Kernel-mode Virtual Machine (KVM)

Tom Eastep
Linuxfest NW
April 26-27, 2008
Bellingham, Washington

Outline

1. Introduction to Virtualization Techniques
2. Pros and Cons
3. Where does KVM fit in
4. QEMU-kvm
5. Creating a virtual disk
6. Installing an OS on a virtual disk
7. Running the VM
8. Graphical Tools
9. Networking Options
10. Demo

Preliminaries – Disclaimers

- This presentation is not sponsored by Hewlett-Packard
  - The company is blameless for anything that I say ☺
- My goal is to give you some hints about what works for me

Terminology

- **Virtualization:** Running several OS environments on a single physical system
- **Host**: The system hosting one or more virtual machines
- **Host OS**: The operating system running on the Host
- **Guest**: A virtual machine environment
- **Guest OS**: The OS running on a particular guest
- **Hypervisor**: A virtual machine monitor that runs in a layer between the virtual machine(s) and the underlying hardware.
  - **See** http://en.wikipedia.org/wiki/Hypervisor

Virtualization Techniques

- Single OS Image – Virtuozo™, Vservers, OpenVZ, Zones.
  - “chroot on steroids”
  - Hard to establish protection zones
  - Networking sometimes confuses administrators
- Full Virtualization – Vmware™, VirtualPC™, Virtualbox™, QEMU
  - Run multiple unmodified guest OSes
  - Hard to Virtualize X86 efficiently
- Para-virtualization – Xen
  - Run multiple guest OSes ported to a special architecture (Xen/X86)
  - Full virtualization with AMD & Intel's *Pacifica and Vanderpool* extensions (think of them as ring “-1”).

Some Virtualization Products

Product	Requires special host kernel	Supports unmodified Guests	Pros	Cons
Single-OS Products	Yes	No	Minimal Cost per additional VM	Guests must run the same OS as the Host. Guest OS crash kills Host and all Guests.
VMware™ Server	No, but choices are limited	Yes	Slick tools for managing Guests	Restrictive License. Narrow host OS support
Xen	Yes	Yes (with Pacifica or Vanderpool) . No, otherwise.	Best Performance.	No accelerated graphics support for host
QEMU	No	Yes	Broad emulation	Poor Performance (even with kqemu)
KVM	No	Yes (requires Pacifica or Vanderpool)	Good Performance. Supported by your distribution	A little flakey yet. Requires more memory per VM than Xen.

What fits where?

Product	Best Fit
Single-OS Products	You want the minimum cost per VM
Xen	Performance , stability and software fault isolation are your most important needs.
VMware Server	You are willing to run an OS that is supported by VMware and/or you use VMware commercial products.
QEMU	You want flexibility in your host OS and host graphics hardware ; performance is not so important and/or your CPU doesn't have virtualization support (kqemu can help performance)
KVM	You want flexibility in your host OS and host graphics hardware ; you want good performance and your CPU has virtualization support

KVM

- Part of Linux Kernel since 2.6.20
- Requires CPUs that include virtualization support.
  - Must be enabled in your BIOS
  - **vmx** (Intel) or **svm** (AMD) in /proc/sys/cpuinfo 'flags'
- Kernel Modules
  - One generic (kvm)
  - One vendor-specific (kvm-intel and kvm-amd)
- Allows the Linux Kernel to act as a type 1 hypervisor for guests
  - See http://www.qumranet.com/art_images/files/8/KVM_Whitepaper.pdf
- Doesn't do emulation itself
  - Emulation provided by a modified version of QEMU

QEMU-kvm

- Modified version of QEMU
- Available with current Distributions
- Bitness of Guest need not be the same as that of the host
  - But see the chart at http://kvm.qumranet.com/kvmwiki/Guest_Support_Status for gotchas.
- Performance is good
- No restrictions on host hardware or OS (other than it must be 2.6.20 or later)

KVM

- Under OpenSuSE™ 10.3, this is not quite ready for prime time. I've experienced:
  - A spontaneous reboot of the host when starting a guest.
  - Guests starting in full-screen mode.
  - Startup of guest 'hangs'
  - Graphical management tool (virt-manager) doesn't work with QEMU (Xen only)
  - Guest attempt to re-boot causes QEMU-kvm to loop.
- In the next round of distributions, I think it will be a very attractive virtualization option
- The remainder of this presentation describes what you can do in the mean time

Creating a Virtual Disk

`qemu-img create -f qcow` ***file Size***

- **qcow** (copy-on-write) – The most flexible disk image format supported by QEMU
- ***file*** – Name of the disk image file
- ***size*** – Size of the disk

Example:

```
qemu-img create -f qcow kvm/Fedora.img 10G
```

Preparing System (as root)

- Install your distributions KVM package(s).
- Create a *kvm* group, if your distribution doesn't do that for you when you install the KVM package.
- Add yourself to the *kvm* group.
- In /etc/udev/rules.d, add an entry for KVM if it doesn't already exist:
  - `KERNEL=="kvm", MODE="0660", GROUP="kvm"`
- Load the KVM Modules
  - `modprobe kvm`
  - `modprobe kvm-intel` or `modprobe kvm-amd`

Installing an OS on a virtual disk

```
sudo qemu-kvm disk\
  -net nic,model=rtl8139\ #Emulated NIC
  -net user               #Use simple networking model
  -soundhw es1370\        #Emulated Sound Card
  -m memory \             #VM RAM
  -cdrom  install-image \ #Installation Image
  -no-reboot\             #Re-boot is broken on my system
  -boot d\                #boot from CDROM
  -daemonize              #fork to background
```

- ***disk*** – Image file created using qemu-img
- ***memory*** – Amount of ram in MB
- ***install-image*** – installation DVD/CD or .iso file

Running the VM

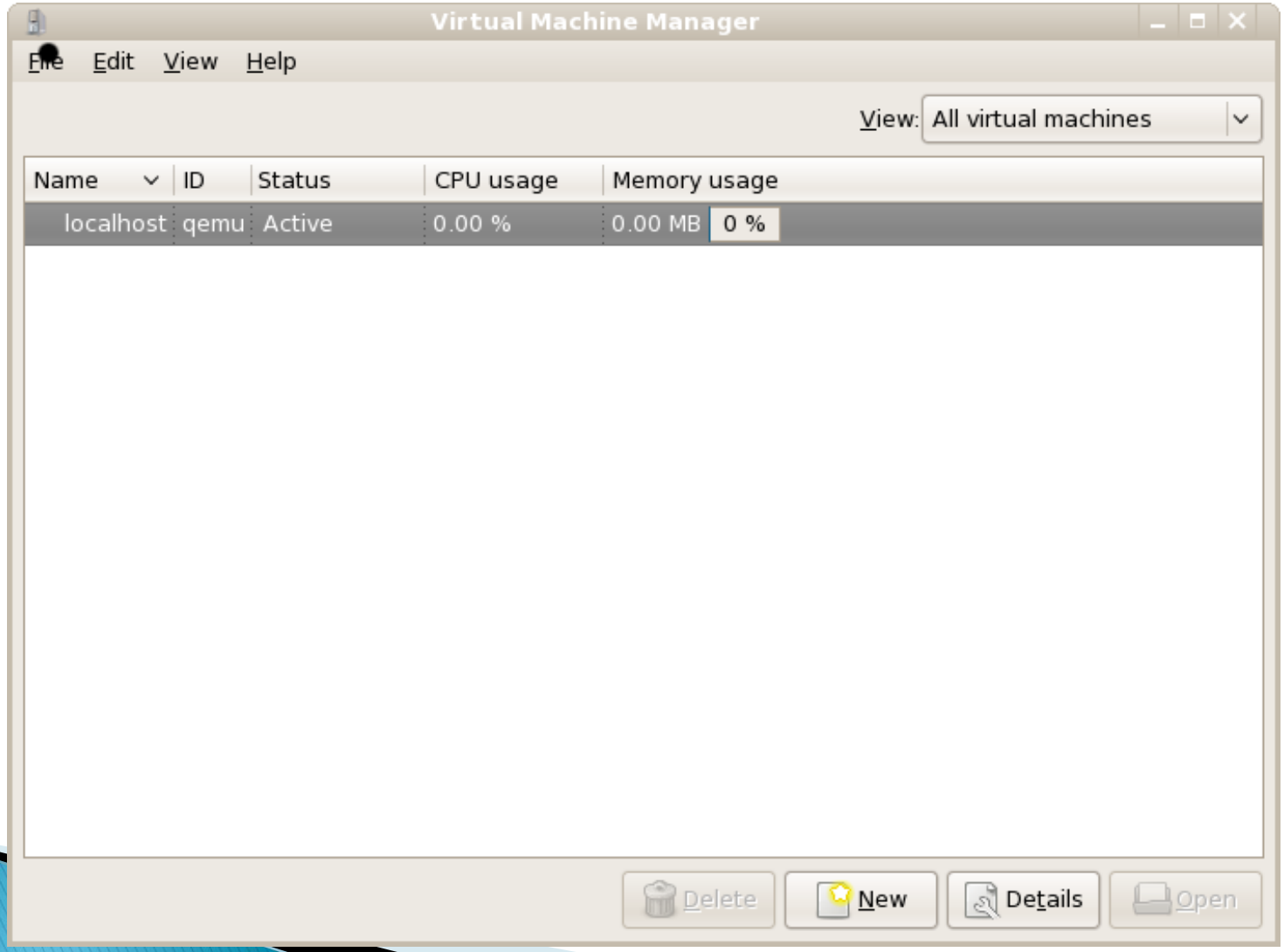
```
sudo qemu-kvm disk\
  -net nic,model=rtl8139\ #Emulated NIC
  -net user               #Use simple networking model
  -soundhw es1370\        #Emulated Sound Card
  -m memory \             #VM RAM
  -no-reboot\             #Re-boot is broken on my system
  -daemonize              #fork to background
```

- ***disk*** – Image file created using qemu-img
- ***memory*** – Amount of ram in MB

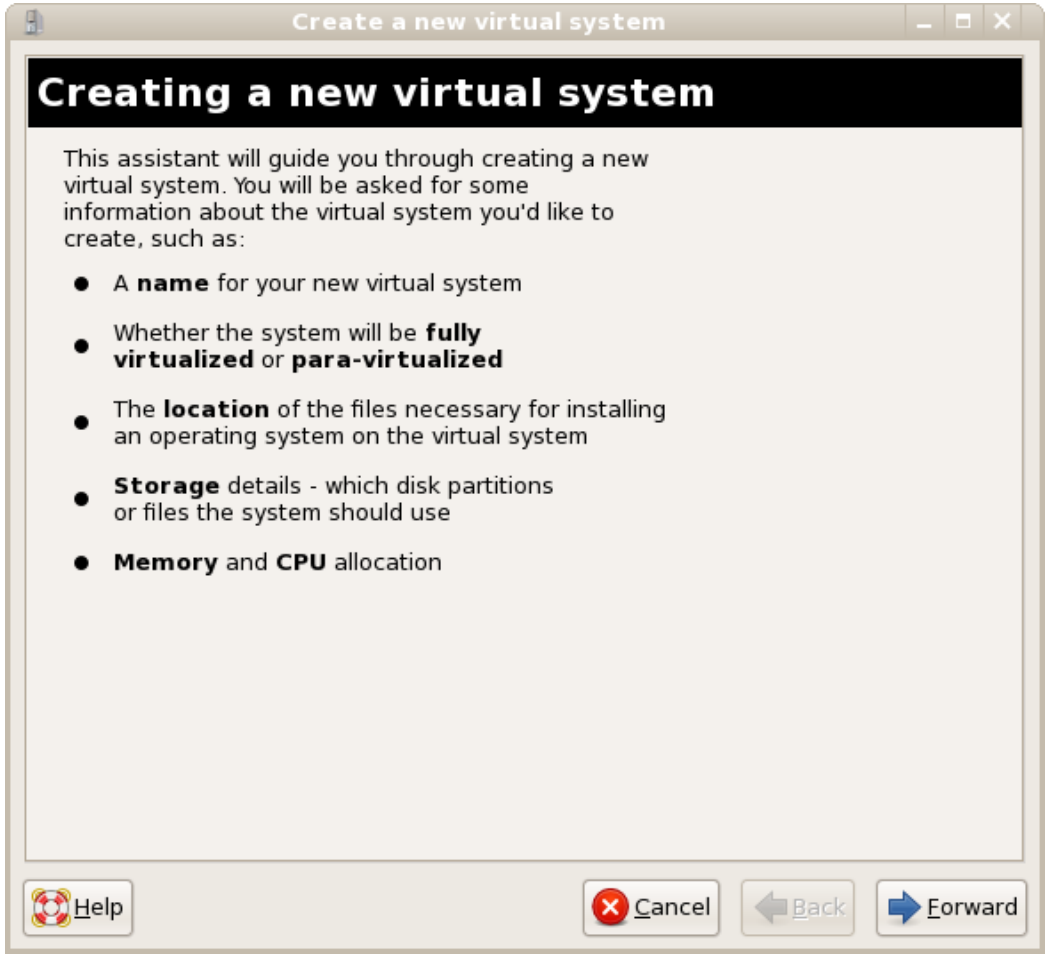
Doing it all the Easy Way

- If you run a Distribution like Fedora 8 that has a virt-manager that supports QEMU and QEMU-kvm, you can do all of this graphically

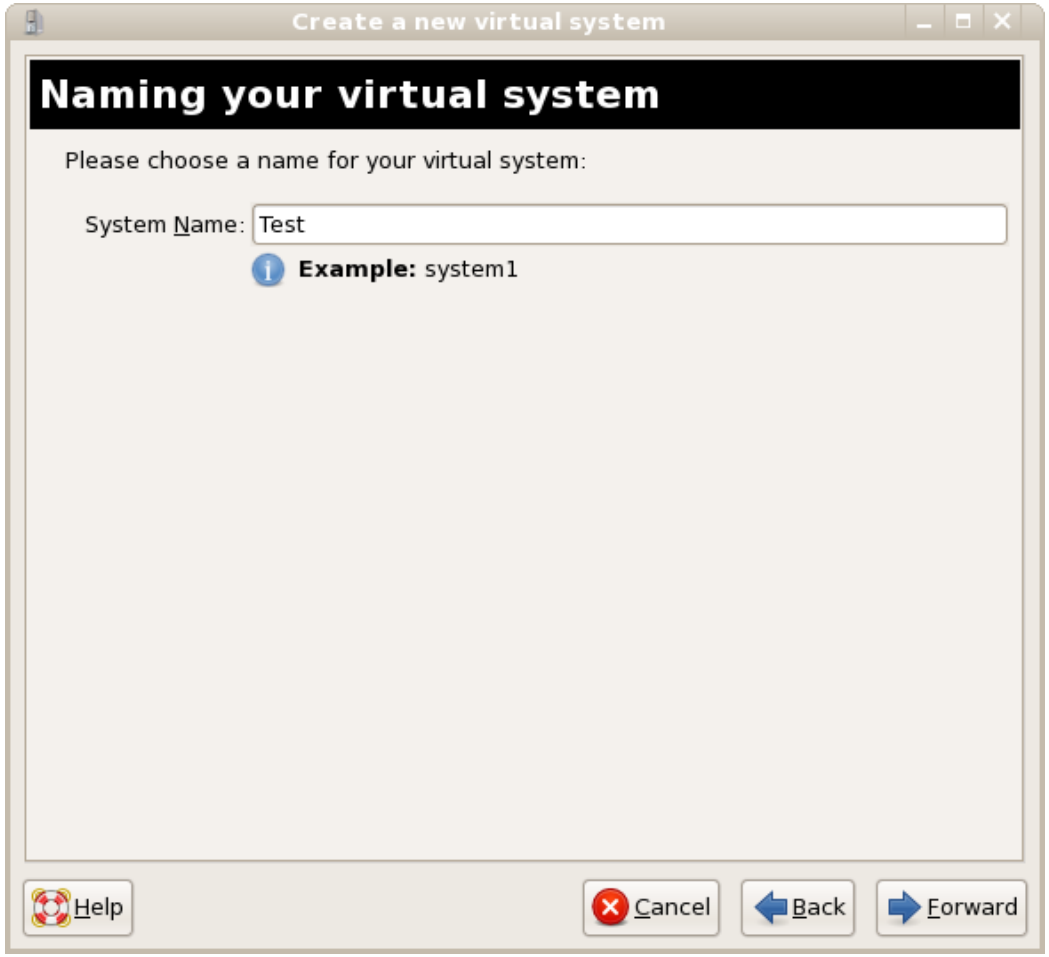
Doing it all the Easy Way (continued)



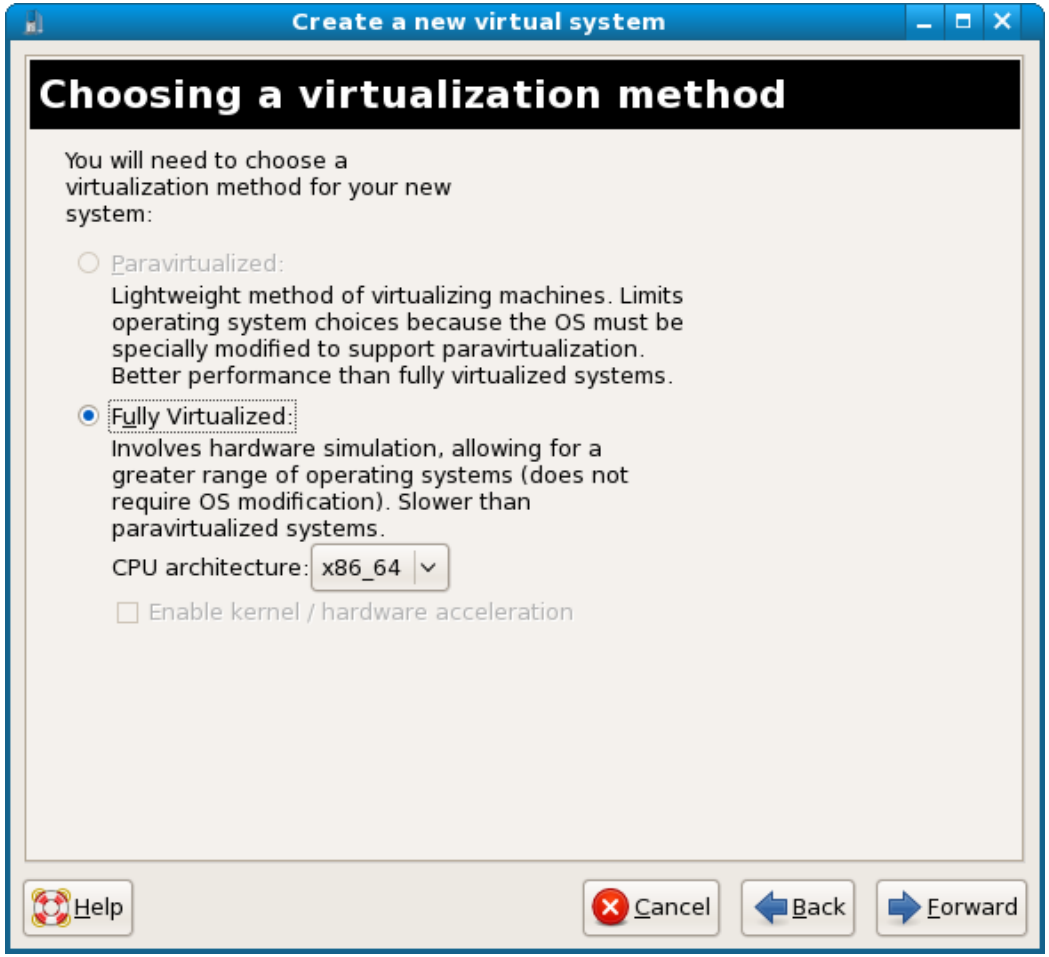
Doing it all the Easy Way (continued)



Doing it all the Easy Way (continued)



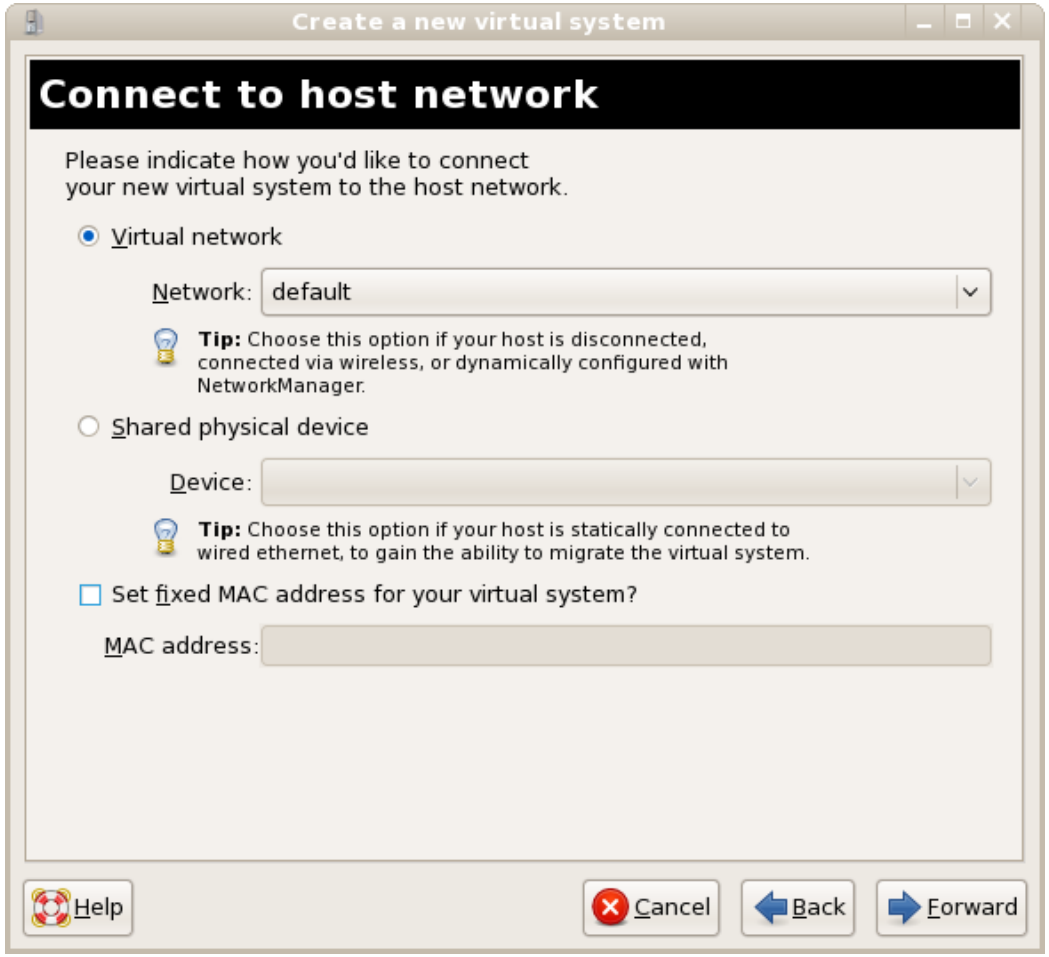
Doing it all the Easy Way (continued)



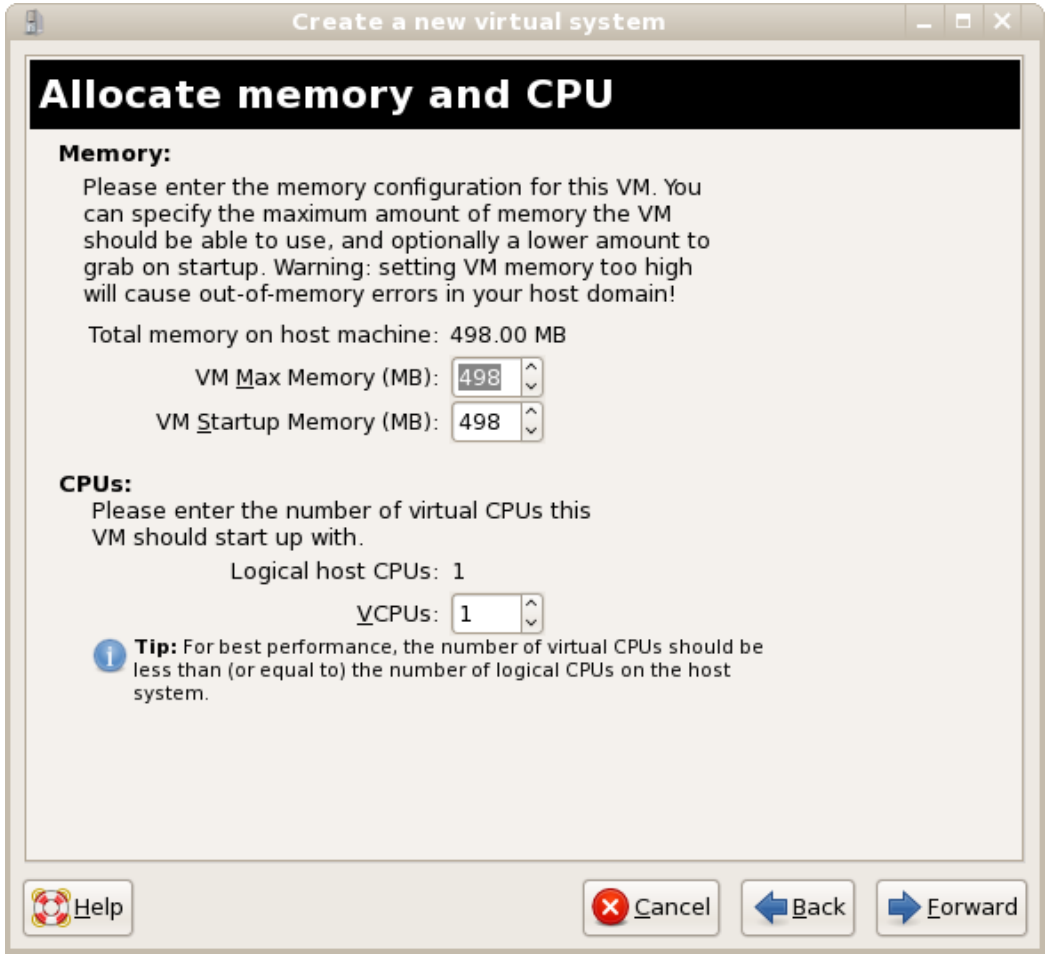
Doing it all the Easy Way (continued)

Doing it all the Easy Way (continued)

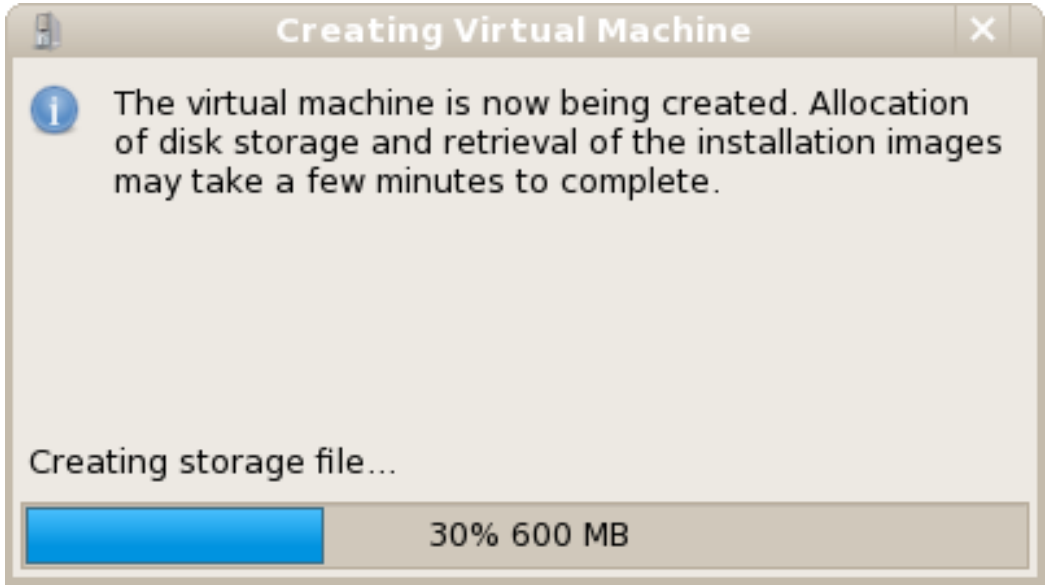
Doing it all the Easy Way (continued)



Doing it all the Easy Way (continued)



Doing it all the Easy Way (continued)

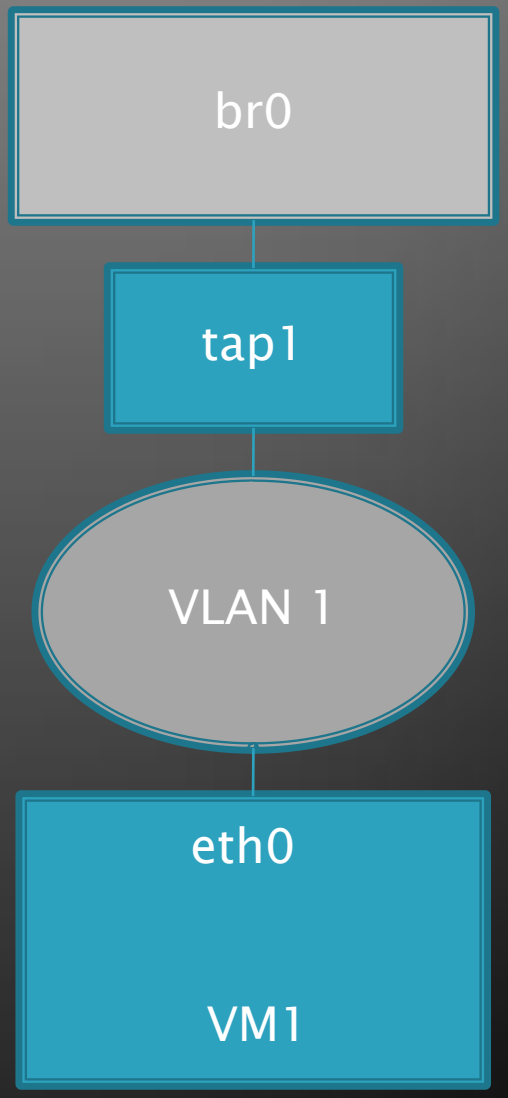


Networking Options – User

- Preceding slides used "**–net user**"
  - DHCP server build into QEMU supplies an IP address to the host
  - QEMU creates sockets to support connections from VM to the rest of the world
  - Performance poor
  - Good choice for initial guest OS installation

Networking Options – Bridging

- Each VM is associated with a VLAN
- Each VLAN is associate with a TAP device in the host
- As if there was a network card in the guest and one in the host connected back to back
- The TAP devices may be ports on a host bridge

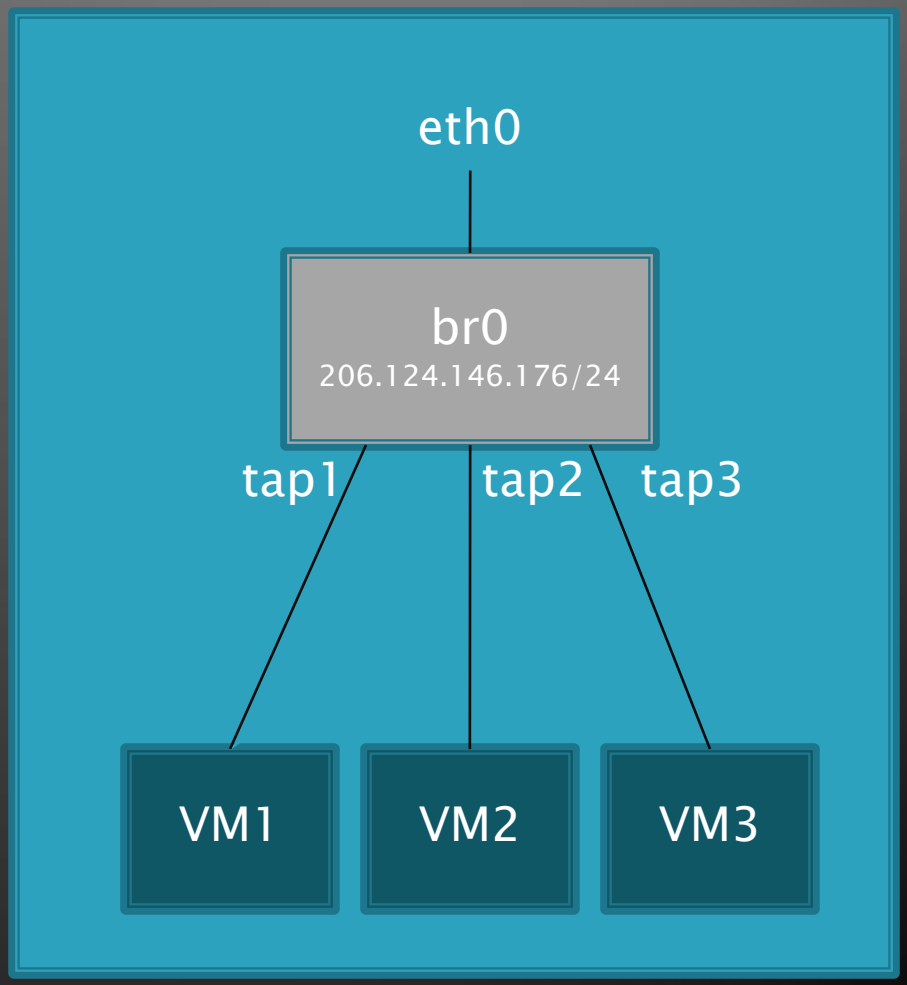


Networking Options – Bridging

- Private Bridge
  - The ‘eth0’ in each VM is associated with a separate VLAN
  - Each VLAN is associated with a TAP device in the host
  - The TAP devices are ports on the bridge
  - Bridge has a private IP address
  - NAT provides VMs access to external networks

Networking Options – Bridging

- Public Bridge
  - The ‘eth0’ in each VM is associated with a separate VLAN
  - Each VLAN is associate with a TAP device in the host
  - The TAP devices are ports on the bridge
  - The host’s network adaptor is also a port on the bridge
  - The Bridge has an IP configuration
  - Guests have public IP addresses



Networking Options – Configuring the Bridge

- http://www.shorewall.net/pub/shorewall/contrib/kvm/kvm
- Install in /etc/init.d/
  - Configure using your distribution's tool (insserv, chkconfig, …)
- Configure by editing:
  - `INTERFACES=""` (Private Bridge)
  - `INTERFACES="eth0"` (Public Bridge)
  - `TAPS="tap1 tap2 tap3 tap4"` (Tap devices)
  - `ADDRESS=192.168.0.254/24` (IP Configuration of the Bridge)
  - `BROADCAST=192.168.0.255`
  - `MODULES="kvm kvm-adm"` (Modules to load)
  - `OWNER=teastep` (Owner of the bridge)
- Loads Modules
- Configures /proc/sys/dev/rtc/
- Allows VMs to be started by ordinary users

Networking – Warning

- If you decide to do it yourself rather than use my script:
  - Use **tunctl** to create your tap devices
  - If you use OpenVPN, you will find that packets flow from the VM to the Host but not in the other direction

Starting VMs

- http://www.shorewall.net/pub/shorewall/contrib/VM
  - Copy to /usr/local/bin/*vmname* for each VM
  - Edit each copy as needed:
    - **VLAN=*vlan*** (Virtual LAN number)
    - **MAC=*mac*** (Media Access Control Address)
    - **DISK=*imagefile*** (Disk Image)
- No sudo – no need to enter a password to start a VM

DEMO