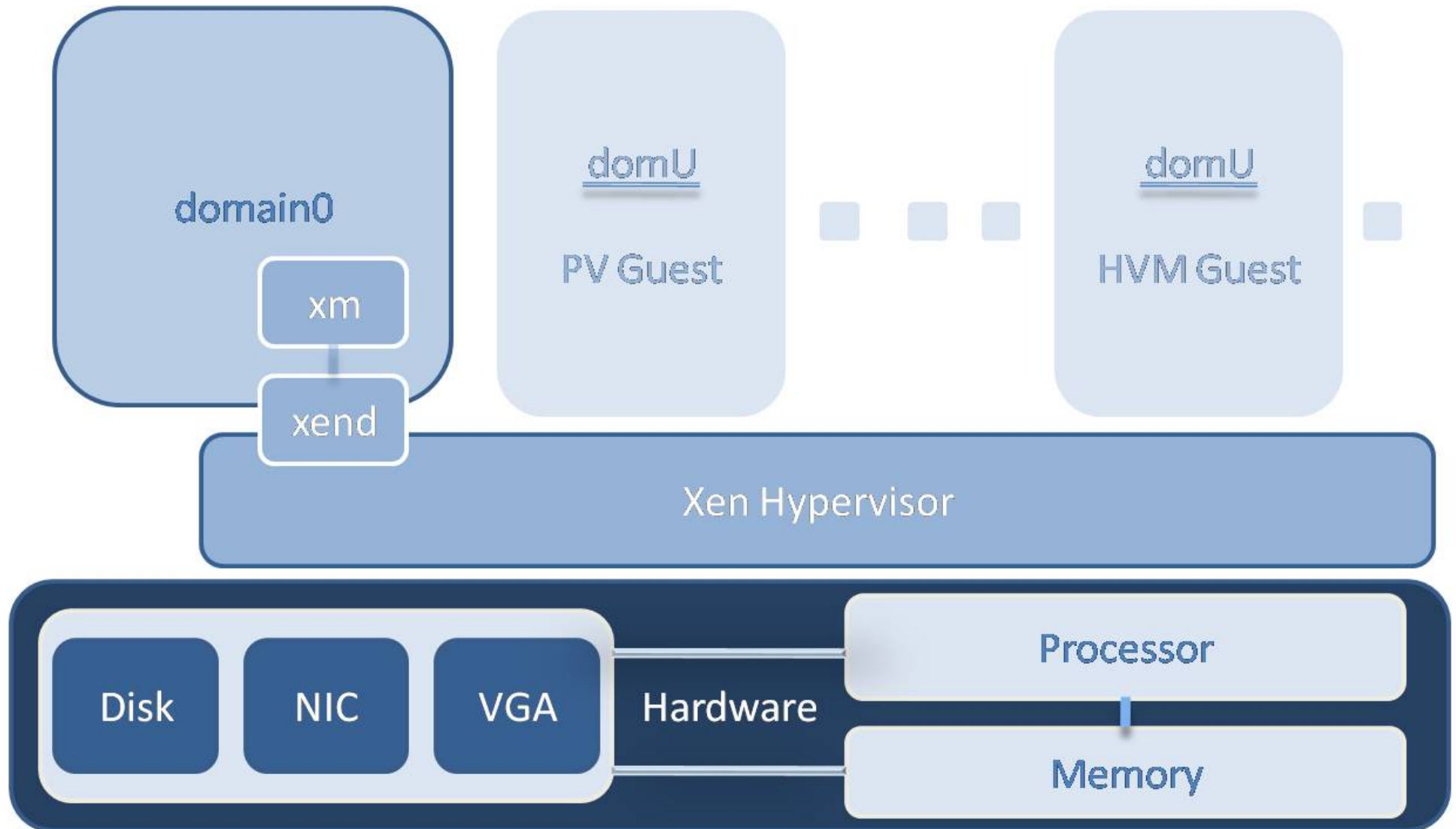
Xen is powered by a growing and active community and a diverse range of products and services

Xen offers high performance and secure architecture

Xen Terminology

General Virtualization System	Xen
Virtual Machine Monitor (VMM)	Hypervisor
	xend (Xen daemon)
	xend config.sxp
Privileged Guest	dom0 (domain0)
Virtual Machine (VM)	domU (guest domain)
VM configuration	Guest config. file

Xen Architecture



Xen: Hypervisor Role

Thin, privileged abstraction layer between the hardware and operating systems

Defines the virtual machine that guest domains see instead of physical hardware

- Grants portions of the full physical resources to each guest
- Exports simplified devices to guests
- Modifies hard-to-virtualize portions of x86 arch.
- Enforces isolation among guests

Xen: Domain0 Role

Creates and manages guest VMs

xm (Xen management tool)

A client application to send commands to xend

Interacts with the Xen hypervisor

xend (Xen daemon)

A daemon runs as a server to communicate with the hypervisor

Supplies device and I/O services

- Runs (backend) device drivers
- Provides domain storage

Xen System Start-up

Xen: Boot Process

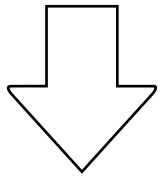
Normal Linux Boot Process

Xen Boot Process

Sample of Xen GRUB Configuration

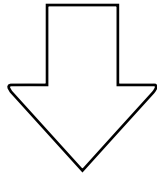
Normal Linux Boot Process

BIOS



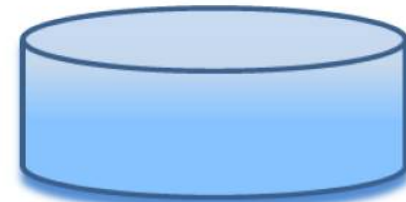
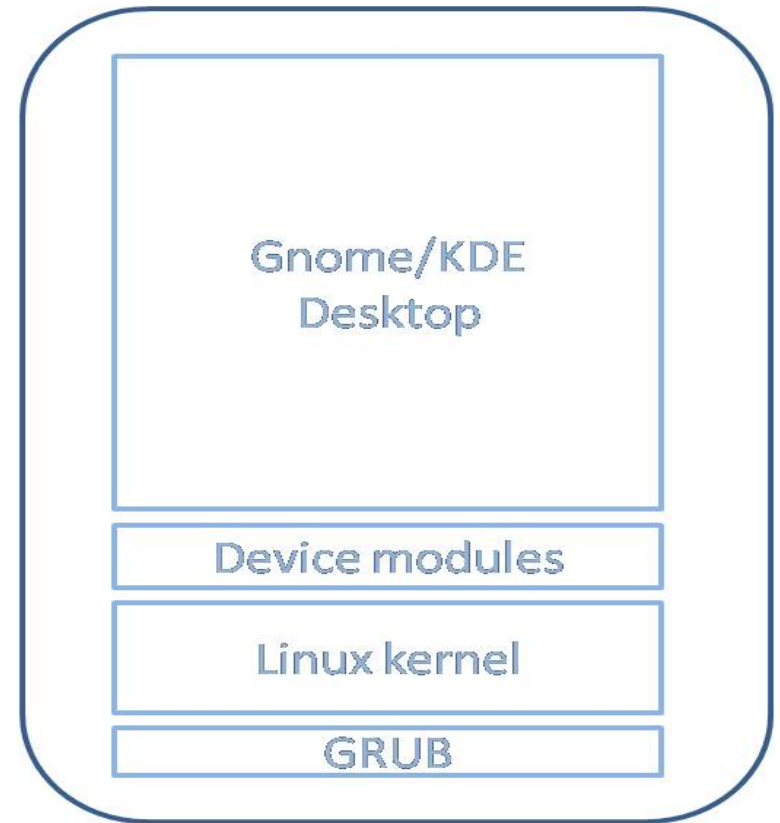
Master Boot Record (MBR)

GRUB



Kernel
Module

Linux

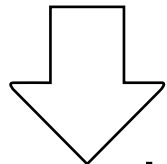


Sample of Linux GRUB Configuration

```
title Ubuntu 2.6.24-23
root (hd0,0)
kernel /boot/vmlinuz-2.6.24-23-generic
root=/dev/sda1
initrd /boot/initrd.img-2.6.24-23-generic
```

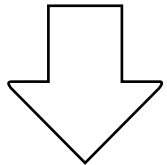
The Xen Boot Process

GRUB starts



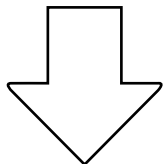
Kernel

Hypervisor starts



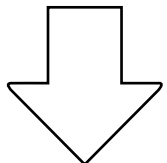
Module

Domain0 starts



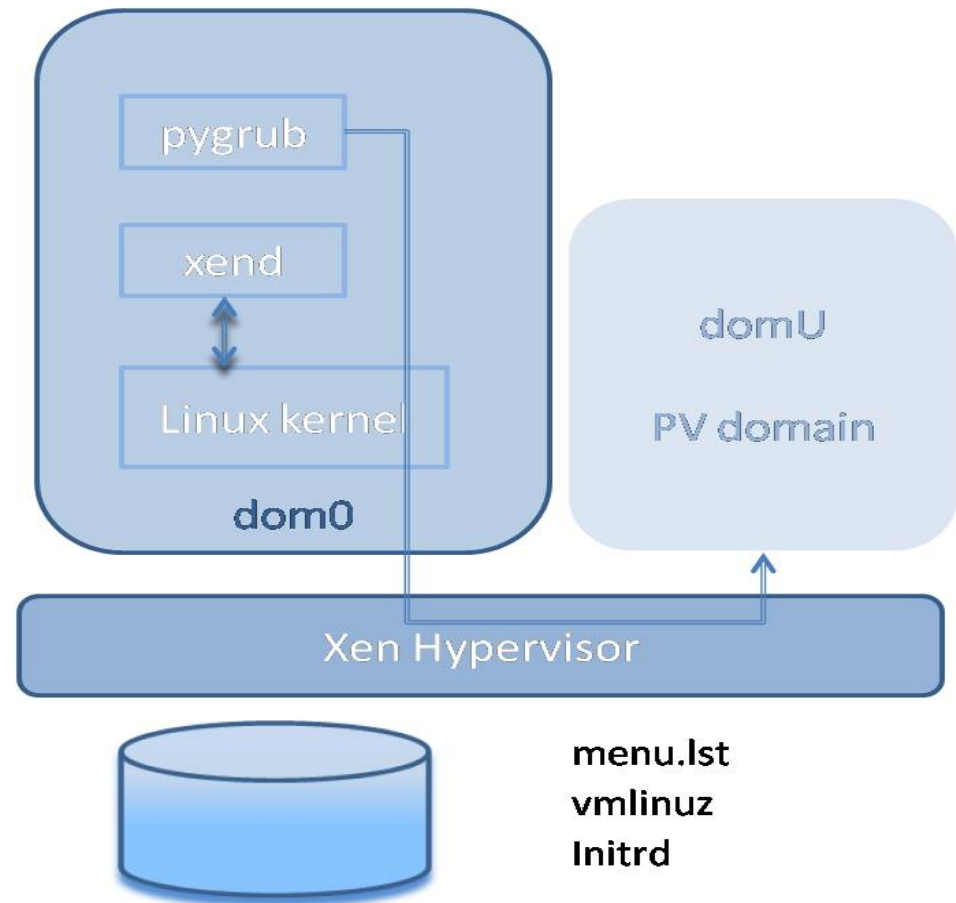
Daemon

Xend starts



xm

Guest domain starts



Sample of Xen GRUB Configuration

```
title Xen 3.4
root (hd0,0)
kernel /boot/xen-3.4.0.gz
module /boot/vmlinuz-2.6.18.8-xen root=/dev/sda1
module /boot/initrd.img-2.6.18.8-xen
```

Guest Access Methods

The simplest way - console:

```
xm console domU_name
```

A better way:

```
ssh you@xxx.xxx.xxx.xxx
```

Simple graphics:

set up a ssh server in domU and enable xorgforward

e.g. `ssh -x you@xxx.xxx.xxx.xxx`

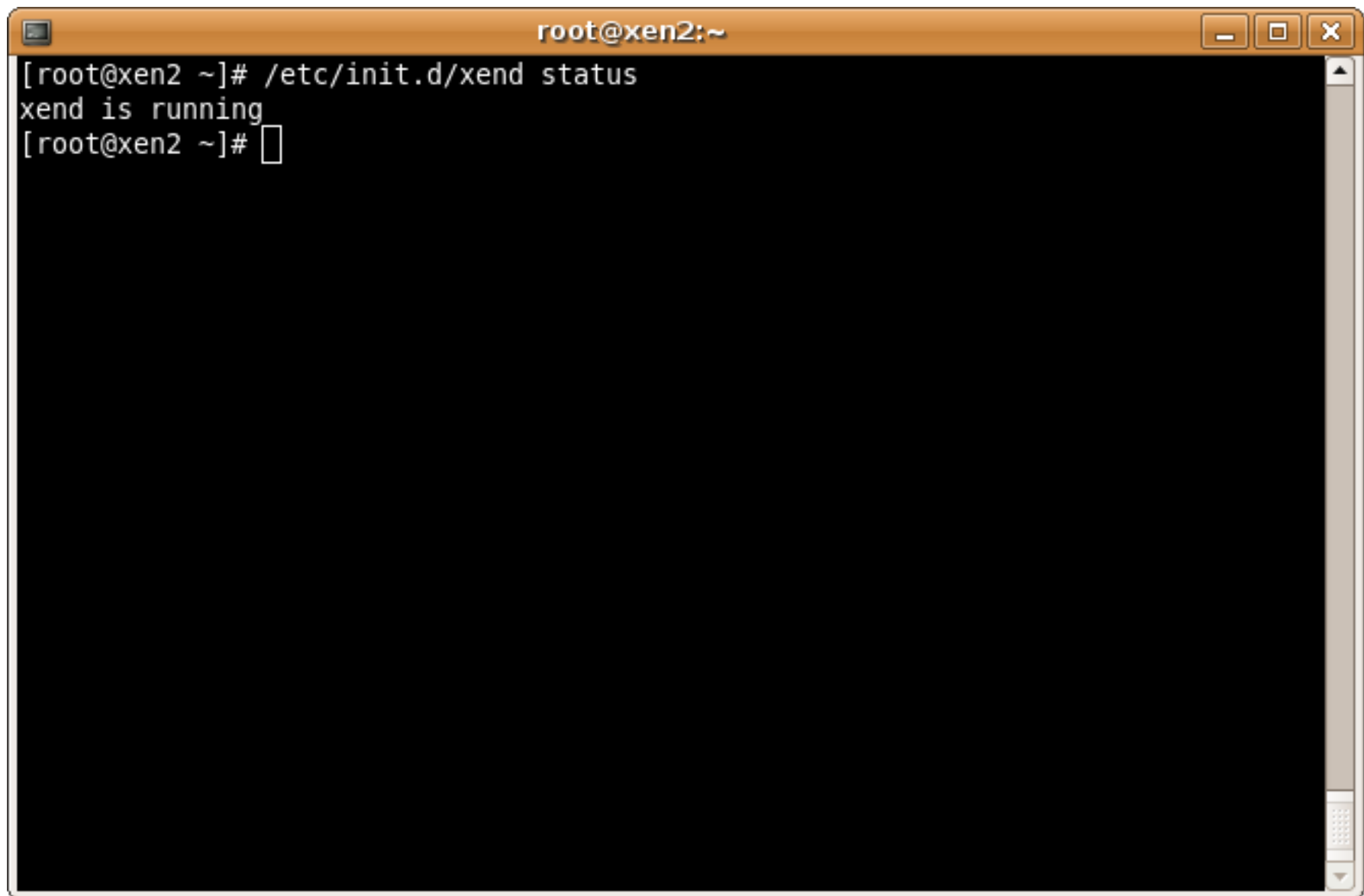
Better graphics - SDL (only for HVM) or VNC:

- Install `vncviewer` package
- Enable the `vnc` or `sdl` option in guest config file
- Connect to it (by localhost or ip)

Demos: Xen Introduction

Xen Basics

Using Pre-built Xen Guests

A terminal window with a title bar that reads "root@xen2:~". The window contains the following text: "[root@xen2 ~]# /etc/init.d/xend status", "xend is running", and "[root@xen2 ~]# " followed by a cursor. The window has standard Linux window controls (minimize, maximize, close) in the top right corner and a scrollbar on the right side.

```
root@xen2:~  
[root@xen2 ~]# /etc/init.d/xend status  
xend is running  
[root@xen2 ~]#
```

A status message showing xend running

root@xen2:~

```
[root@xen2 ~]# xm info
host                : xen2.cslabs.clarkson.edu
release             : 2.6.18-128.1.6.el5xen
version             : #1 SMP Wed Apr 1 09:53:14 EDT 2009
machine             : x86_64
nr_cpus             : 8
nr_nodes            : 1
sockets_per_node    : 2
cores_per_socket    : 4
threads_per_core    : 1
cpu_mhz             : 2327
hw_caps             : bfebfbff:20100800:00000000:00000140:040ce3bd:00000000:0
00000001
total_memory        : 8190
free_memory         : 0
node_to_cpu         : node0:0-7
xen_major           : 3
xen_minor           : 1
xen_extra           : .2-128.1.6.el5
xen_caps            : xen-3.0-x86_64 xen-3.0-x86_32p hvm-3.0-x86_32 hvm-3.0-x
86_32p hvm-3.0-x86_64
xen_pagesize        : 4096
platform_params     : virt_start=0xffff800000000000
xen_changeset       : unavailable
cc_compiler         : gcc version 4.1.2 20080704 (Red Hat 4.1.2-44)
cc_compile_by       : mockbuild
cc_compile_domain   : centos.org
cc_compile_date     : Wed Apr 1 09:00:24 EDT 2009
xend_config_format  : 2
[root@xen2 ~]#
```

A terminal window titled 'root@xen2:~' with standard window controls. The command '[root@xen2 ~]# xm list' has been executed. The output is a table with columns: Name, ID, Mem(MiB), VCPUs, State, and Time(s). The table contains one entry: 'Domain-0' with ID 0, 8026 MiB of memory, 8 VCPUs, and a state of 'r-----' (running), having been running for 2215.0 seconds. The prompt '[root@xen2 ~]#' is followed by a cursor.

```
[root@xen2 ~]# xm list
```

Name	ID	Mem(MiB)	VCPUs	State	Time(s)
Domain-0	0	8026	8	r-----	2215.0

```
[root@xen2 ~]#
```

xm list on a machine with no guests running

```
root@xen2:~  
[root@xen2 ~]# xm create /xen/confs/management -c  
Using config file "/xen/confs/management".  
Started domain management  
Bootdata ok (command line is root=/dev/xvda2 ro)  
Linux version 2.6.18-xen (root@berylum) (gcc version 4.1.3 20070929 (prerelease  
) (Ubuntu 4.1.2-16ubuntu2)) #1 SMP Fri Oct 26 21:31:32 EDT 2007  
BIOS-provided physical RAM map:  
Xen: 0000000000000000 - 0000000010800000 (usable)  
No mptable found.  
Built 1 zonelists. Total pages: 67584  
Kernel command line: root=/dev/xvda2 ro  
Initializing CPU#0  
PID hash table entries: 2048 (order: 11, 16384 bytes)  
Xen reported: 2327.496 MHz processor.  
Console: colour dummy device 80x25  
Dentry cache hash table entries: 65536 (order: 7, 524288 bytes)  
Inode-cache hash table entries: 32768 (order: 6, 262144 bytes)  
Software IO TLB disabled  
Memory: 235248k/270336k available (1996k kernel code, 26596k reserved, 867k data  
, 172k init)  
Calibrating delay using timer specific routine.. 4657.42 BogoMIPS (lpj=23287146)  
Security Framework v1.0.0 initialized  
Capability LSM initialized  
Mount-cache hash table entries: 256
```

Beginning of xm create

```
root@xen2:~  
/dev/xvda2: clean, 19213/460288 files, 178906/919721 blocks  
[ OK ]  
* Checking file systems...  
fsck 1.40.8 (13-Mar-2008)  
[ OK ]  
* Mounting local filesystems...  
[ OK ]  
* Activating swapfile swap...  
[ OK ]  
* Checking minimum space in /tmp...  
[ OK ]  
* Configuring network interfaces...  
[ OK ]  
* Starting system log daemon...  
[ OK ]  
* Starting kernel log daemon...  
[ OK ]  
* Starting OpenBSD Secure Shell server sshd  
[ OK ]  
* Starting MySQL database server mysqld  
[ OK ]  
* Checking for corrupt, not cleanly closed and upgrade needing tables.  
* Starting periodic command scheduler crond  
[ OK ]  
* Starting web server apache2  
apache2: Could not reliably determine the server's fully qualified domain name,  
using 127.0.0.1 for ServerName  
[ OK ]  
* Running local boot scripts (/etc/rc.local)  
[ OK ]  
Ubuntu 8.04 management tty1  
management login: █
```

End of xm create

```
root@xen2:~  
[root@xen2 ~]# xm list
```

Name	ID	Mem(MiB)	VCPUs	State	Time(s)
Domain-0	0	3560	8	r-----	2238.8
atp	1	511	1	-b----	52972.6
atp2	16	511	1	r-----	66456.6
auth	3	127	1	-b----	14.3
comm	5	255	1	-b----	90.6
docs	6	511	1	-b----	347.4
list	8	255	1	-b----	42.1
management	9	255	1	-b----	75.9
osp1	10	255	1	-b----	13.7
osp2	11	255	1	-b----	13.3
svn	17	255	1	-b----	38.8
web1	14	511	1	-b----	359.9

```
[root@xen2 ~]#
```

xm list showing several vms running

```
root@xen2:~  
xentop - 13:31:21 Xen 3.1.2-128.1.6.el5  
14 domains: 2 running, 12 blocked, 0 paused, 0 crashed, 0 dying, 0 shutdown  
Mem: 8387120k total, 8124404k used, 262716k free CPUs: 8 @ 2327MHz
```

NAME	STATE	CPU(sec)	CPU(%)	MEM(k)	MEM(%)	MAXMEM(k)	MAXMEM(%)	VCPUS	
NETS	NETTX(k)	NETRX(k)	VBDS	VBD 00	VBD RD	VBD WR	SSID		
web1	--b---		359	0.0	524068	6.2	524288	6.3	1
1	709138	49028	0	0	0	0	0		
docs	--b---		347	0.0	524044	6.2	524288	6.3	1
1	141559	22629	0	0	0	0	0		
Domain-0	-----r		2238	2.6	3645636	43.5	no limit	n/a	8
6	10943	25892	0	0	0	0	0		
comm	--b---		90	0.0	261944	3.1	262144	3.1	1
1	8758	18502	0	0	0	0	0		
svn	--b---		38	0.0	261932	3.1	262144	3.1	1
1	2591	10613	0	0	0	0	0		
atp	--b---		52972	0.0	524048	6.2	524288	6.3	1
1	1267	13007	0	0	0	0	0		
atp2	-----r		66425	74.7	524104	6.2	524288	6.3	1
1	437	12102	0	0	0	0	0		
management	--b---		75	0.0	261752	3.1	262144	3.1	1
2	318	13406	0	0	0	0	0		
auth	--b---		14	0.0	130740	1.6	131072	1.6	1
2	102	13780	0	0	0	0	0		

Delay Networks vBds VCPUs Repeat header Sort order Quit

xentop sorted by network transfer

Pre-built Images

Sources:

- stacklet.com
- rpath.org
- jumpbox.com

Advantages:

- Simple to download and extract the image
- Available with different distributions OSes and pre-installed applications
- Saves on installation and setup time.

Note:

- Check kernel version to match `vmlinz*` in `/boot`
- Check image path to match current directory

Session 2

Xen Setup

Outline of Xen Setup

Installing Xen From Package From Source

Configurations

Hypervisor Configuration

Guest Configuration

- General Guest Configuration
- PV-specific Configuration
- HVM-specific Configuration

Guest Management Tools

Convirt

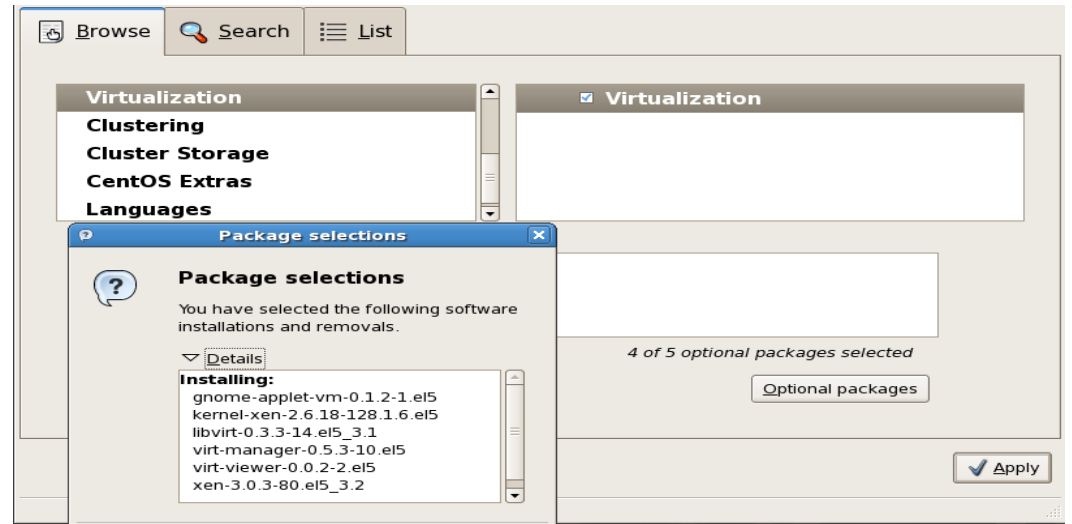
Zentific

Virt-manager

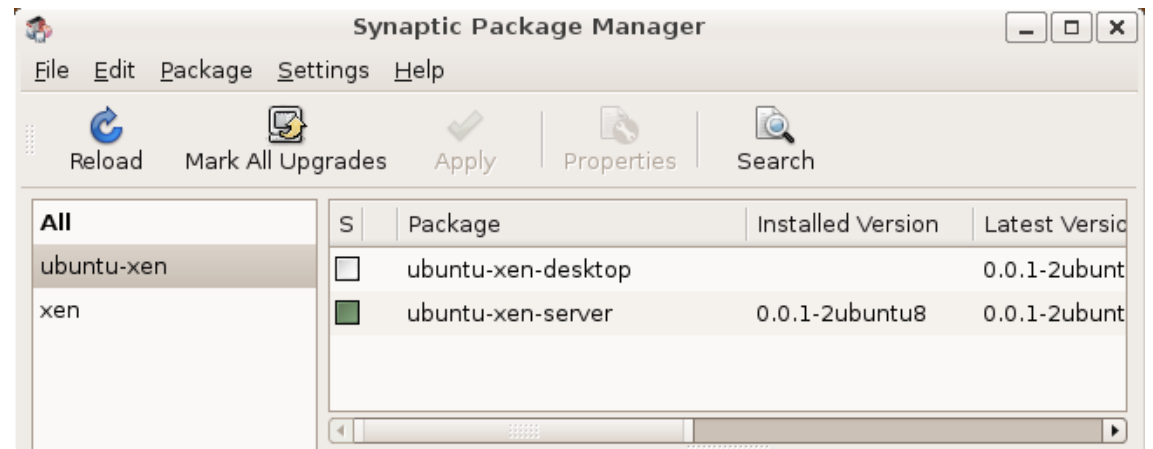
Guest Access Methods

Install Xen from a Package

(1) root:~>yum install xen



(2) root:~>apt-get install ubuntu-xen-server



Installing Xen from a Package (cont.)

OpenSUSE - Install with YaST

<http://www.susegeek.com/general/how-to-install-configure-xen-virtualization-in-opensuse-110/>

Gentoo - Install with portage

<http://www.gentoo.org/doc/en/xen-guide.xml>

OpenSolaris - default/xVM packages

<http://deepenintocs.blogspot.com/2008/05/solaris-xvm-xen-on-solaris.html>

NetBSD - Xen 3.x package support as of BSD4.0

<http://www.netbsd.org/ports/xen/howto.html>

Installing Xen from Source

Reasons to use the latest Xen version:

- Performance Optimization
- Security and Bug Fixes
- Cutting-Edge Features
- Ability to Patch/Customize

Xen source code is maintained by mercurial

<http://www.xen.org/download/index.html>

Example: Xen from Source

- Install required packages

- (e.g. `apt-get install gawk libssl-dev libx11-dev gettext libncurses-dev build-essential python-dev texlive-latex-base transfig tetex-extra bcc bin86 pciutils-dev git-core curl texinfo libc6-dev mercurial bridge-utils graphviz`)

- Download the latest Xen release

- (e.g. `hg clone http://xenbits.xen.org/xen-3.3-testing.hg`)

- Compile and Install

- (e.g. `make world; make install`)

- Create a ramdisk

- (e.g. `depmod 2.6.18.8-xen;mkinitramfs -o /boot/initrd.img-2.6.18.8-xen 2.6.18.8-xen`)

- Configure the bootloader

- (e.g. `update-grub`)

Configurations

Xend Configuration

Guest Configuration

- General Guest Configuration

- Kernel Configuration

 - PV-specific Configuration

 - HVM-specific Configuration

- Network Configuration

- Storage Configuration

Common Xend Configuration Options

Xen daemon's configuration in `/etc/xen/xend-config.sxp`

- Configure Xen to listen for remote connections
 - e.g. http
- Constrain dom0 CPU and memory resources
 - Set maximum/minimum resources for the dom0
- Set up the virtual network
 - Bridging, Routing, or NAT
- Configure live migration network parameters
 - Live migration configuration

Sample Xend Configuration

```
$(logfile /var/log/xen/xend.log)  
$(loglevel DEBUG)
```

```
$(xend-http-server no)  
(xend-unix-server yes)  
(xend-unix-path /var/lib/xend/xend-socket)
```

```
(network-script network-bridge)  
(vif-script vif-bridge)
```

```
(dom0-min-mem 256)  
(dom0-cpus 0)
```

```
$(xend-relocation-server no)  
$(xend-relocation-port 8002)
```

```
$(vnc-listen '127.0.0.1')  
(vncpasswd ")
```

Guest Configurations

Guest Configuration Option Categories:

- General
- Kernel (different for PV and HVM)
- Network
- Storage

Recap:

PV Guest (Paravirtualized)

HVM Guest (Hardware-based fully virtualized)

General Guest Configuration Options

(For both PV and HVM guests)

`name`

- The name of the guest
- (defaults to configuration filename)

`vcpus`

- The number of virtual CPUs
- (defaults to 1)

`memory`

- The amount of memory (in MB)
- (defaults to 128)

PV Kernel Configuration Options

`kernel`

- The location of the xen-modified kernel in dom0

`ramdisk`

- The location of the initial RAM disk image in dom0

`or`

`bootloader`

- The location of the bootloader (e.g. pygrub)

PV Kernel Configuration Options (cont.)

`root`

- The partition to use as root inside the guest

`extra`

- The parameters appended to the kernel command line (refer to a normal Linux boot process)

`vfb`

- Virtual framebuffer for PV guest to use vnc instead of console

Sample PV Guest Configuration

```
vcpus = 1
memory = 64
kernel = "/boot/vmlinuz-2.6.18.8-xen"
ramdisk = "/boot/initrd.img-2.6.18.8-xen"
vif = [ ' ' ]
disk = [ 'phy:hda1,xvda1,w' ]
root = "/dev/xvda1"
vfb = ['type=vnc,vncunused=1']
extra = 'xencons=tty'
```

HVM Kernel Configuration Options

`kernel`

- The location of the kernel

`builder`

- The domain build function ("`hvm`" for an unmodified kernel)

`device_model`

- The location of the device emulation tool (e.g. "`qemu_dm`")
to emulate the hardware

`boot`

- Boot order (floppy, CD-ROM, hard drive)

`vnc`

- Enable vnc utility for the guest to display

Sample HVM Guest Configuration

```
vcpus = 1
memory = 512
kernel = "/usr/lib64/xen/boot/hvmloader"
builder = "hvm"
device_model = "/usr/lib64/xen/bin/qemu-dm"
boot = "cd"
disk = [ 'tap:aio:/xen/images/hvm.disk,
          ioemu:hda,w',
          'phy:/dev/cdrom, ioemu:hdc:cdrom,r'
        ]
vif = [ 'type=ioemu, bridge=eth0' ]
vnc = 1    (or sdl = 1)
```


Installing HVM guests from CD

- Allocate Image for the VM
 - (we'll get to this in a bit)
- Create HVM Config File with install CD as first boot device
- Boot the guest
 - (e.g. `xm create /path/to/guest.cfg`)
- Follow normal installation process
- Edit the HVM Config file to remove the CD

Network Configurations

(for both PV and HVM guests)

Network Configuration Process

Guest Network Configuration Options

Xen Virtual Network Modes

- Bridging
- Routing
- NAT

Network Configuration: Process

1. Set up Xen virtual network in dom0

- Enable `network-script` and `vif-script` options in `xend` config file to specify the network mode
- Restart `xend`

(e.g. `/etc/init.d/xend restart`)

2. Configure domU's network interface to Xen virtual network

- Specify the network parameters in guest config file
- Boot the guest and configure the guest network inside as normal process

Note: The guest's network configuration should be the same as described in guest's xen configuration file

Network Configurations Options

Array of virtual interface network parameters
specify 'MAC Address, IP Address,' for each interface

Anything left blank is auto assigned

Examples

```
vif = [ ' ' ]
```

```
vif = [ 'mac=00:16:3e:36:a1:e9,  
        ip=192.168.1.25, bridge=xenbr0' ]
```

(bind a domU virtual interface to a specified dom0 interface)

Network Modes

Bridging mode

Guest domains are (**transparently**) on the same network as dom0

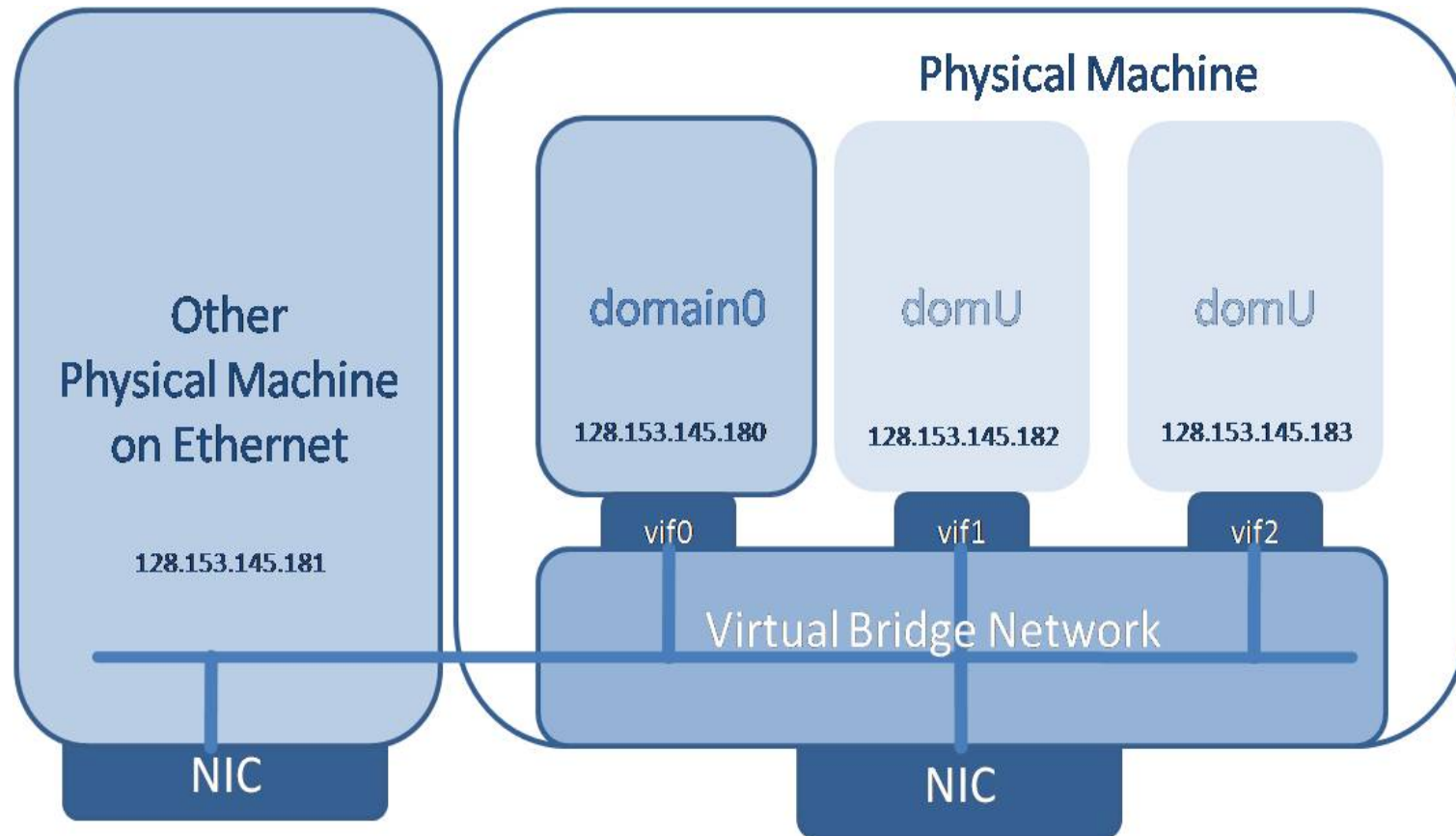
Routing mode

Guest domains **sit behind** dom0. Packets are relayed to the network by dom0

NAT mode

Guest domains **hide behind** dom0 using dom0's IP for external traffic

Bridging Mode

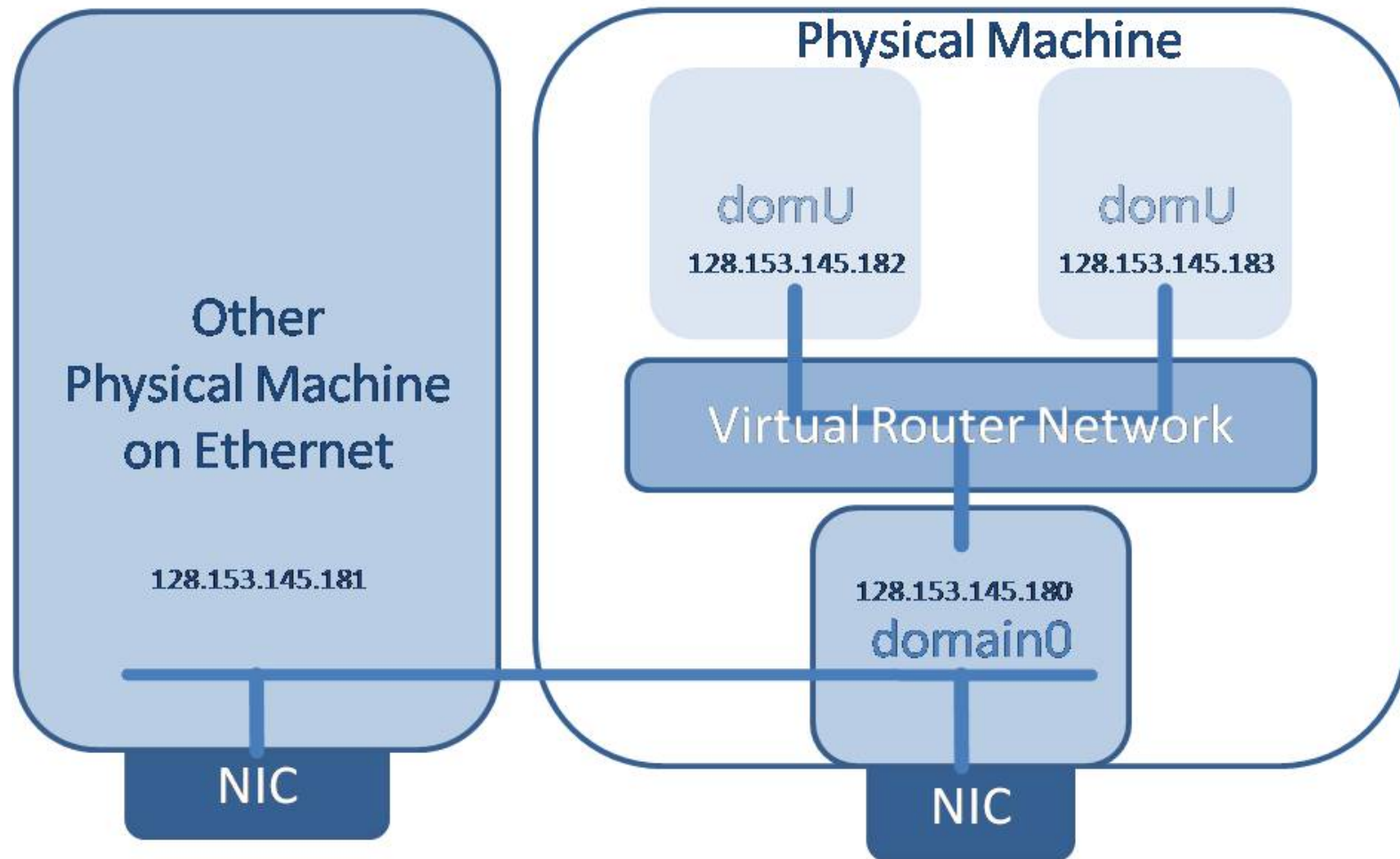


DomUs are (transparently) on the same network as dom0

Bridging Mode Configuration

- Default network mode for Xen
- No need to do anything if guest has network interface
 - All you need is a `vif` statement in the domU's config file
- Uses bridge-utils to set up a software bridge in dom0
 - Automatically set up, no need to configure

Routing Mode

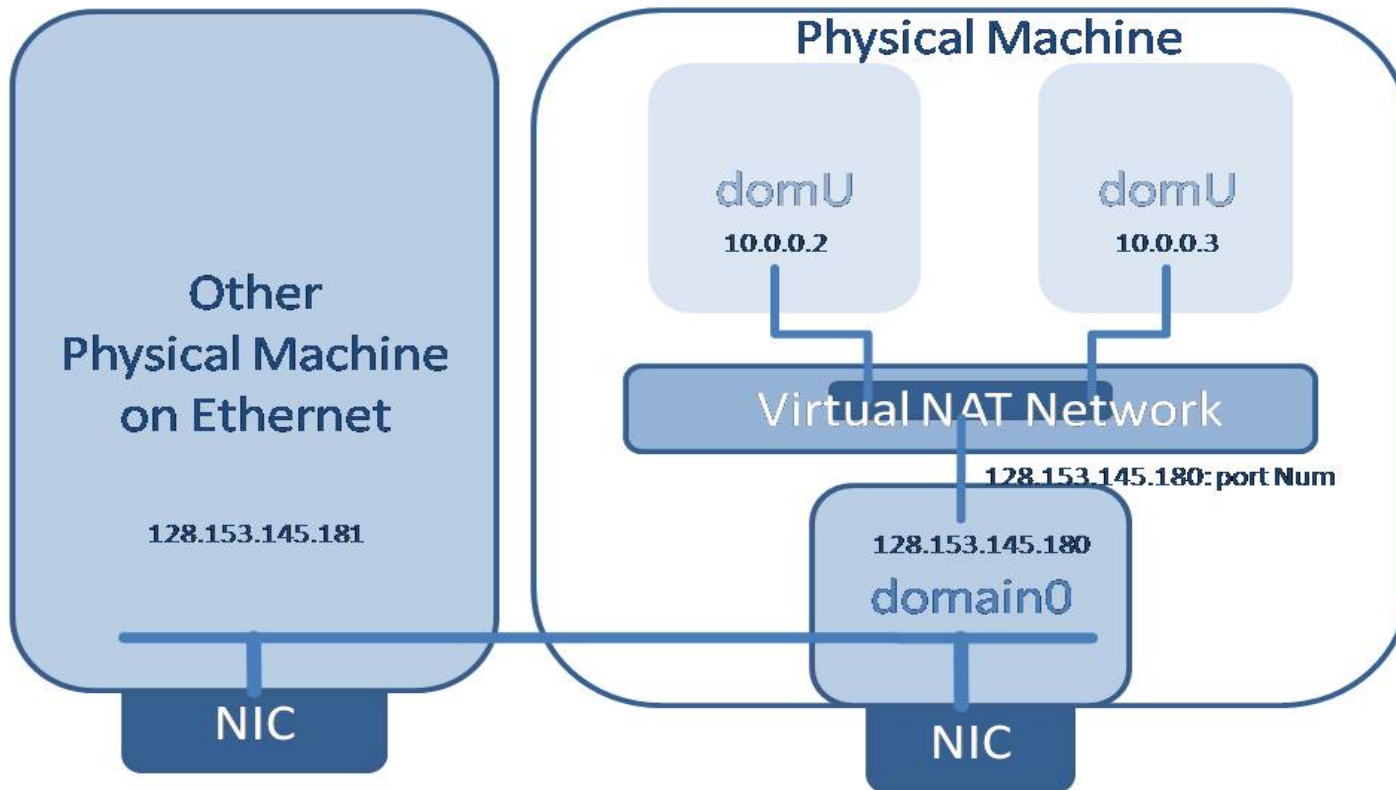


DomUs sit behind dom0. Packets are relayed to the network

Routing Mode Configuration

1. Modify `xend` configuration
 2. `(network-script network-route)`
`(vif-script vif-route)`
 3. Set guest's *gateway* to *dom0's external IP* in addition to the vif statement
`GATEWAY="128.153.144.204"` (ie. Dom0's IP)
`NETMASK="255.255.255.0"` (based on size of virtual subnet)
3. Xen uses `iptables` in `dom0` to set up the software router A
set up based on `xend` configuration

NAT Mode



DomUs hide behind dom0 using dom0's IP for external traffic

NAT Mode Configuration

1. Modify `xend` configuration
2. `(network-script network-nat)`
`(vif-script vif-nat)`
- 3.
4. Set guest's *gateway* to *dom0's internal IP* in addition to the vif statement
5. `GATEWAY="10.0.0.1"`
6. (ie. Dom0's internal IP address on the private network segment)
`NETMASK="255.0.0.0"`
(based on size of the private network segment)
3. Use iptables in dom0 to do the NAT translation
Again done automatically based on the `xend` configuration

Storage Configuration

(for both PV and HVM guests)

Local Storage

- Raw File
- Partition
- Partitioned File

Network Storage

- AoE
- iSCSI
- NFS
- NBD
- DRDB

Storage Configuration Options

Array of disk specifications

```
' real dev in dom0, virtual dev in domU, Access (r,  
w) '
```

SCSI (sd) and IDE(hd) examples

```
disk= [ 'phy:sda, sda, w',  
        'phy:/dev/rom, cdrom:hdc, r' ]  
disk= [ 'tap:aio:hdb1, hdb1, w',  
        'phy:/dev/LV/disk1, sda1, w' ]
```

Xen Virutal Device examples

```
disk= [ 'tap:aio:hdb1, xvdb1, w',  
        'phy:/dev/LV/disk1, xvda1, w' ]
```

Local Storage

Raw File

- Use a filesystem within a single file
- Takes advantage of loopback devices

Partition

- Use a partition on a logical partition
- Can be physical partition or LVM volume

Partitioned File

- Less common
- Treats a raw file as a disk (instead of single partition)

Local Storage: Raw File for PV

- Allocate storage

- (e.g. `dd if=/dev/zero of=/path/to/image.img bs=1024k count=1024`)

- Format the storage

- (e.g. `mkfs.ext3 -F /path/to/image.img`)

- Mount the storage

- (e.g. `mkdir /mnt/tmp; mount -o loop /path/to/new/image.img /mnt/tmp`)

- Install the operating system

- (e.g. `debootstrap hardy /mnt/tmp or cp -a /* /mnt/tmp`)

- Note: needs to have the right drivers

- Create the guest configuration file

- (as in previous examples)

- Modify various files on the guest and unmount the storage

- (e.g. `/etc/fstab`, `/etc/hostname`, `/etc/ifconfig`)

Local Storage: Raw File for HVM

- Allocate storage

- (e.g. `dd if=/dev/zero of=/path/to/image.img bs=1024k count=1024`)

- Create the guest configuration file

- (as in previous examples)

- Install the operating system

- (see section on installing HVM guests from CD)

Network Storage

ATA over Ethernet (AoE)

- Export block devices over the network
- Lightweight *Ethernet layer* protocol
- No built-in security

Internet Small Computer System Interface (iSCSI)

- Export block devices over the network
- Network layer protocol
- Scales with network bandwidth
- Client and user-level security

Network File System (NFS)

- Exports *file system* over the network
- Network layer protocol
- Known performance issues as root file system

Network Storage (con't)

Network Block Device (NBD)

- Exports block devices over the network
- Network layer protocol
- Scales with network bandwidth
- Not recommended as root file system

Distributed Replicated Block Device (DRBD)

- Export and share block devices over the network
- Integration with Heatbeat
- No additional storage server necessary
- Limited to two nodes (without stacking)

Network Storage Example: AoE

- Install required packages
 - Install `Vblade` on the storage server
 - Install `aoe-tools` and the `aoe` module in the domain0
- Export a guest image from the storage server
 - `"vbladed 1 1 eth0 /dev/..."` (for partitions)
 - `"vbladed 1 1 eth0 /path/to/image.img"` (for files)
- Point the guest configuration to the image
 - `"disk = ['phy:etherd/e1.1,xvda1,w']"`
- Notes
 - Remember: AoE provides **no** security
 - Never use the same shelf/slot for two images

Network Storage Example: DRBD

- Install required packages
 - Ubuntu/Debian: `drbd8-utils` and `drbd8-module`
 - Redhat/CentOS: `drbd` and `drbd-km`
- Configure DRBD
 - Mostly beyond the scope of this presentation
 - Disable `sendpage` in `/etc/modprobe.d/drbd.conf`
(`options drbd disable_sendpage=1`)
- Point the guest configuration to the image
 - `"disk = ['drbd:resource,xvda,w']"`
- Documentation:
 - <http://www.drbd.org/users-guide/ch-xen.html>

Guest Management Tools

Goals:

- Create guest images
- Manipulate guest domains
- Automated generate guest config files
- Monitor the resource usage and allocations

Popular open source tools:

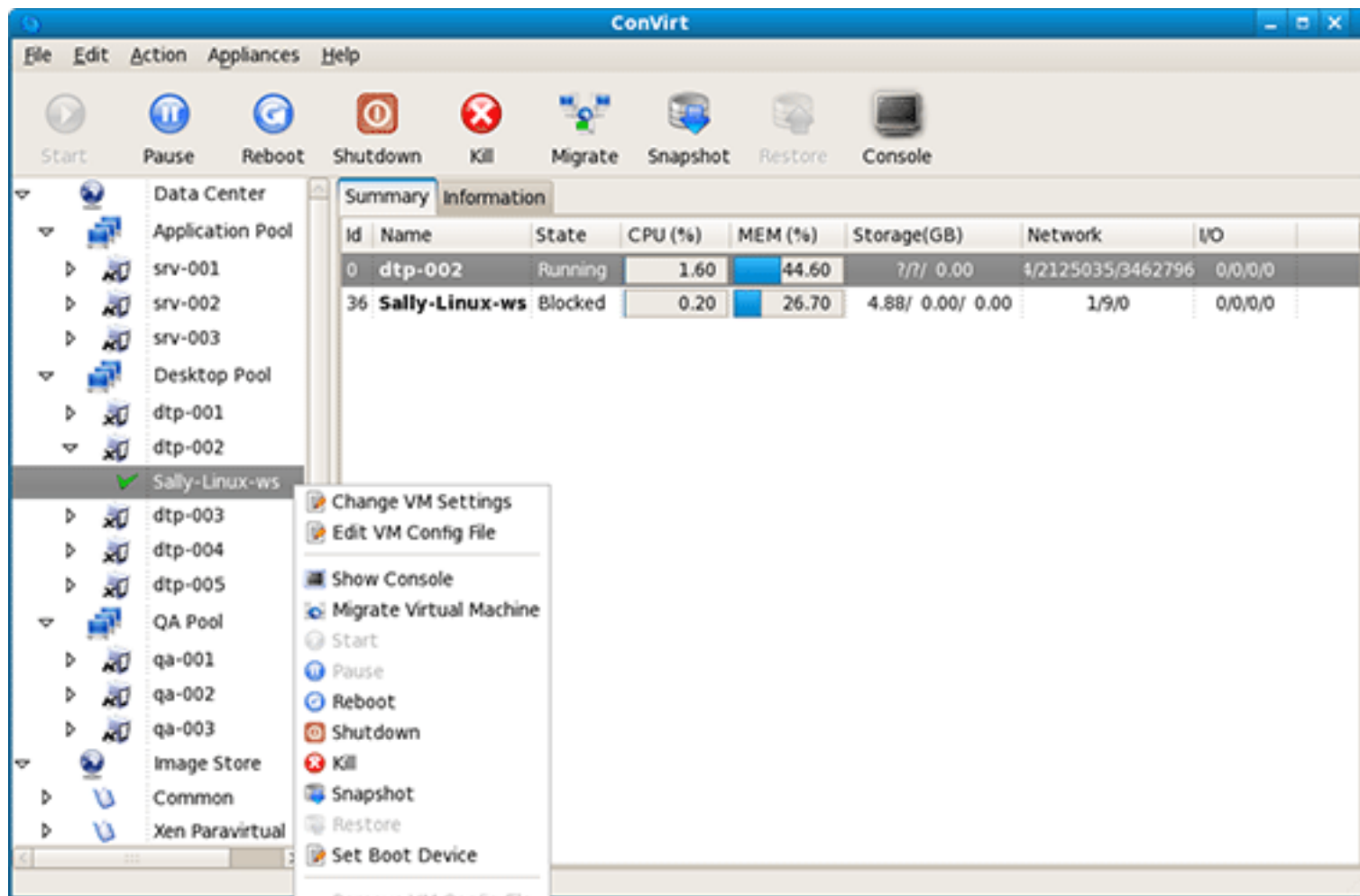
- Convirt
 - Third-party product and support
 - Enterprise-level tool
- Zentific
 - Third-party
 - Web-based tool
- Virt-manager
 - Red-Hat project
 - Desktop tool

Convirt

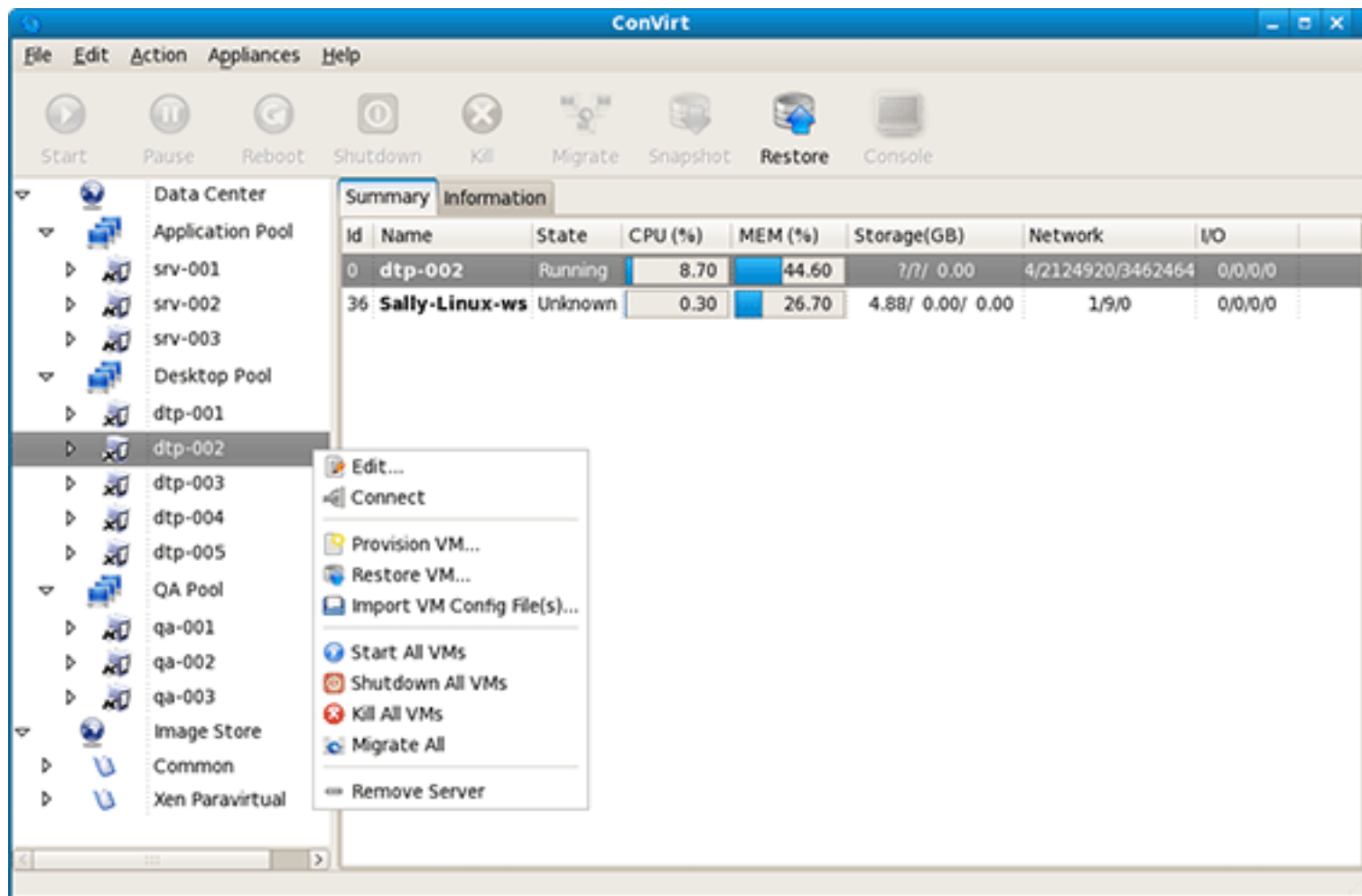
Designed for enterprise-level full datacenter management

Allows for managing the complete lifecycle of Xen (and KVM) guests and hosts

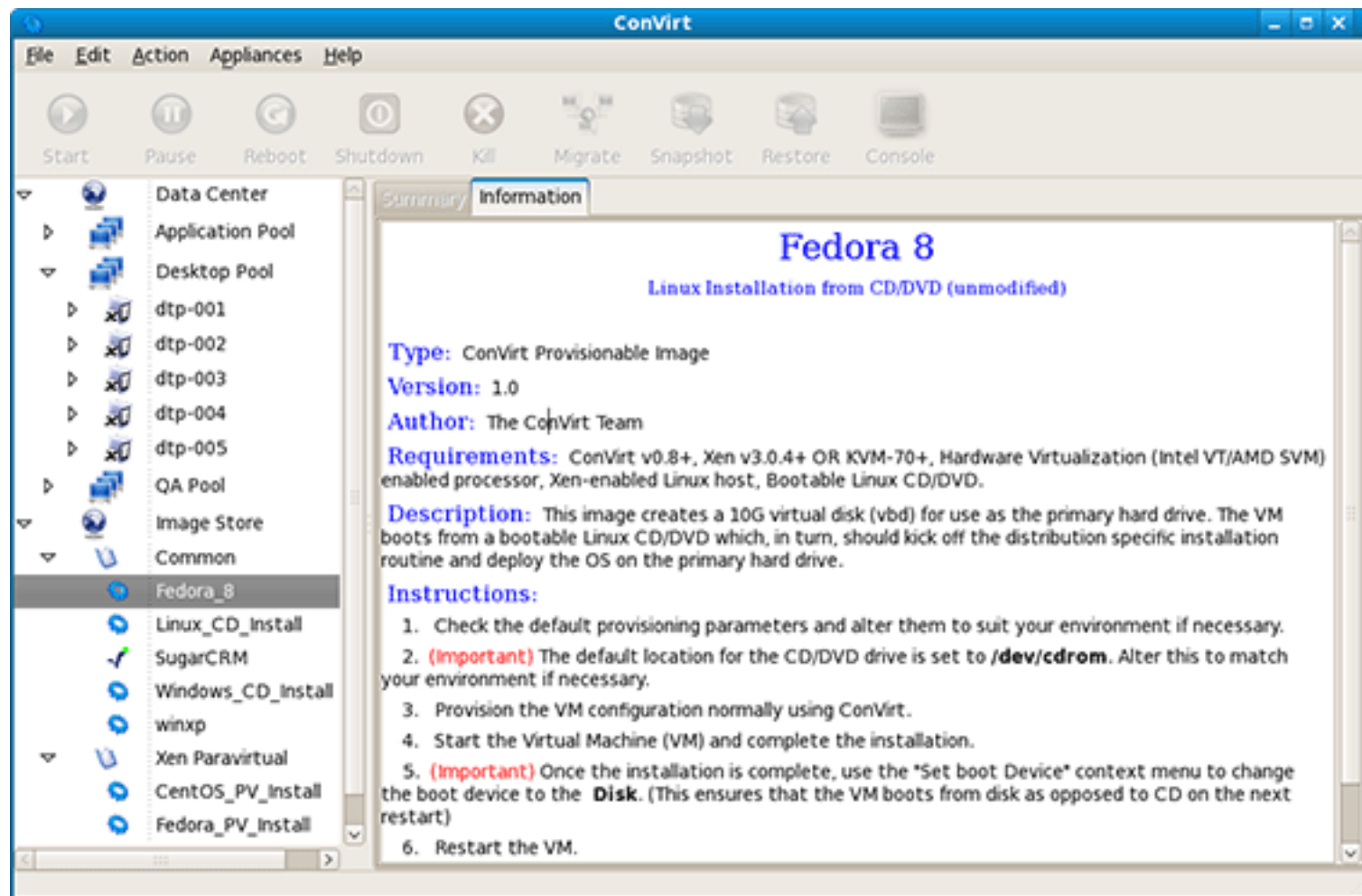
Open Source with commercial support



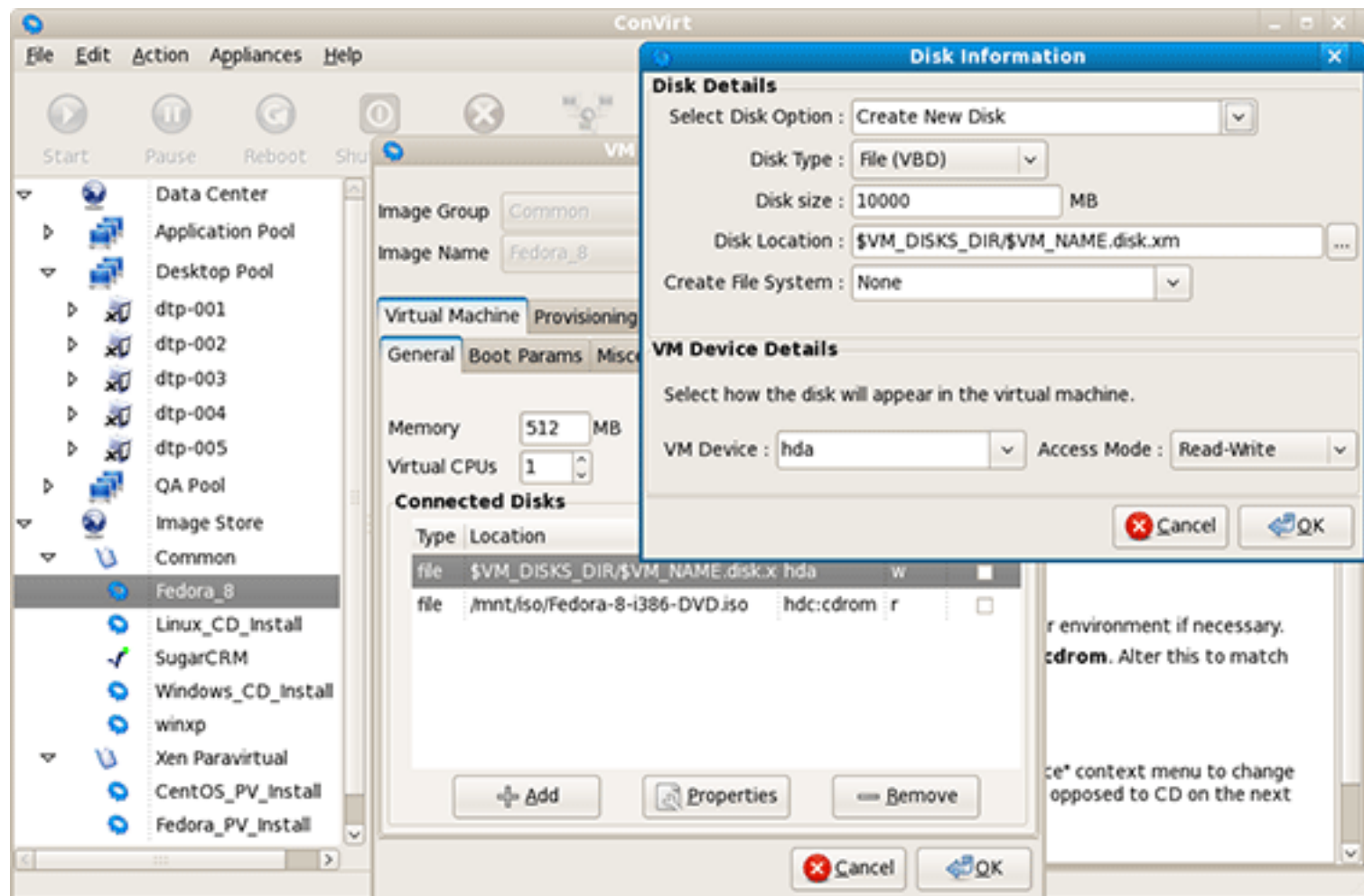
Manage VMs



Manage domain0s



Provision based on templates

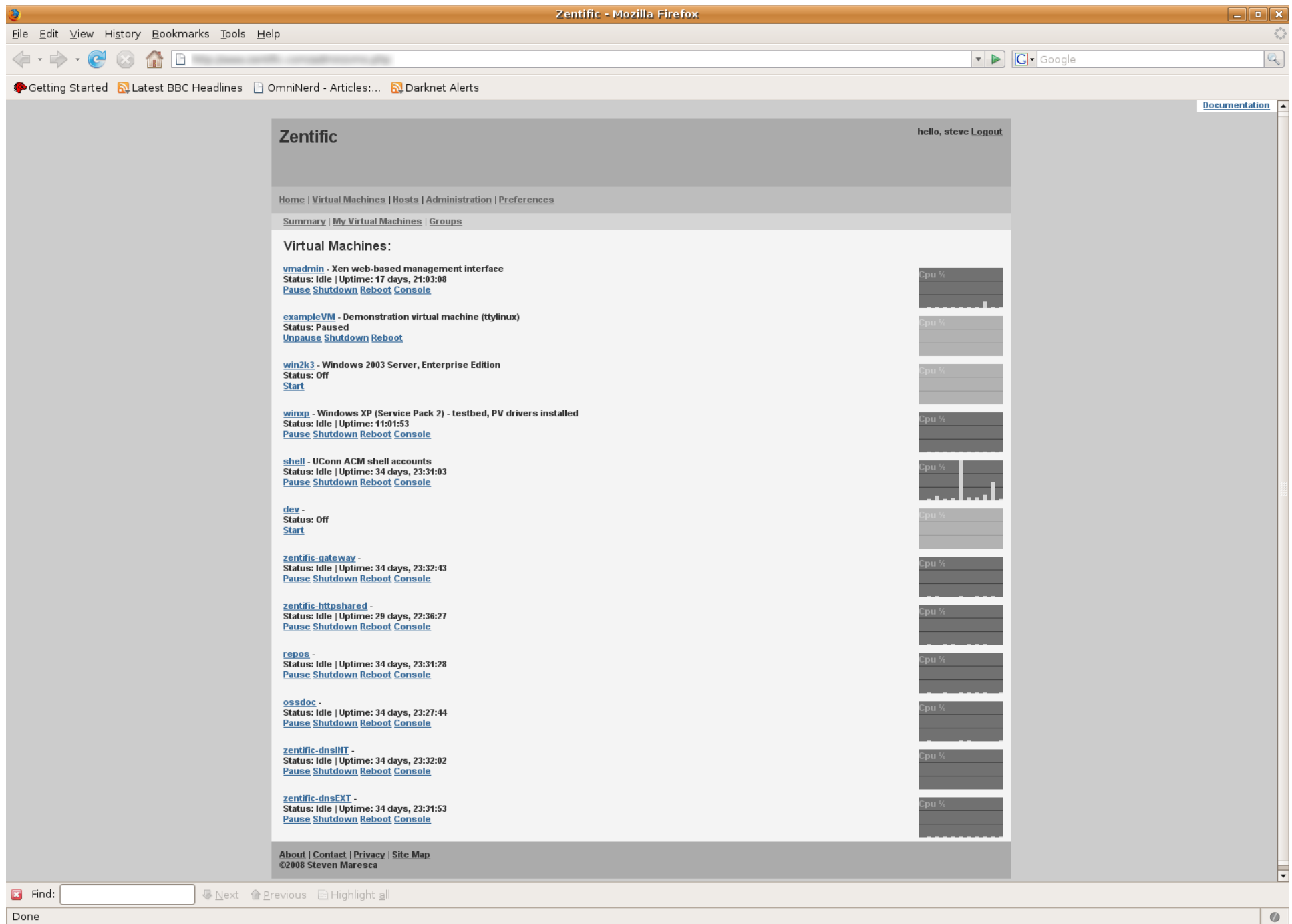


Manage VM configuration

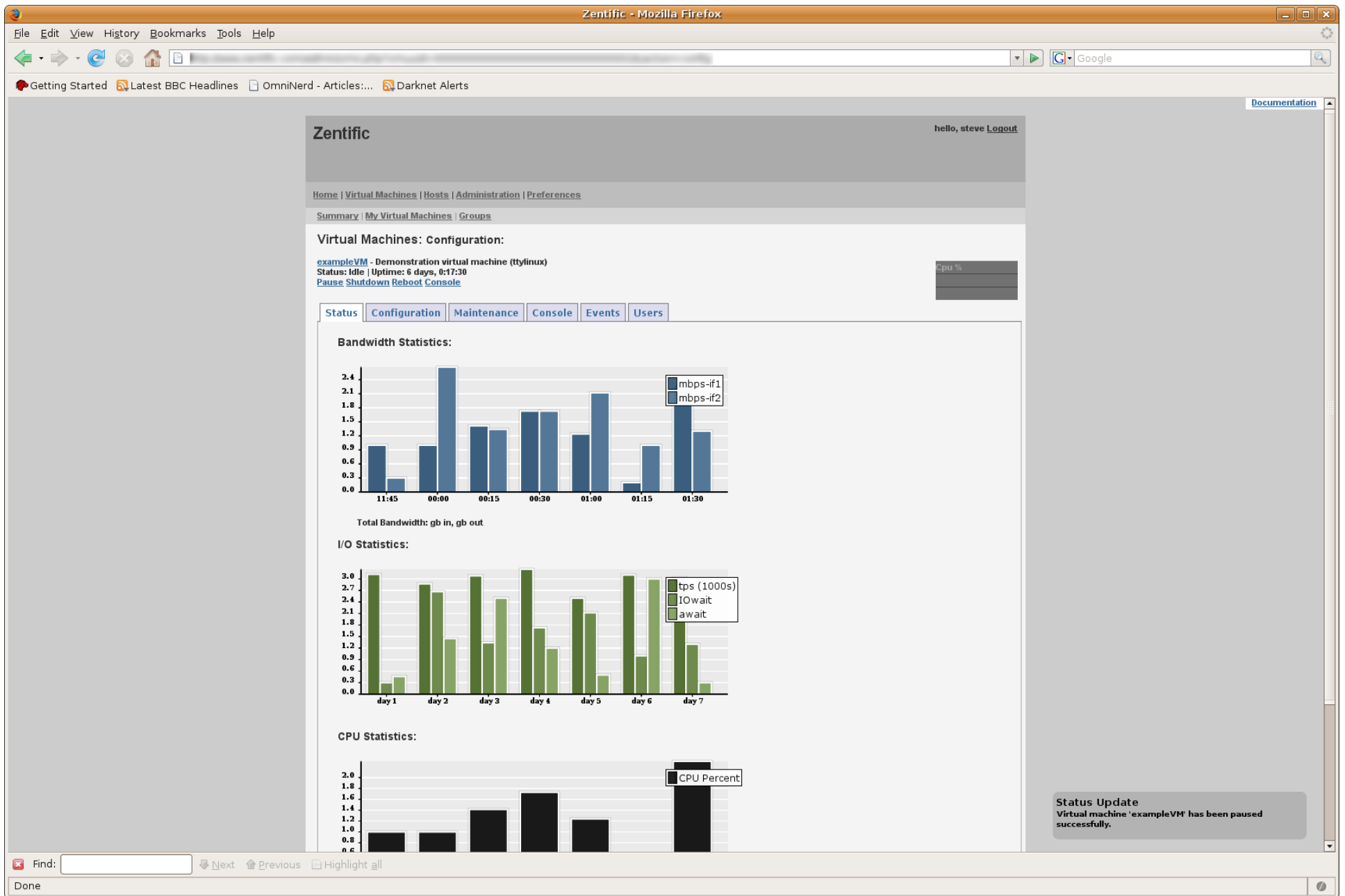
Zentific

Open source web-based management tool

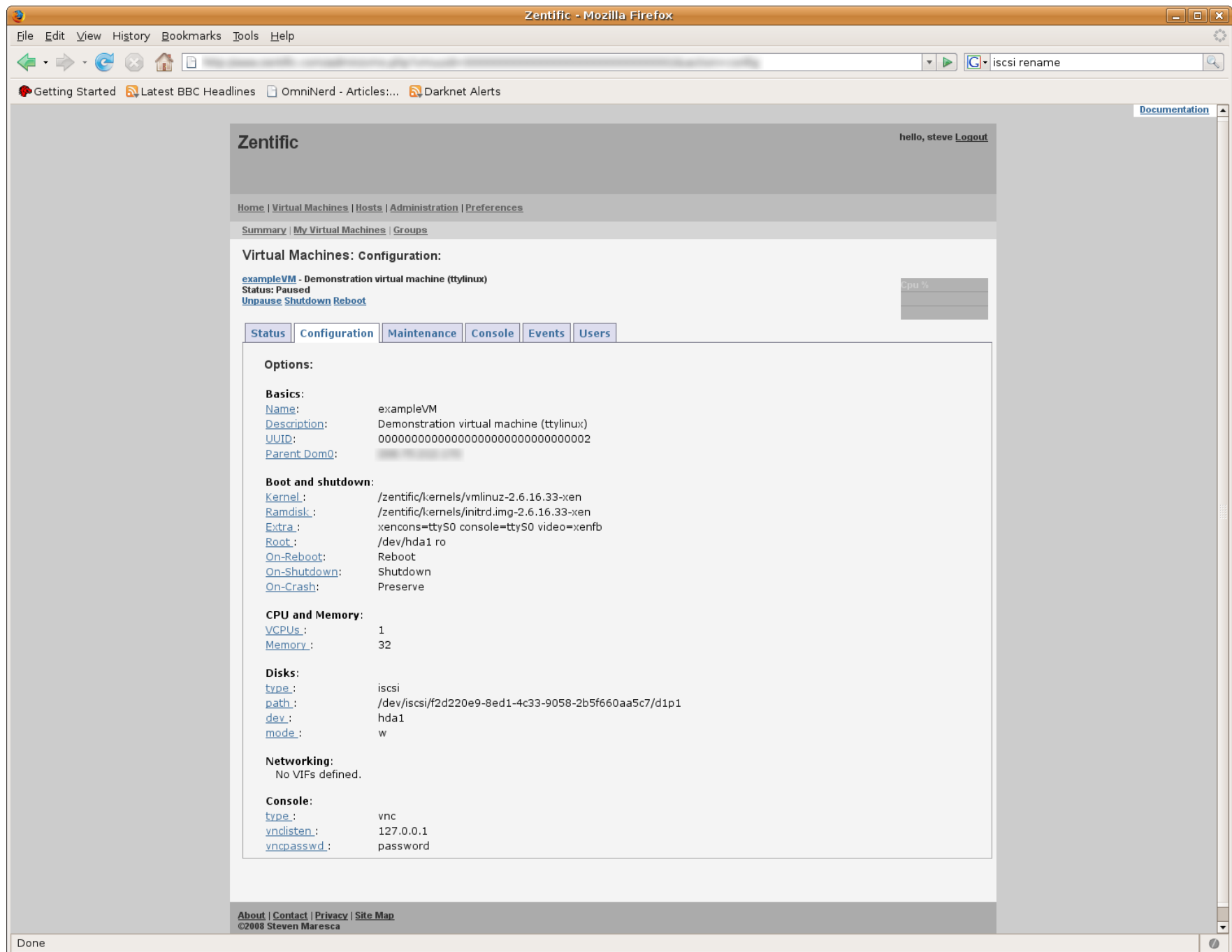
Allows for managing and provisioning Xen guests



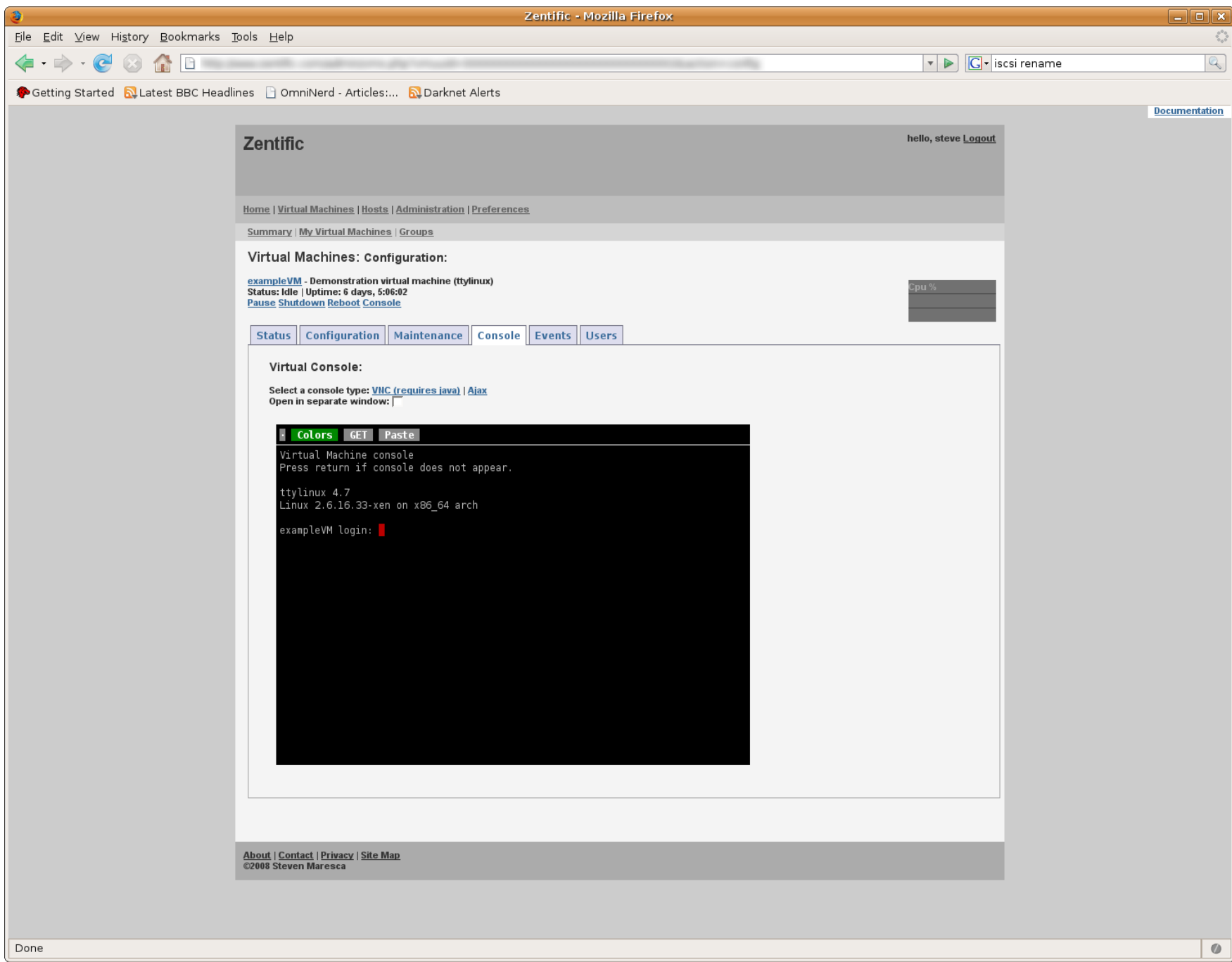
Main dashboard



VM Status Panel



VM Configuration



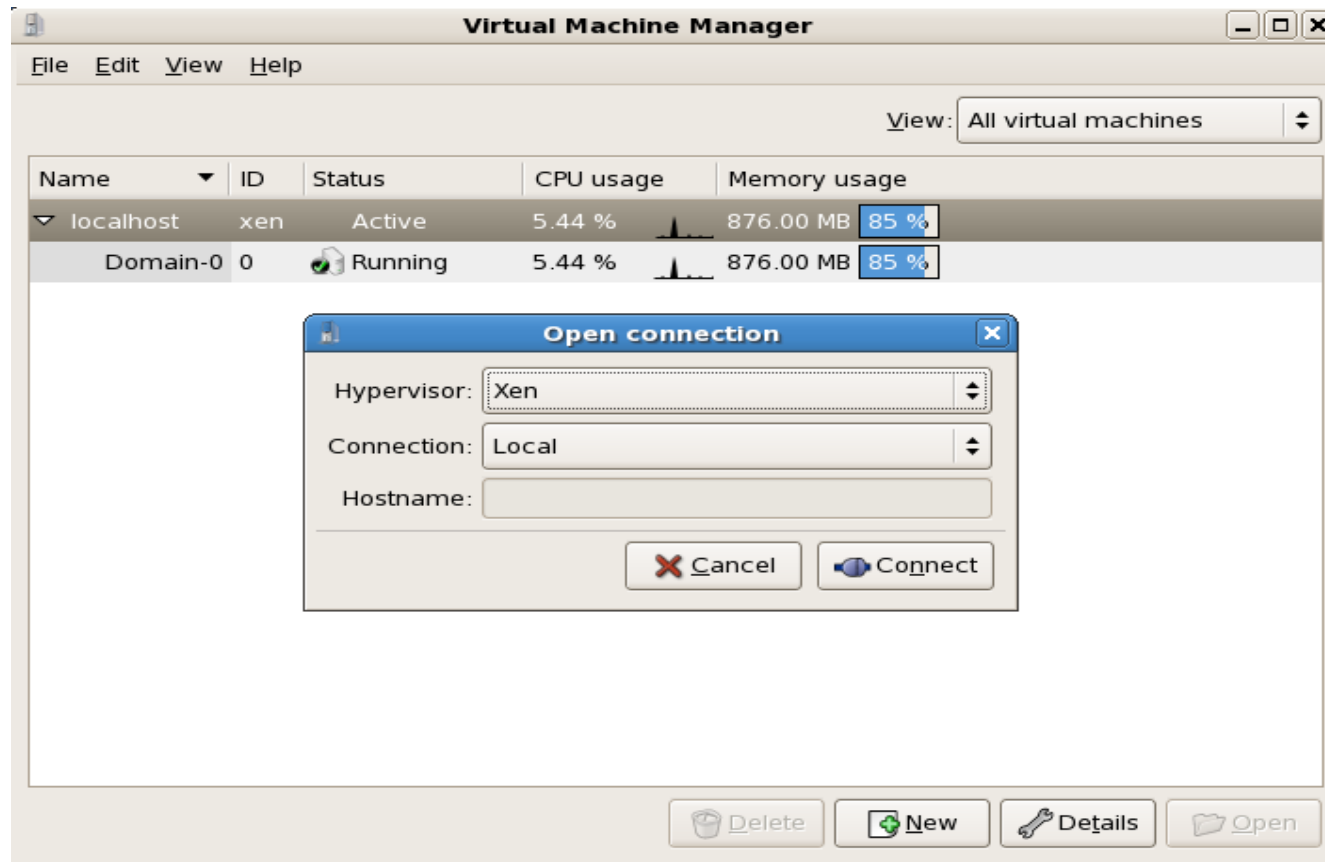
Web-based console

Virt-manager in CentOS

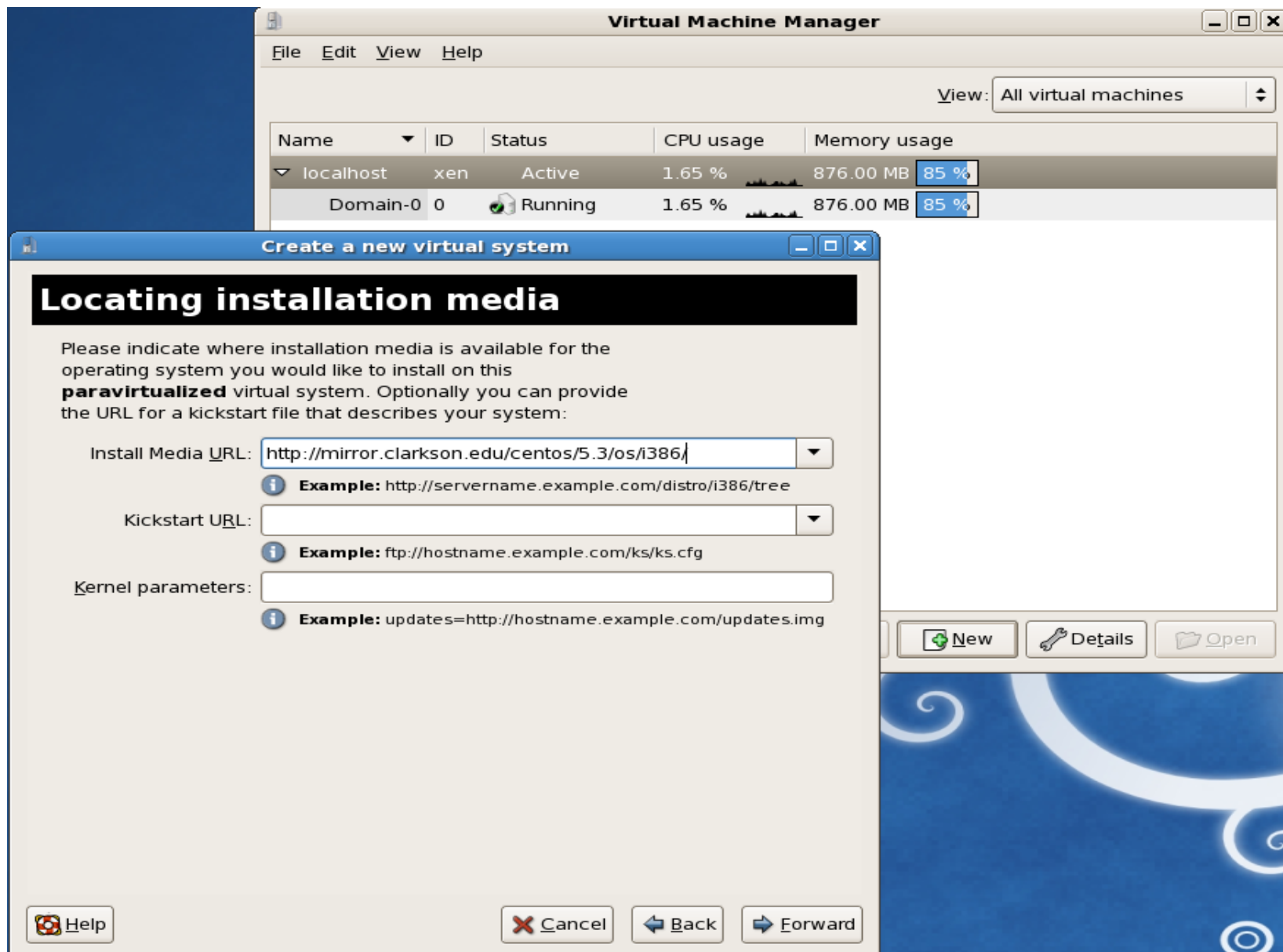
Desktop user interface for managing Virtual Machines

Allows for Xen guest performance monitoring, resource allocation, and domain creation.

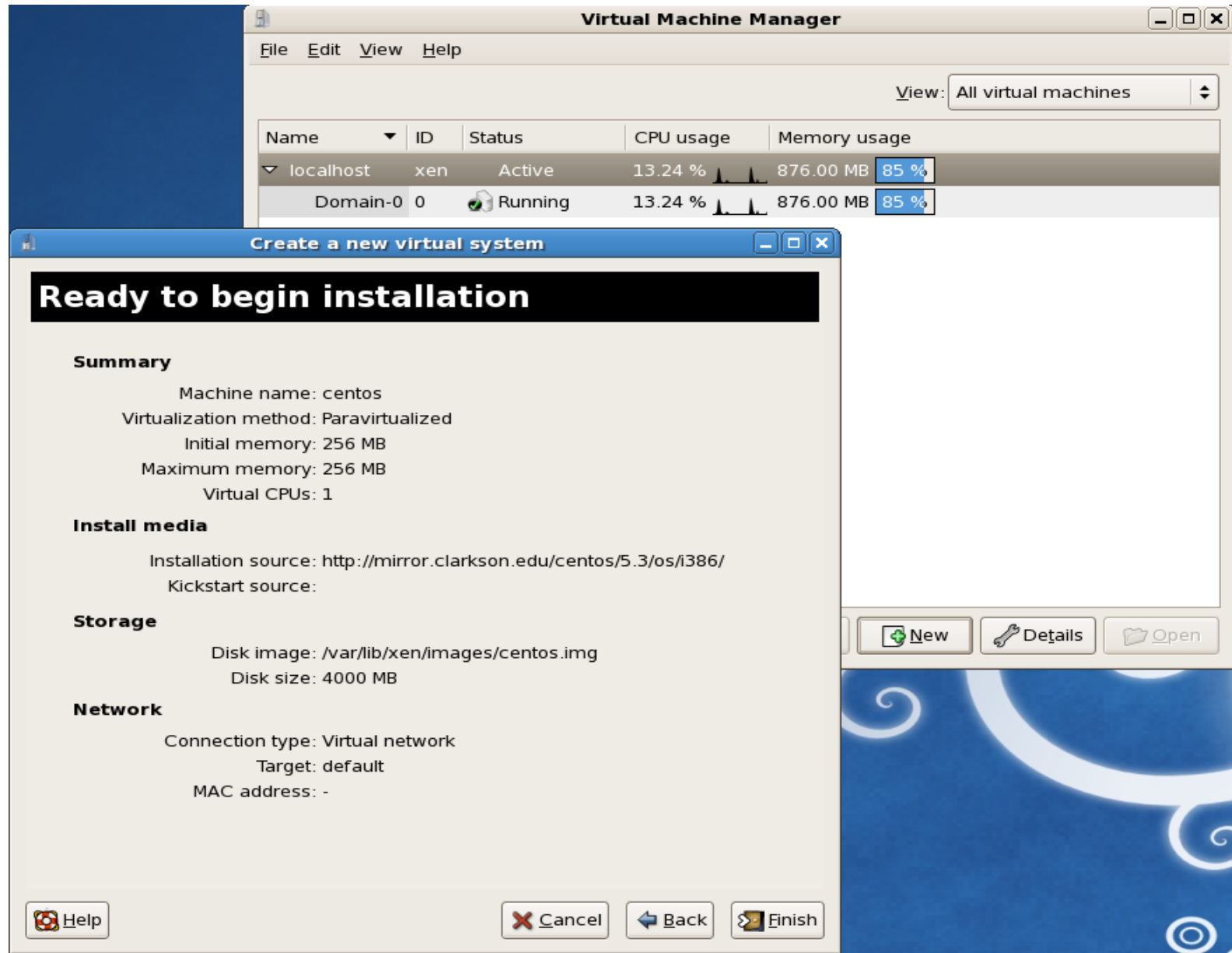
Open source with Red-Hat support



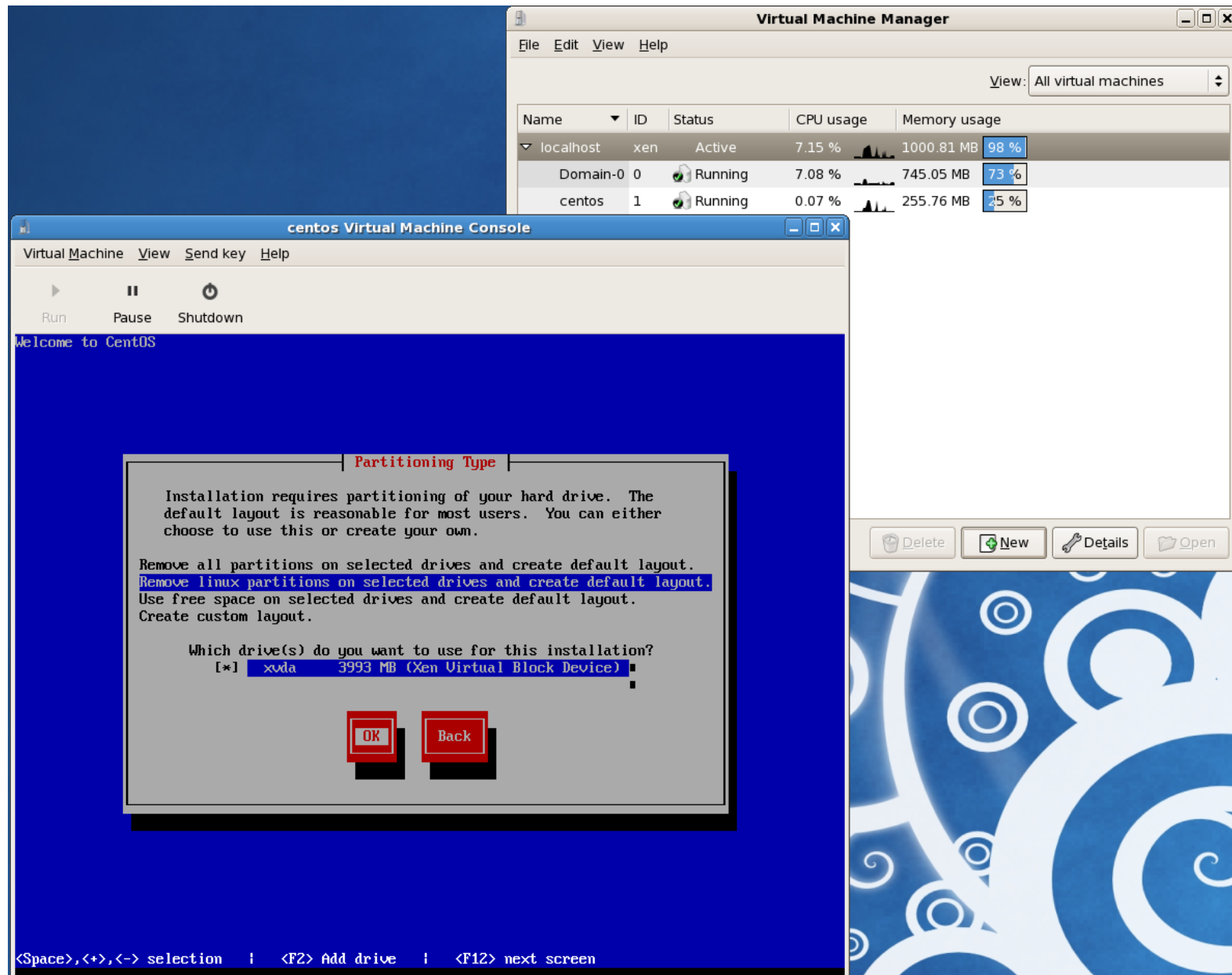
Virt-manager GUI Interface



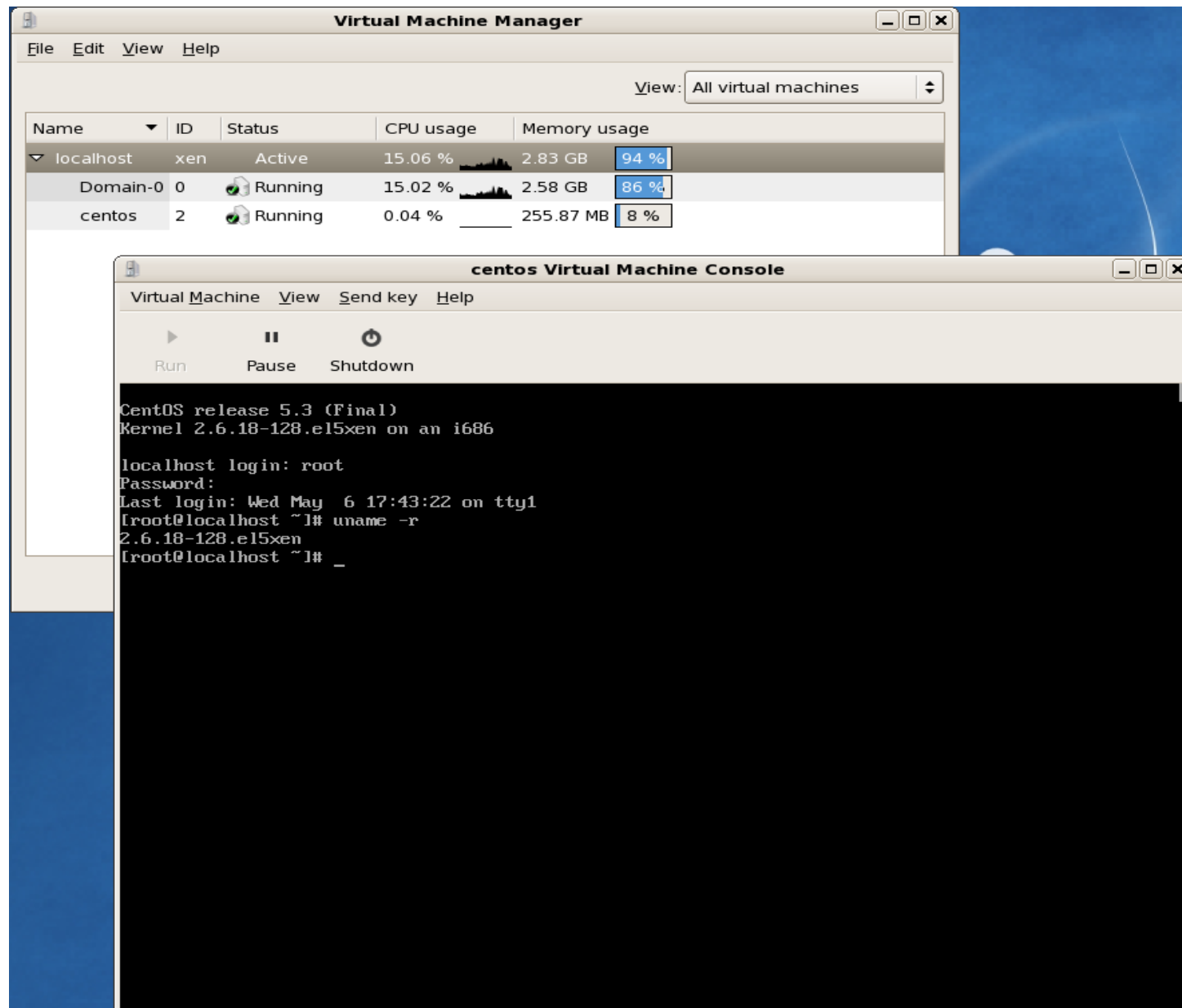
Creating a PV CentOS guest by URL



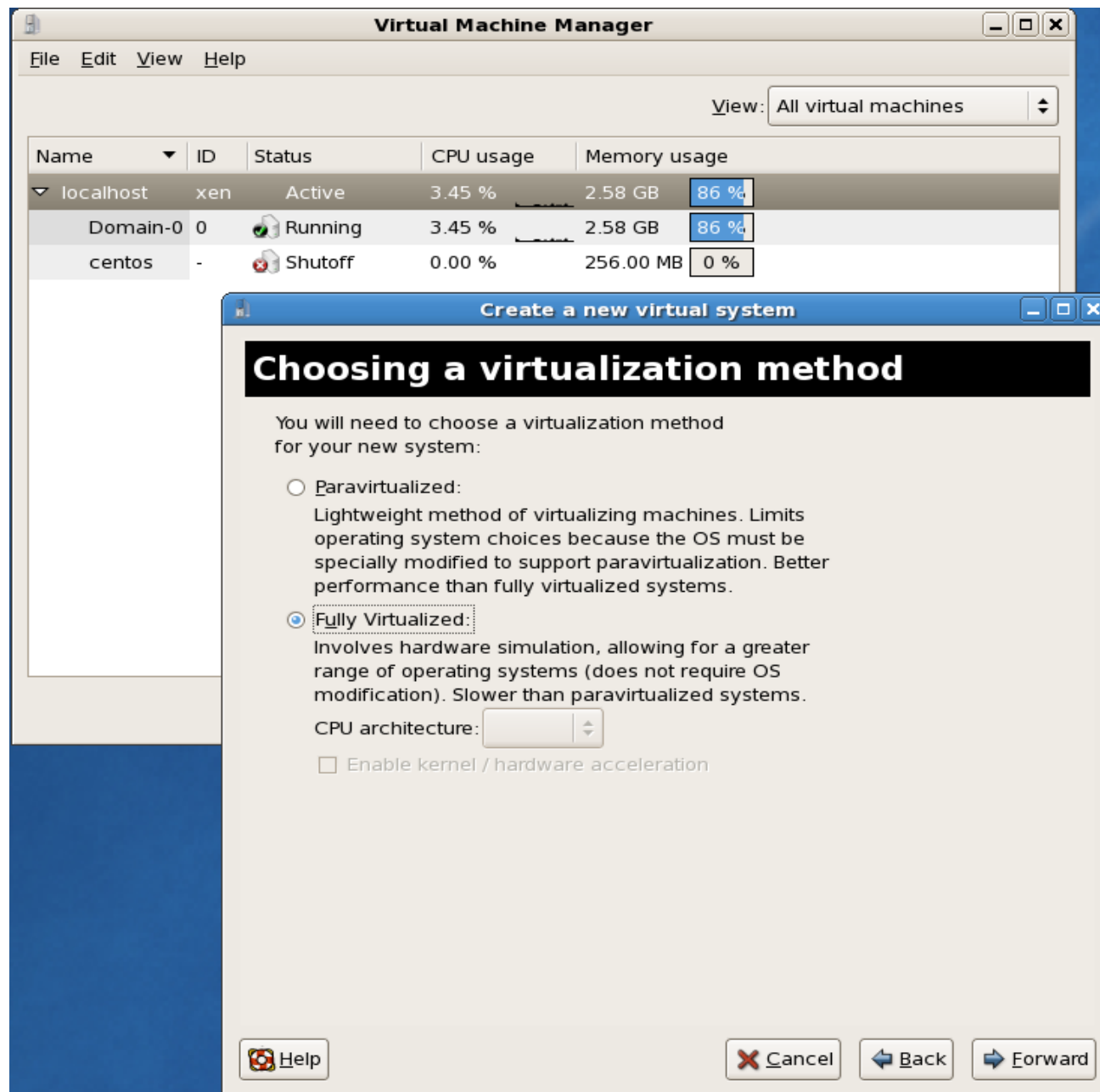
Finishing the configuration of a PV guest



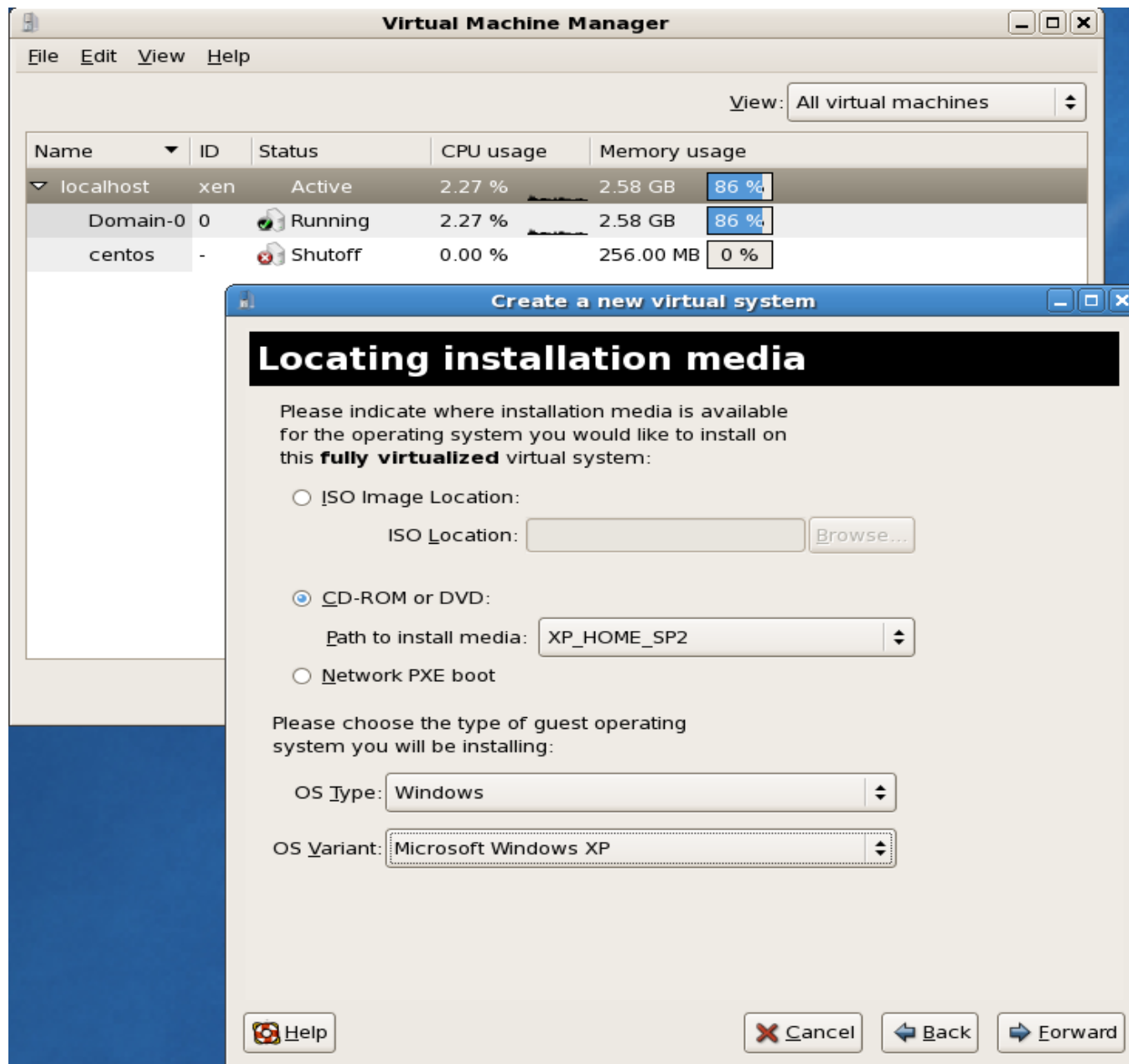
In the process of installing PV guest



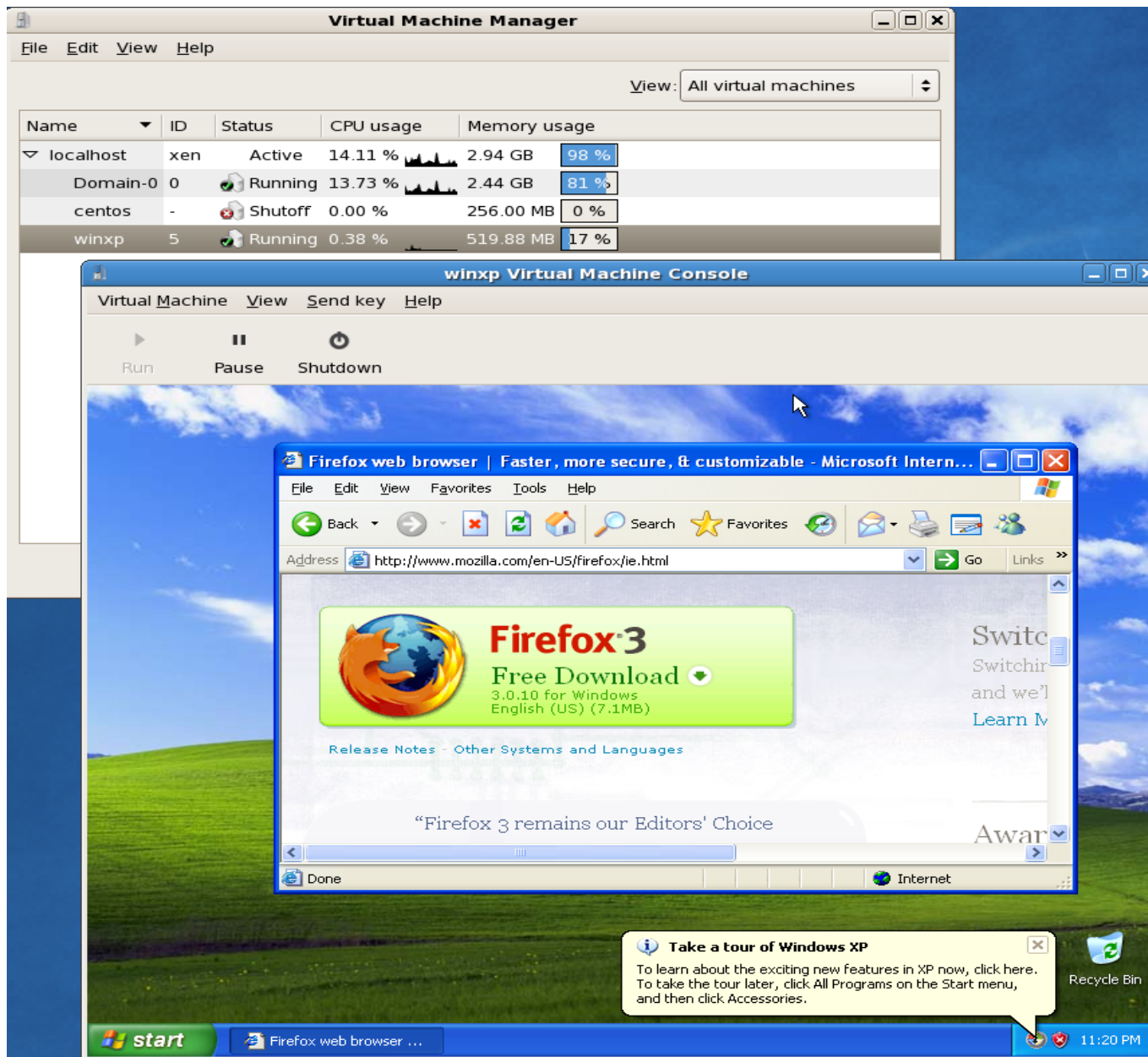
Accessing A PV CentOS guest



Configuring a HVM Guest



Installing a HVM WINXP guest from CD-ROM



A running WinXP HVM guest in Xen

Demo: Guest Image Creation

Summary: Image Creation

1. Distro-specific tools

virt-install

virt-manager

vmbuilder

debootstrap etc.

2. Prebuilt Images

e.g. stacklet.com

3. Copy another system's fs (may cause driver issues)

4. HVM images to run as PV guest

5. Install from ISO by hdc:cdrom

Summary: Guest Boot Options

1. Boot from dom0 kernel
2. Boot from domU kernel (put domU kernel into dom0)
3. Boot from dom0 pygrub (access domU kernel from dom0)

Session 3

Xen Troubleshooting and Advanced Topics

Session Outline

Troubleshooting

Advanced Network Configurations

Guest Relocation

Xen Device Models

Xen Stub Domains

Troubleshooting

Common problems

Domain0 will not boot into Xen kernel

- Make sure grub configuration is correct
- Make sure it is a Xen-modified kernel
- Make sure you have included all the correct modules in the ramdisk
- Check the boot messages to diagnose what stage is causing the error

Xend is not running

- Make sure you're in the Xen kernel on dom0 (`uname -r`)
- Try restarting it manually (make sure you have root access)
- Check your xend configuration file for typos
- Read carefully about the error information, maybe some scripts or xen tools are broken (RH libvirt conflicts)

Common problems (cont.)

DomU is hanging or frozen

- Many possible reasons
- A few things to check:
 1. Make sure the Xen pygrub/kernel path is correct
 2. Make sure ramdisk/initrd is present
 3. Make sure all required kernel modules are included
 4. Make sure the root device is specified correctly in guest config file (for paravirt) or grub config (for hardware assisted)
 5. Make sure that the console is specified properly in guest config file
(extra = 'xencons=tty')

Steps to solve other problems

Step 1: Double-check the Xen system and configurations

Step 2: Google for how-to or error Information

Step 3: Collect the system information

- Xen version, Linux distro, kernel version

- Xen dmesg

- Xen logs

- Xend configuration file

- Xen guest configuration file

Step 4: Look for help in Xen-user mailing list.

- Be polite

- Make sure to search first

Xen Mailing Lists

<http://lists.xensource.com>

Before Emailing:

Search the archives to see if someone already asked

<http://xen.markmail.org>

Things to include:

Figure out when, exactly, you experience the issue

Include the information you gathered (see step 3)

Explain what you are experiencing and what you expected

Notes:

It may take a few different people to figure out what's wrong

It may take time to hear back

Networking problems

DomU has no network:

- Think of Xen network just a Linux network except the network interface may be a little different. Nothing special!
 - Make sure dom0's network works properly
 - Make sure dom0's network interface is correct
 - Simplify domU's network to be one vif in bridging mode
 - Make sure domU's config file describes the same as in domU
 - Make sure domU's frontend and backend driver is not broken
- Like a Linux network, use Linux network tools to diagnose

Network Configuration Tools

ifconfig

- `ifconfig -a`
- `ifconfig eth0 up`
- `ifconfig eth0 xxx.xxx.xxx.xxx`

ethtool

- `ethtool eth0`
- `ethtool -K eth0 tso off`
- `ethtool -s eth0 speed 100 duplex full autoneg off`

brctl

- `brctl show`
- `brctl addif xenbr0 eth0`

route

- `route`
- `route del default gw xxx.xxx.xxx.1`

iptables

- `iptables -L`
- `iptables -A INPUT -p tcp -m state --state NEW --dport 21 -j DROP`

Network help from the mailling lists

Posting a network problem is a little different from the other Xen problems.

But still the same:

Search before asking

Be courteous

1. Describe your network design scheme (with a diagram if possible)
2. Describe xend network mode (Bridging/Routing/NAT?)
3. Post guest network configuration options
4. Use network tools to collect your network status
5. Describe your network symptoms

Advanced Network Configuration

Advanced Network Configurations

Multiple interfaces in dom0

Multiple interfaces in domU

Secure domUs

Limit domU bandwidth rate

Advanced Config. for Dom0 Network

Dom0 has multiple physical interfaces

Motivation:

Disaggregate domUs to different network segments

Procedure:

1. You can set up a virtual network for each network interface in dom0
2. Run the desired network script for each interface in dom0
(e.g. `/etc/xen/scripts/network-bridge start vifnum=0 netdev=eth1
bridge=xenbr1`; `/etc/xen/scripts/network-bridge start vifnum=0 netdev=eth2
bridge=xenbr2`)
3. Bind the domU's vif to the virtual network
(e.g. `vif = ['bridge=xenbr1',]`)

Advanced Config. for domU Interfaces

DomU has multiple interfaces

Motivation:

Allow a domU to interface with different network segments

Procedure:

Modify guest configuration file

E.g. Two different interfaces

```
vif = ['bridge=xenbr0', 'bridge=xenbr1']
```

Secure DomUs

Prevent domUs from accessing the outside or the dom0

Motivation:

Secure the domUs, but allow themselves to communicate

Procedure:

1. Create a dummy bridge in dom0 by *brctl*

```
brctl addbr dummybr0
```

2. Configure domUs to connect to that dummy bridge by:

```
vif = ['bridge = dummybr0']
```

Note: Don't attach the bridge to an actual physical interface

DomU Network Bandwidth Limits

Restrict domU's network bandwidth

Motivation:

Prevent domUs from abusing their network bandwidth and provide a better performance isolation

Procedure:

Configure domU's vif option with parameter *rate*

(e.g. `vif = ['...', rate=50Kb/s']`)

Note:

It is a new feature incorporated in Xen 3.3.1 or above.

Refer to `tools/python/xen/xm/create.py` :

`http://xenbits.xensource.com/xen-unstable.hg?
file/dadadf9ae7/tools/python/xen/xm/create.py`

Migration

Guest Relocation

Cold Relocation

Warm Migration

Live Migration

Cold Relocation

Scenarios:

- Moving guest between domain0s without shared storage or with different architectures or hypervisor versions

Command:

```
scp
```

Process:

- Shut down a guest
- Move the guest from one domain0 to another by manually copying the image and configuration files
- Start the guest on the new domain0

Cold Relocation (cont.)

Benefits:

- Hardware maintenance with less downtime
- Shared storage not required
- Domain0s can be different
- Multiple copies and duplications

Limitation:

- More manual process
- Service should be down during copy

Warm Migration

Scenarios:

- Movement of guests between dom0s when uptime is not critical

Command:

```
xm migrate
```

Process:

- Pauses a guest
- Transfers guest state across network to a new Domain0
- Resumes guest on destination host

Warm Migration (cont.)

Benefits:

- Guest and processes remains running
- Less data transfer than live migration

Limitations:

- For a short time, the guest is not externally accessible
- Requires shared storage
- Network connections to and from guest are interrupted and will probably timeout

Live Migration

Scenarios:

- Load balancing
- Hardware maintenance
- Power management

Command:

```
xm migrate --live
```

Process:

- Copies a guest's state to a new domain0
- Repeatedly copies dirtied memory until transfer is complete
- Re-routes network connections

Live Migration (cont.)

Benefits:

- No down time
- Network connections to and from guest often remain active and uninterrupted
- Server is still online

Limitations:

- Requires shared storage
- Guests on same layer 2 network
- Sufficient resources needed on target machine
- Domain0s must be similar

Guest Relocation (Summary)

Cold Relocation

- Completely manual
- No requirements

Warm Migration

- Automated
- Some requirements
 - Shared storage must be used
 - Dom0s must be highly similar

Live Migration

- Least downtime
- Most requirements
 - Shared storage must be used
 - Dom0s must be highly similar
 - Dom0s must be on the same subnet

Device Models

Xen Device Models

PV Split Driver Model

QEMU Device Model

Device Passthrough

PV Drivers in HVM

PV Split Driver Model

Generic backends

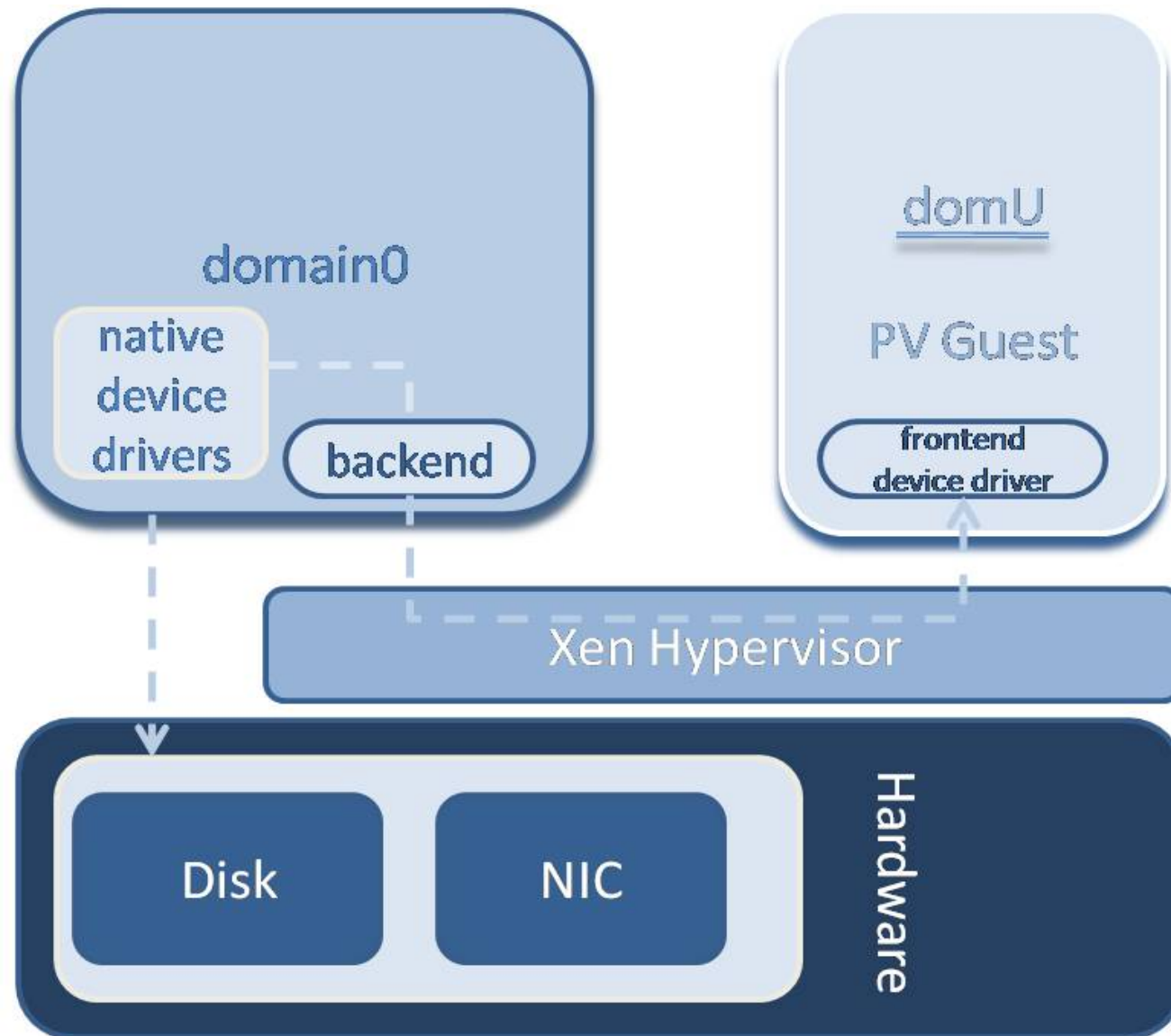
- Loaded in dom0
- Multiplexed to the native device driver

Generic frontends

- Loaded in guest domain
- Connects to the corresponding backend driver
- Guests use standard Xen virtual device drivers

The native device-specific drivers are in domain0

PV Device Model



PV Device Model: Network

Goal: virtualize different topologies (OSI 5 layers)

Bridging mode -> Data layer

Routing mode -> IP layer

Frontend-backend device model

Backend - sits in domain0

Virtual Interface Device (vif)

Frontend - resides in guest

Virtual Ethernet Device (veth)

PV Device Model: Storage

- Goal: virtualize different block devices
- Frontend-backend device model
 - Backend - sits in domain0
 - Block Backend (blkback)
 - Frontend - resides in guest
 - Block Frontend (blkfront)

HVM QEMU Device Model

- Provides emulation of devices
- Provides illusion of exclusive access to each guest
- Used primarily for HVM guests
- In guest config `device_model` set to `qemu_dm` binary

Device Passthrough

- Guests are granted full access to specific PCI devices
- The actual device driver runs in the guest

Benefits:

- Highest performance for a device
- Useful when virtualization does not support a device
- Moving a buggy driver from the domain0 to a guest

PV Drivers in HVM

Hybrid approach: mixture of both advantages

- HVM: Avoidance of modified base kernel
- PV: Device driver performance close to native

An option for closed source operating system to obtain the performance of PV guests

Xen-aware HVM guest

- Use Xen drivers in unmodified guest

Windows GPL PV drivers for network card

Stub Domains

Stub Domains

Roles of Domain0:

- Domain manager
- Domain builder
- Device drivers

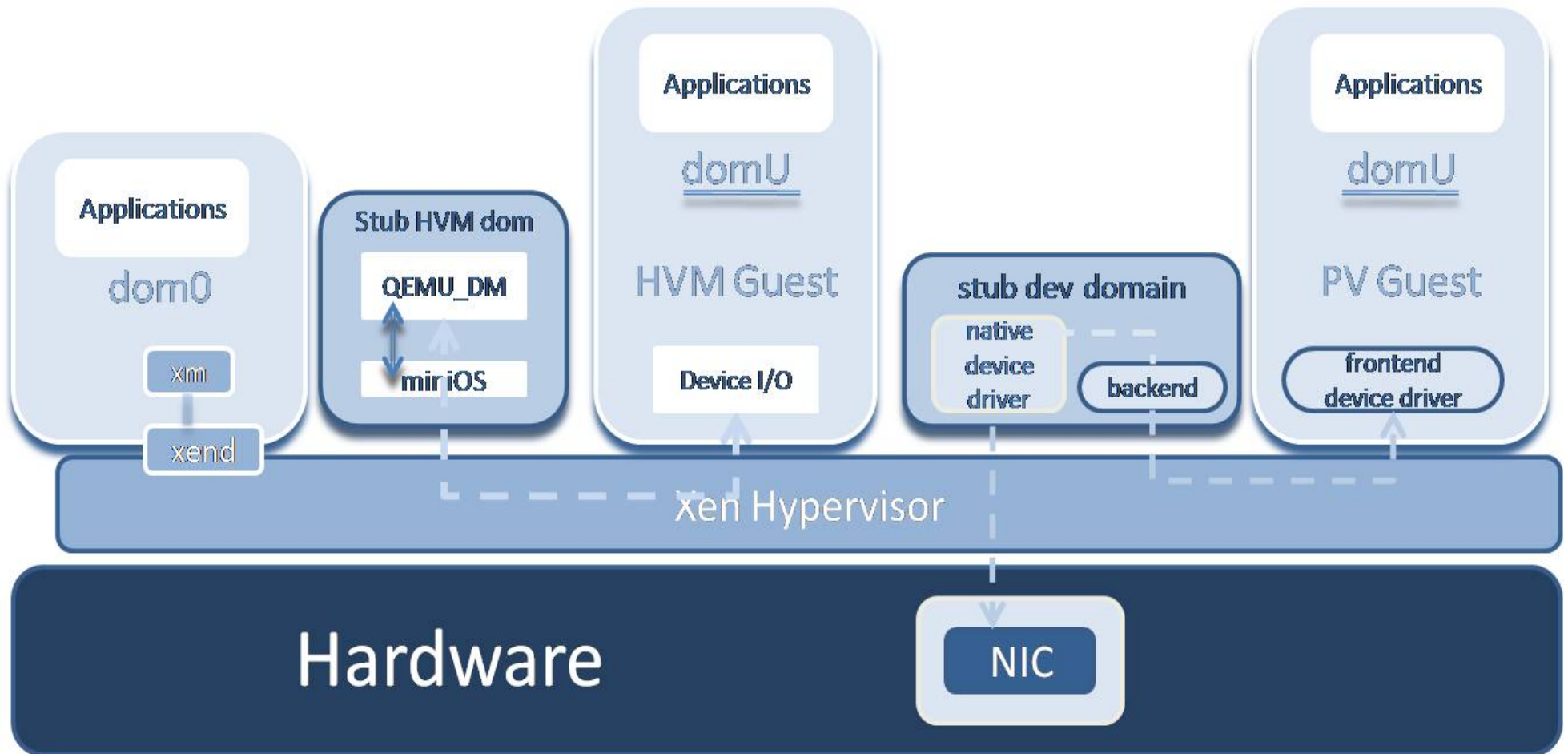
Problems:

- Waiting for domain0 to response
- Using domain0 resources (e.g. CPU time, memory)
- Scalability issues
- Security issues

Solution:

Stub domains to run Xen components

Stub Domain Model



Stub Domains

Benefits:

- Relieve domain0
- Have the same resource access as PV guests
- Hypervisor schedules them directly
- Improves accounting
- Improves performance
- More secure

Session 4

Xen Security and Examples

Outline of Xen Security and Examples

Secure Xen

Xen Use cases

Other Xen Topics

Secure Xen

General Notes

Hypervisor

Domain0

Guests

Other tricks

Insecure Virtualization

Virtualization systems take the place of a typical operating system on bare hardware

Virtualization systems can and do have vulnerabilities just like operating systems but:

- Hypervisors are typically thinner than general purpose operating systems; less code to secure?
- Placing applications and device drivers in other domains reduces the attack surface
- Open source hypervisors, like Xen, can have more eyes looking for vulnerabilities

Secure the components

Apply the standard practices of system security

- Secure hypervisor - keep patching your hypervisor
- Secure dom0
- Secure domUs

Adding Introspection

The in-progress project: Xenaccess

A full featured introspection for running domains
securing VMs' memory, CPU states, network, etc.

Refer to <http://www.bryanpayne.org/research/papers/acsac07.pdf>

Secure the Domain0 & Guests

- Follow standard system security practices
- Minimize number of software packages
- Minimize number of running services
- Minimize number of open network ports
- Deploy firewall and intrusion detection systems

Advanced Options for Securing the Hypervisor

Secure Hypervisor (IBM's SHype)

- Mandatory Access Control
- Don't allow co-location of VMs belonging to competing organizations
- xm label and policy specified in policy XML file
- Virtual Trusted Platform Module (vTPM)

Xen Security Modules (NSA's XSM)

- Emphasis on decomposing Domain0
- Identify all privileges that Domain0 usually has and allow those to be granted individually to dom0 or domU
- Based on SELinux

More advanced Securing Tips (cont.)

Dom0: Move services to stub domains

Guest: pvgrub instead of pygrub

- Secures guest domain startup

VM-aware Hardware

e.g. Intel VT-d chipset contains IOMMU unit

- Interrupt remapping based on VCPU rather than physical CPU
- Prevents insecure memory access through driver DMA

Xen Use Cases

Use case 1: Efficient use of 1 server

Application:

Consolidation of several lightly used services onto one server

Motivation:

Better resource utilization

Benefits:

Easy backup and secure

Reduced hardware costs

Keep services isolated

Use Case 2: Dynamic load balancing

Application:

Balance workload dynamically across many servers by relocating guests or scaling CPU Freq when detecting idle VCPUs

Motivation:

- Minimize the number of running physical machines
- Maximize the utilization of each machine

Benefits:

- Save power
- Better resource utilization
- Stable performance
 - A spike in one machine's usage won't affect others

Use case 3: Parallel computing cluster

Application:

Xen Cluster for parallel computing

Components:

- MPICH (parallel computing package)
- Xen (Virtual master and slave nodes)

Motivation:

- Virtual master runs a NFS server to manage slave nodes.

Benefits:

- Flexible relocation
- Balanced workload
- Homogeneous computing machines

Use case 4: High Availability

Application:

High Availability using a replicated filesystem and heartbeat

Motivation:

- Uninterrupted service

Benefits:

- Uninterrupted service
- Identical machines (even without identical hardware)

Use case 5: Cloud Computing

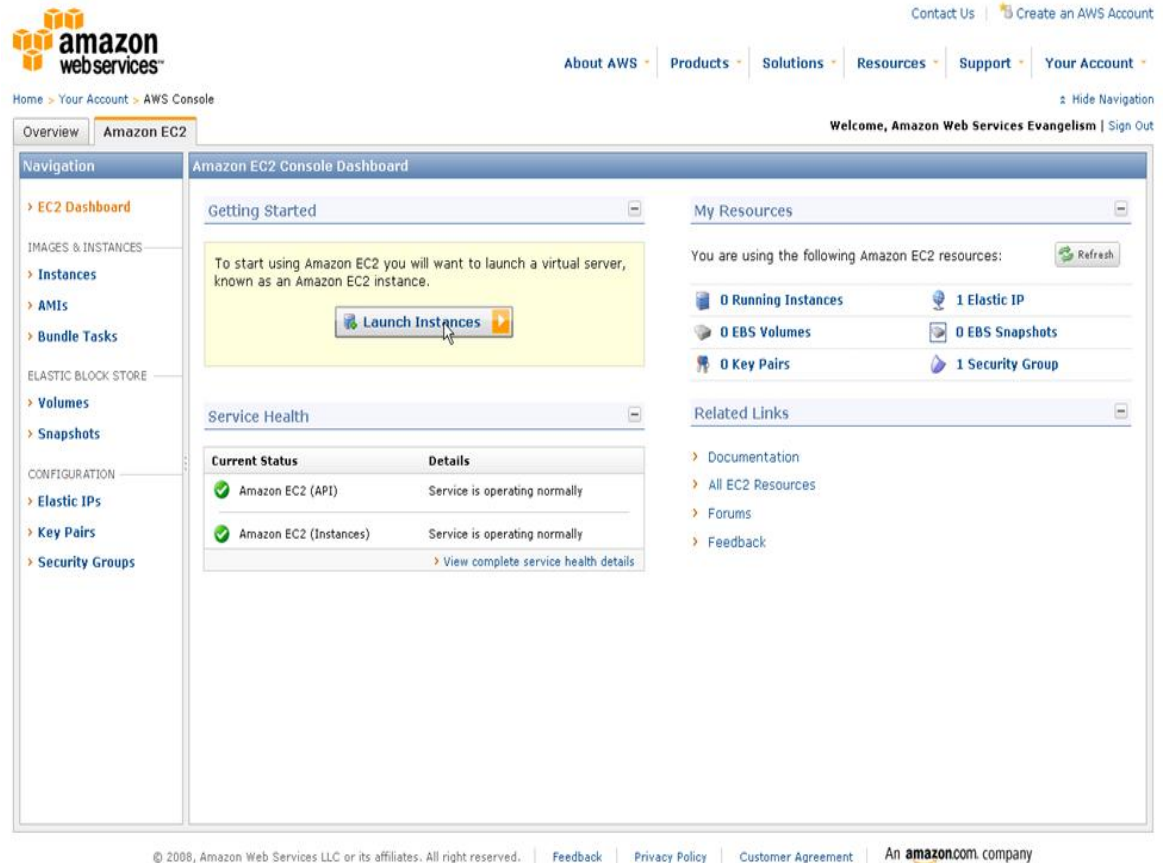
Application:
Private or Public Clouds

Components:

- Xen backend
- Web frontend interface

Examples:

- Amazon EC2
- Slicehost
- Eucalyptus, Nimbus, Ganeti
- Ubuntu Cloud

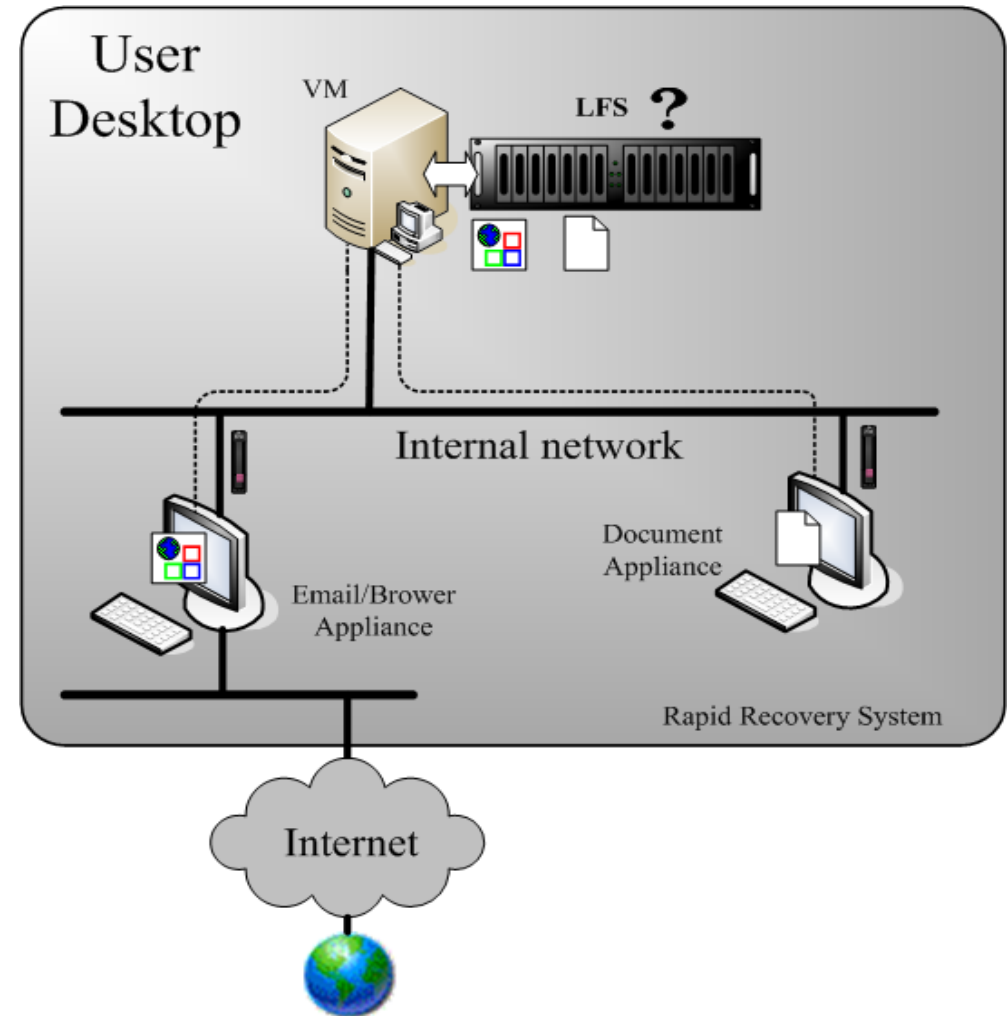


Use case 6: Not just for servers, desktops too!

Application:
Secure user Desktop

Motivation:
Isolate applications in VMs
Run different types of applications in different VMs that have different permissions

Benefits:
Quick recovery from malware
Revert to old copy of VM
Good isolation from other VMs
Prevent one program with a memory leak from affecting everything



Other Xen Topics

Nested VM

P2V

Time Synchronization

Xen Performance Tools

Virtualization APIs

Memory Management

Nested VMs

A physical machine can run Xen and HVM guests.

The HVM guests can run Xen and PV guests.

Uses:

- Test virtualization in a VM
- Teaching virtualization

P2V : Physical Machine to a VM

Conversion of a physical machine into a virtual machine

Scenarios:

- Virtualizing existing infrastructure
- Support legacy applications
- Backup

Available Tools:

- Use existing backup tools to create a file backup (e.g. clonezilla)
- P2V liveCD
- XenSever Convertor
- Xen express install CD
- Various third-party tools

Clock Drifting in DomU

Causes:

- Virtual CPU frequency is inaccurate
- Scheduler busy

Solutions:

- Sync with dom0 by setting `/proc/sys/xen/independent_wallclock = 0`
or
- Set up NTP client in DomU(`ntpdate`)

Note:

Can only be accurate to the ms level

Xen APIs

libvirt

- Provides a uniform interface with different virtualization technologies

Mainline Virtualization API (pv_ops)

- Provides a uniform paravirtualization interface in Linux kernel to avoid modifying Kernel (Linux2.6.30)

Xen Application Programming Interface

- Defines a stable XML-PRC API for controlling and managing Xen

Open Virtual machine Format (OVF)

- Define a set of metadata tags that can be used to deploy virtual environment across multiple virtualization platforms

Performance Measurement

CPU

`xm top / xentop`

Buffer

`xentrace`

Disk I/O

`xenmon`

Hardware Events

`xenoprof`

Memory Ballooning and Overcommit

Traditional allocation of memory to each image

- 4GB with 1GB per guest results in a max. of 3 VMs

Using memory overcommitment, more memory can be allocated than is on the system

- The same scenario with overcommitment would allow for 4 or more VMs
- Memory allocated to, but unused by, a VM is available for use by other VMs
- Reduces wasted resources

[http://www.xen.](http://www.xen.org/files/xensummitboston08/MemoryOvercommitXenSummit2008.pdf)

[org/files/xensummitboston08/MemoryOvercommitXenSummit2008.pdf](http://www.xen.org/files/xensummitboston08/MemoryOvercommitXenSummit2008.pdf)

Useful Resources & Acknowledgments

Papers:

- Original Xen paper (SOSP03): "Xen and the Art of Virtualization"
- Other research papers including materials from Xen Summits

Links:

- Xen Mailing list - <http://www.xen.org/community/>
- Xen Wiki - <http://wiki.xensource.com/xenwiki/>
- Xen Blog - <http://blog.xen.org>

Books:

- *The Definitive Guide to the Xen Hypervisor*
- *Running Xen: A Hands-On Guide to the Art of Virtualization*

Document Information

Special thanks go out to the **Clarkson Open Source Institute (COSI)** and the rest of the **Applied C.S. Laboratories** at Clarkson University.

<http://cosi.clarkson.edu>

<http://cslabs.clarkson.edu/>

License

This work is licensed under a Creative Commons Attribution-Share Alike 2.0 Generic License.

You may use, distribute, and reuse this document freely, as long as you give credit to its authors and share alike, under the full terms of this license.

The full text of this license is available online at:
<http://creativecommons.org/licenses/by-sa/2.0/>

