Blueprints

Quick Start Guide for installing and running KVM

Blueprints

Quick Start Guide for installing and running KVM

Note

Before using this information and the product it supports, read the information in "Notices" on page 25.

Second Edition (March 2010)

Contents

Quick Start Guide for installing and running KVM

Kernel-based Virtual Machine (KVM) is a Linux kernel virtualization hypervisor that allows you to host different guest operating systems. If you are looking for a way to create a basic KVM to use as a sandbox or a one-time-only production environment, this Blueprint has the information you need. It shows you how to install KVM on Red Hat Enterprise Linux (RHEL) 5.4 host systems, set up a Linux bridge for guests' network, as well as create a KVM guest and start installation of a guest operating system on it. Key tools and technologies discussed in this demonstration include Kernel-based Virtual Machine (KVM), virt-viewer, virt-manager, virt-install, Linux bridge, and virtualization.

Scope, requirements, and support

You can learn more about this blueprint, including the intended audience, the scope and purpose, the hardware and software requirements for the tasks detailed in this blueprint, and the types of support available to you.

Intended audience

This Blueprint is intended for Linux system administrators and programmers who want to experiment with using KVM, or have a minimal need for using KVM.

Scope and purpose

This Blueprint provides a quick starting point to get a KVM guest up and running. It assumes that the KVM guests to be created will be using local storage and does not discuss other storage options. The process of installing an operating system on a KVM guest is also out of the scope of this paper as the process will be the same as installing on any physical machine. Instructions for installing Windows XP are found in , Setting up Windows XP guest using virt-manager. If you need to create and manage more than a few KVMs, see *The developer's approach to installing and managing KVMs* at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmadv/kvmadvstart.htm.

Test environment

These instructions have been tested on a System x® HS21 blade system running the stock Red Hat Enterprise Linux 5.4 kernel:

```
# uname -a
Linux testmachine.ibm.com 2.6.18-164.el5 #1 SMP Tue Aug 18 15:51:48 EDT 2009
x86_64 x86_64 x86_64 GNU/Linux
```

Hardware requirements

See “Enabling KVM support on your hardware” on page 3 for more information.

Software requirements

The host system should be running Red Hat Enterprise Linux 5.4. For more information about installing Red Hat Enterprise Linux 5.4, see the Red Hat Enterprise Linux 5 Installation Guide at http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Installation_Guide/index.html.

Other considerations

This Blueprint assumes that the created KVM guests will be using local storage. Note however that live migration is only possible if a guest is running on shared storage. Therefore if you would like to have the capability of live migration in the future, set up a shared storage.

Author names

Monza Lui

Kersten Richter

Other contributors

Santwana Samantray

Subrata Modak

IBM® Services

Linux offers flexibility, options, and competitive total cost of ownership with a world class enterprise operating system. Community innovation integrates leading-edge technologies and best practices into Linux.

IBM is a leader in the Linux community with over 600 developers in the IBM Linux Technology Center working on over 100 open source projects in the community. IBM supports Linux on all IBM servers, storage, and middleware, offering the broadest flexibility to match your business needs.

For more information about IBM and Linux, go to ibm.com/linux (https://www.ibm.com/linux)

IBM Support

Questions and comments regarding this documentation can be posted on the developerWorks® Virtualization Blueprint Community Forum:

http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1272

The IBM developerWorks discussion forums let you ask questions, share knowledge, ideas, and opinions about technologies and programming techniques with other developerWorks users. Use the forum content at your own risk. While IBM will attempt to provide a timely response to all postings, the use of this developerWorks forum does not guarantee a response to every question that is posted, nor do we validate the answers or the code that are offered.

Typographic conventions

The following typographic conventions are used in this Blueprint:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Also identifies graphical objects such as buttons, labels, and icons that the user selects.
Italics	Identifies parameters whose actual names or values are to be supplied by the user.
`Monospace`	Identifies examples of specific data values, examples of text like what you might see displayed, examples of portions of program code like what you might write as a programmer, messages from the system, or information you should actually type.

KVM overview

Kernel-based Virtual Machine (KVM) is a hardware-assisted, fully virtualization solution for Linux on x86 hardware that contains virtualization extensions – specifically Intel VT or AMD-V. After you have installed KVM, you can run multiple guests (virtual machines), with each one running a different operating system image. Each of these virtual machines will also have private, virtualized hardware, including a network card, storage, memory, and graphics adapter.

For more information about the list of KVM-supported guest operating systems, go to http://www.linux-kvm.org/page/Guest_Support_Status

In the next sections, you will learn about enabling your system for KVM, installing the KVM-related software, setting up a network bridge on your host system, and finally installing a KVM guest operating system using either virt-manager or virt-install. You will also find some tips about installing your guest operating system.

Enabling KVM support on your hardware

The host machines need to be running either Intel VT or AMD-V chipsets that support hardware-assisted virtualization.

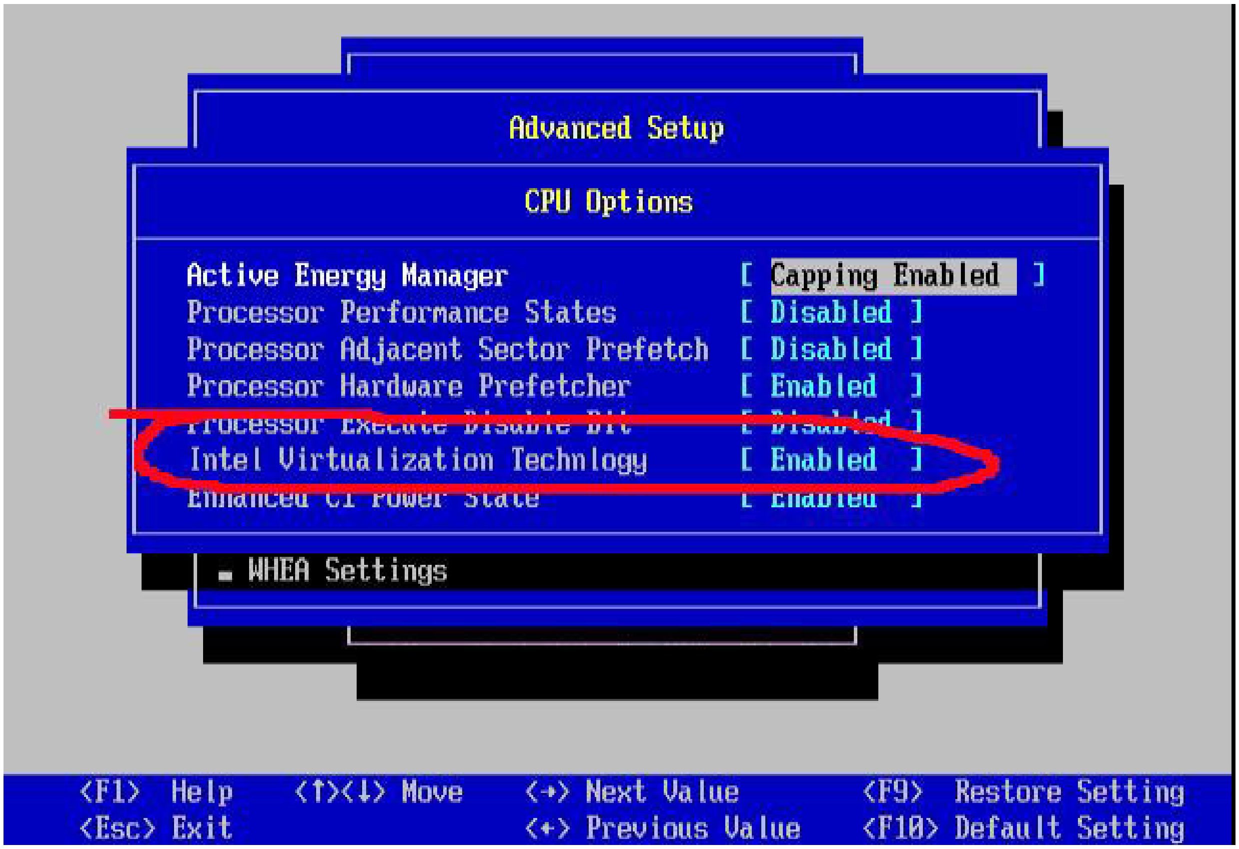
If your system is already installed with Linux, you can also determine if your system CPU supports KVM by running the following command:

```
# grep -E 'vmx|svm' /proc/cpuinfo
flags  : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts
acpi mmx fxsr sse sse2 ss ht tm pbe nx lm constant_tsc arch_perfmon pebs bts pni monitor ds_cpl vmx
est tm2 ssse3 cx16 xtpr lahf_lm ida
flags  : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr
sse sse2 ss ht tm pbe nx lm constant_tsc arch_perfmon pebs bts pni monitor ds_cpl vmx est tm2 ssse3 cx16
xtpr lahf_lm ida
```

If this command returns output, then your system supports KVM. The **vmx** CPU feature flag represents Intel VT chipset while the **svm** flag represents AMD-V. Note which KVM flag was returned as it will be useful for loading the right module later.

Next you need to ensure the KVM-related feature is enabled in the BIOS. Note that IBM delivers System x with this feature already enabled. However, you should still verify that your system is enabled with this feature as the above command will still work even if the feature is disabled. Here are the steps that were used to determine if the HS21 blade test machine running with an Intel Xeon chip had the KVM-related feature enabled:

1. Power off the machine completely.
2. Power on the machine and enter the BIOS by pressing F1 during boot.
3. In the BIOS menu, select **Advanced Step → CPU Options**.



```
Active Energy Manager                [ Capping Enabled  ]
Processor Performance States         [ Disabled ]
Processor Adjacent Sector Prefetch   [ Disabled ]
Processor Hardware Prefetcher        [ Enabled  ]
Processor Execute Disable Bit        [ Disabled ]
Intel Virtualization Technlogy       [ Enabled  ]
Enhanced C1 Power State              [ Enabled  ]
```

4. Make sure the **Intel Virtualization Technology** option is **Enabled**.
5. Save the setting and exit the BIOS.

Now your hardware should be ready for deploying KVM solution.

Installing and configuring KVM-related software

You can install and configure KVM-related software on a host system that already has Red Hat Enterprise Linux 5.4 installed.

If you prefer to install the KVM-related software during the installation of your host system, refer to the Red Hat Enterprise Linux 5 Virtualization guide at http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Virtualization_Guide/.

Installing KVM after host system is installed with Red Hat Enterprise Linux 5.4

Follow these steps to install KVM-related packages to your Red Hat Enterprise Linux 5.4 host system:

Procedure

1. Make sure your RHEL5.4 install is NOT running a Xen kernel. KVM does not work with any Xen kernel. A Xen kernel name looks like this: 2.6.18-164.el5Xen. Check the kernel version your system is running by issuing:

   ```
   # uname -a
   Linux testmachine.ibm.com 2.6.18-164.el5 #1 SMP Tue Aug 18 15:51:48 EDT 2009
   x86_64 x86_64 x86_64 GNU/Linux
   ```

   This output shows the stock kernel that comes with RHEL5.4. The testing for this Blueprint was performed on machines running this kernel.

2. Install the KVM software using yum:

   ```
   # yum install kvm
   ```

3. Install additional virtualization management packages:

   ```
   # yum install virt-manager libvirt libvirt-python python-virtinst
   ```

Configuring KVM after installing the packages

In RHEL5.4, the KVM feature is compiled and provided as modules by default.

About this task

After you have installed the KVM-related packages, load the right KVM modules by following these steps:

Procedure

1. Insert KVM module by running the following command:

   ```
   # modprobe kvm
   ```

2. Insert the chip-specific KVM module by running one of these commands:

   For the AMD chip (svm flag)

   ```
   # modprobe kvm-amd
   ```

   For Intel chip (vmx flag)

   ```
   # modprobe kvm-intel
   ```

   You can verify that the modules are inserted and running.

   ```
   # lsmod|grep kvm
   kvm-intel         86248  3
   kvm                223264  1 kvm_intel
   ```

   If you need help determining which chip your CPU contains, see "Enabling KVM support on your hardware" on page 3.

3. Start the libvirtd daemon service:

   ```
   # /etc/init.d/libvirtd start
   Starting libvirtd daemon:                               [  OK  ]
   # /etc/init.d/libvirtd status
   libvirtd (pid  6584) is running.
   ```

4. Set up libvirtd to start on every reboot:

   ```
   # chkconfig libvirtd on
   ```

What to do next

The KVM-related software is now configured and ready to be used.

Network considerations

After libvirt is installed, a default virtual network (NAT forwarding) is made available to all KVM guests.

A default virtual network means that you get free network without having to configure anything. However, the network only works in one direction, from inside the guest to the outside world (outbound network only). This default virtual network comes with an isolated virtual bridge device, virbr0. It is defaulted to the 192.168.122.x subnet. The host is assigned the 192.168.122.1 address. You can assign an address to your guest from this subnet by manually setting up your network during or after operating system installation.

To verify the availability of this virtual bridge, look at the **ifconfig** command output. Once the KVM-related software is installed, the **ifconfig virbr0** command produces output like:

```
# ifconfig virbr0
virbr0    Link encap:Ethernet  HWaddr 00:00:00:00:00:00
          inet addr:192.168.122.1  Bcast:192.168.122.255  Mask:255.255.255.0
          inet6 addr: fe80::200:ff:fe00:0/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:57 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:10962 (10.7 KiB)
```

Optional: Setting up a network bridge in the host

If you are planning to only access the guest from the KVM host and to access the outside network from the guests, the default networking set up will be sufficient for you, and you can skip this step and go to the next section. If the KVM guests need full network access (to and from an external host), one of the options is to set up a Linux bridge in the host. Note that the Linux bridge configuration does not work in a wireless host environment.

About this task

You should set up this Linux bridge before the guest OS installation so that the bridge is available for selection during the guest operating system installation. Bridged networking allows you to link two Ethernet network segments using packet forwarding technology. More information about Linux bridges can be found at http://en.wikipedia.org/wiki/Network_bridge.

Note: Be careful when you configure the bridge. If you are accessing the host machine using the same network card you are configuring for the bridge, any discrepancy might cause you to lose your network connection.

Before setting up this bridge, make sure that the network card that you want to use for the bridge is providing the network connection you want for your KVM modules and is working. This card should be setup to provide the same networking capability you want your guest KVM to have.

In the following example, **eth0** is the network card used. This card has already been configured for external access.

```
# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
          inet addr:10.10.1.152  Bcast:10.10.1.255  Mask:255.255.255.0
          inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:664 errors:0 dropped:526 overruns:0 frame:0
          TX packets:163 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69635 (68.0 KiB)  TX bytes:25091 (24.5 KiB)
          Interrupt:74 Memory:da000000-da012800
................
```

If your network card is not set up yet, create its network script and save it in the **/etc/sysconfig/network-scripts/** directory with help from this site: http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/s2-networkscripts-interfaces-eth0.html or the libvirt wiki page about writing network scripts at http://wiki.libvirt.org/page/Networking.

Follow these steps to create a public bridge in the host system.

Procedure

1. Back up the corresponding network script file at a different location. Note that this is important because you will need to refer back to the file and also for network recovery. Issue the following command to back up the network script for **ifcfg-eth0** to the /root directory:

```
# cp /etc/sysconfig/network-scripts/ifcfg-eth0 /root/.
```

Note: Do not copy this file to the same network script directory or any of its subdirectories.

2. Create another copy of the network script for defining a Linux bridge associated with the network card to a new file called **/etc/sysconfig/network-scripts/ifcfg-br0**, where *br0* is the name of the bridge. The complete content of the Linux bridge's configuration file will be based on what is already in the working script of your network card.

   ```
   # cd /etc/sysconfig/network-scripts/
   # cp ifcfg-eth0 ifcfg-br0
   ```

   Note: The name of the bridge is arbitrary and you can name it differently as long as you use the same name in the next step inside the network card's script file.

3. Edit both script files to ensure that the network environment remains the same except that now the packets will go through the bridge. Your network card most likely is configured with a static IP address (`BOOTPROTO=static`) or is configured to get an IP address from a DHCP server (`BOOTPROTO=dhcp`).

   - If your network card is configured with static IP address, your original network script file should look similar to this example:

     ```
     DEVICE=eth0
     BOOTPROTO=static
     HWADDR=00:14:5E:C2:1E:40
     IPADDR=10.10.1.152
     NETMASK=255.255.255.0
     ONBOOT=yes
     ```

     The following table shows the contents of the two network configuration scripts after editing was completed. Edit your scripts accordingly.

Table 1. Bridging network files comparison

/etc/sysconfig/network-scripts/ifcfg-eth0	etc/sysconfig/network-scripts/ifcfg-br0

DEVICE=eth0
TYPE=Ethernet
HWADDR=00:14:5E:C2:1E:40
ONBOOT=yes
NM_CONTROLLED=no
BRIDGE=br0
	DEVICE=br0
	TYPE=Bridge
	NM_CONTROLLED=no
	BOOTPROTO=static
	IPADDR=10.10.1.152
	NETMASK=255.255.255.0
	ONBOOT=yes

     In the left column is the network script file for network card (**eth0**). Note that all the information directly about this network card stays the same, such as the `DEVICE` (device name), `TYPE` (device type), `HWADDR` (hardware address), and `ONBOOT` (whether the device will be activated on boot). The `NM_CONTROLLED=no` option was added because both device should not be controlled by the Network Manager for bridge to work. `BRIDGE=br0` is also added to associate this card with the bridge.

     In the right column is the network script for the bridge (**br0**). Note that all information directly related to this bridge is there, such as `DEVICE` (device name), `TYPE` (device type, `Bridge` is case-sensitive and must be added exactly as represented here with an upper case `'B'` and lower case `'ridge'`), and the `NM_CONTROLLED=no` option to disable Network Manager control to this device. The rest are retained from the network card configuration file (`BOOTPROTO`, `IPADDR`, `NETMASK`, and `ONBOOT`). Note that there should not be a hardware address in this file. These values set up the bridge to behave like the network card: the **ifcfg-br0** file acting as an extension of the **ifcfg-eth0** file where the `BRIDGE=br0` is pointing to the **ifcfg-br0** file.

   - If your network card is configured with dynamic IP address, your original network script file should look similar to this example:

     ```
     DEVICE=eth0
     BOOTPROTO=dhcp
     HWADDR=00:14:5E:C2:1E:40
     ONBOOT=yes
     ```

The following table shows the contents of the two network configuration scripts after editing was completed. Edit your scripts accordingly.

Table 2. Bridging network files comparison

/etc/sysconfig/network-scripts/ifcfg-eth0	**/etc/sysconfig/network-scripts/ifcfg-br0**

`DEVICE=eth0`
`TYPE=Ethernet`
`HWADDR=00:14:5E:C2:1E:40`
`ONBOOT=yes`
`NM_CONTROLLED=no`
`BRIDGE=br0`
	`DEVICE=br0`
	`TYPE=Bridge`
	`NM_CONTROLLED=no`
	`BOOTPROTO=dhcp`
	`ONBOOT=yes`

In the left column is the network script file for network card (**eth0**), which looks exactly like the one in the static IP address scenario. Note that all the information directly about this network card stays the same, such as the `DEVICE` (device name), `TYPE` (device type, `Bridge` is case-sensitive and must be added exactly as represented here with an upper case `'B'` and lower case `'ridge'`.), `HWADDR` (hardware address), and `ONBOOT` (whether the device will be activated on boot). The `NM_CONTROLLED=no` option was added because both device should not be controlled by the Network Manager for the bridge to work. `BRIDGE=br0` is also added to associate this card with the bridge.

In the right column is the network script for the bridge (**br0**). Note that similar all information directly related to this bridge is there, such as `DEVICE` (device name), `TYPE` (device type), and the `NM_CONTROLLED=no` option to disable Network Manager control to this device. The rest are retained from the network card configuration file (`BOOTPROTO` and `ONBOOT`). Note that there should not be a hardware address in this file. These values will set up the bridge to behave like the network card: the **ifcfg-br0** file acting as an extension of the **ifcfg-eth0** file where the `BRIDGE=br0` is pointing to the **ifcfg-br0** file.

4. Restart the network to verify that the configuration works. Note that if you configured the network incorrectly, the network connection may drop and you may lose access to your machine. Check the scripts once again carefully, and then restart the network by running the following command:

   ```
   # service network restart
   ```

5. Disable Netfilter processing in the bridged traffic. Append the following lines to the **/etc/sysctl.conf** file:

   ```
   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0
   ```

   Note: For more information about why disabling Netfilter processing is a good security measure, see the "Network isolation options" section of the *Securing KVM guests and the host system* blueprint at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmsec/kvmsecstart.htm.

6. Reload the kernel parameters with the **sysctl** command:

   ```
   # sysctl -p
   net.ipv4.ip_forward = 0
   ...
   net.bridge.bridge-nf-call-ip6tables = 0
   net.bridge.bridge-nf-call-iptables = 0
   net.bridge.bridge-nf-call-arptables = 0
   ```

7. Check that your network still has the same behavior as it did before you made the bridging changes. However, now **ifconfig** has a different output. The following example shows the first two entries of **ifconfig** in the test environment. Note that the bridge **br0** now acts for **eth0**:

   ```
   br0       Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
             inet addr:10.10.1.152  Bcast:10.10.1.255  Mask:255.255.255.0
             inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
             UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
             RX packets:125 errors:0 dropped:0 overruns:0 frame:0
             TX packets:81 errors:0 dropped:0 overruns:0 carrier:0
   ```

```
          collisions:0 txqueuelen:0
          RX bytes:16078 (15.7 KiB)  TX bytes:18542 (18.1 KiB)
eth0      Link encap:Ethernet  HWaddr 00:14:5E:C2:1E:40
          inet6 addr: fe80::214:5eff:fec2:1e40/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:206 errors:0 dropped:0 overruns:0 frame:0
          TX packets:58 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:27308 (26.6 KiB)  TX bytes:13881 (13.5 KiB)
          Interrupt:74 Memory:da000000-da012800
```

You can also see this bridge by running the following command:

```
# brctl show
bridge name     bridge id               STP enabled     interfaces
virbr0          8000.000000000000       yes
br0             8000.000e0cb30550       no              eth0
```

Results

Your Linux bridge should be up and running.

Creating a KVM guest and preparing to install the guest operating system

After you installed the KVM packages and optionally set up a Linux bridge on the host system for the guests to use, you can create a KVM guest and install the operating system on it.

For a complete listing of supported operating systems, see http://www.linux-kvm.org/page/Guest_Support_Status.

The following instructions set up SuSE Linux Enterprise Server 11 and Red Hat Enterprise Linux 5.4 respectively as a guest operating systems. You can very easily adapt the instructions to set up any other supported operating system.

There are several different methods of creating a guest on a KVM host machine. This Blueprint discusses two of them:

- virt-manager: a GUI tool
- virt-install: a command line tool.

If you prefer an advanced, developer's approach to creating and managing KVMs, see *The developer's approach to installing and managing KVMs* at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmadv/liaaikvmadvstart.htm.

Creating KVM guest using virt-manager

The **virt-manager** tool is a GUI tool used to create and manage KVM guests. This is the tool to use if you prefer using a GUI tool and you want to create a simple KVM guest as a sandbox, and a variety of configuration options are not important to you.

About this task

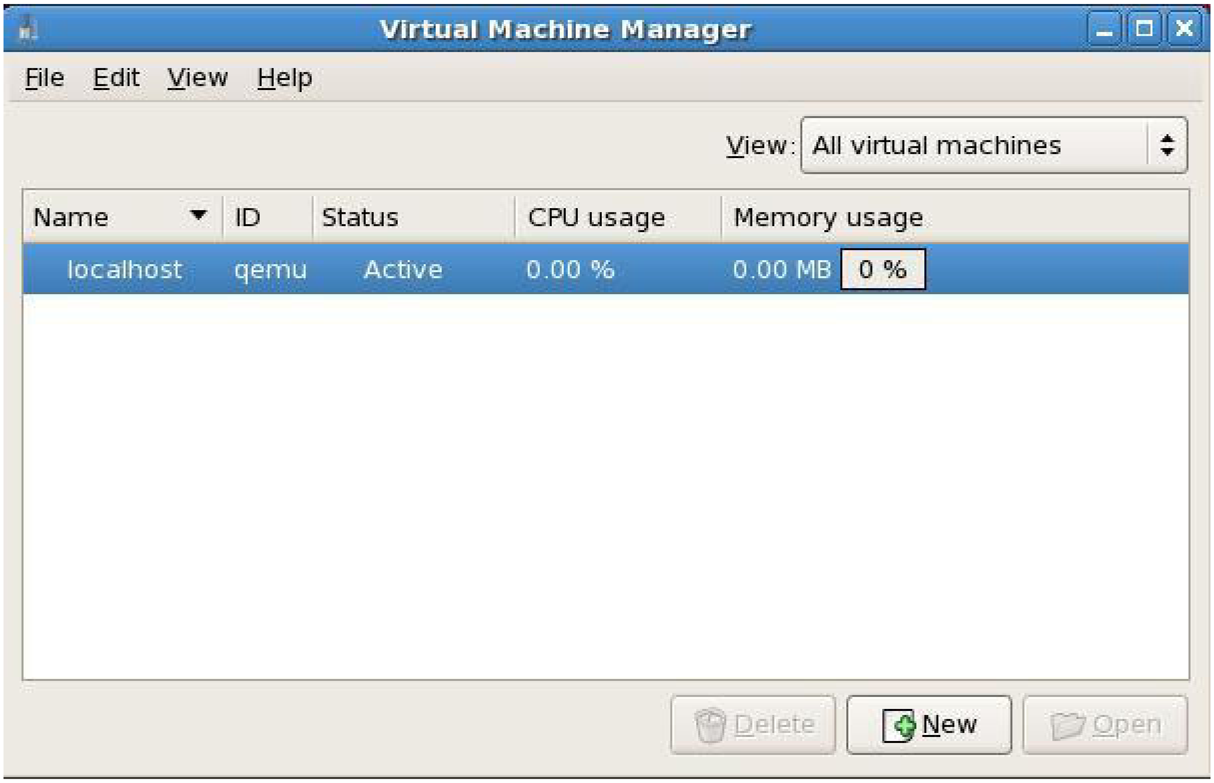
Follow these steps to set up a KVM guest using **virt-manager**. In this example, you will be preparing to install a SLES 11 guest.

Procedure

1. Start your X11 environment. For example, log into your machine using **ssh** with `-X` option.
2. Start **virt-manager** by running the following command:

```
# virt-manager
```

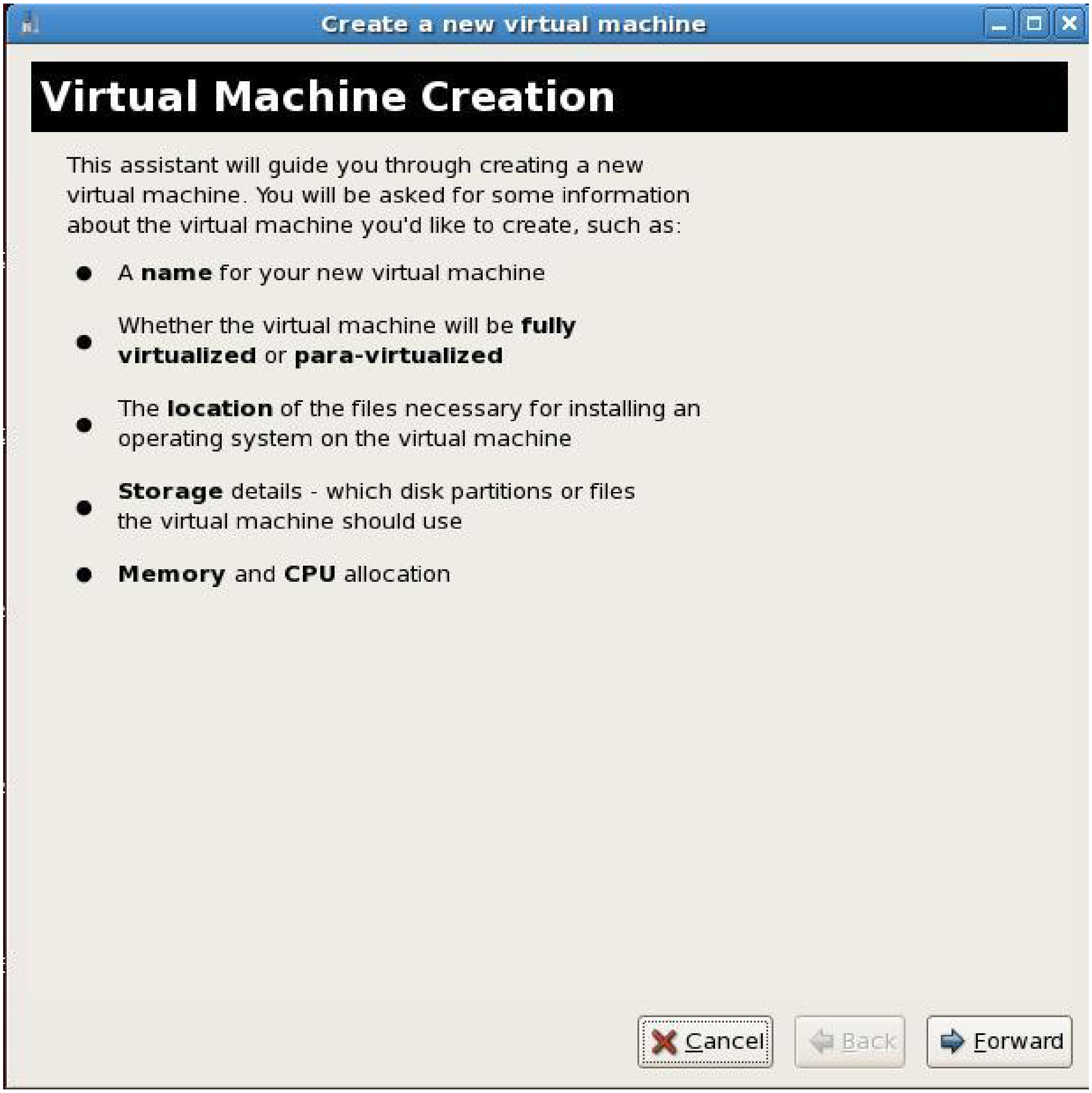
This command opens the following window:



View: All virtual machines

Name	ID	Status	CPU usage	Memory usage
localhost	qemu	Active	0.00 %	0.00 MB 0 %

3. Select the first row, which represents the host domain, and click **New** to create a new guest. This action opens the Virtual Machine Creation dialog as shown below:

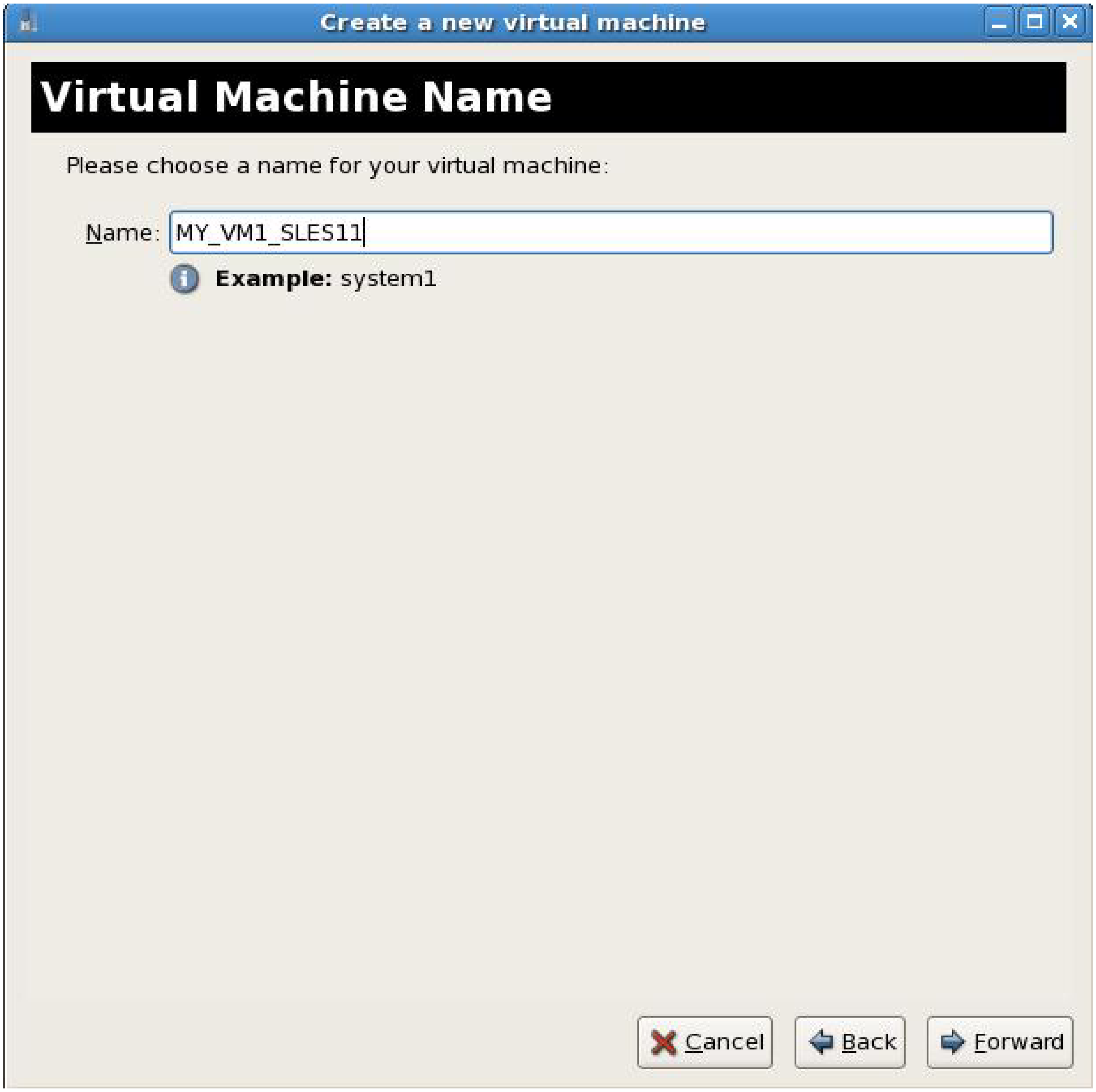


Virtual Machine Creation

This assistant will guide you through creating a new virtual machine. You will be asked for some information about the virtual machine you'd like to create, such as:

- A **name** for your new virtual machine
- Whether the virtual machine will be **fully virtualized** or **para-virtualized**
- The **location** of the files necessary for installing an operating system on the virtual machine
- **Storage** details - which disk partitions or files the virtual machine should use
- **Memory** and **CPU** allocation

4. Click **Forward** to start entering information about your guest.
5. On the Virtual Machine Name dialog, enter a name for your guest. In this example, the virtual machine is called *MY_VM1_SLES11*. Click **Forward**.


Create a new virtual machine

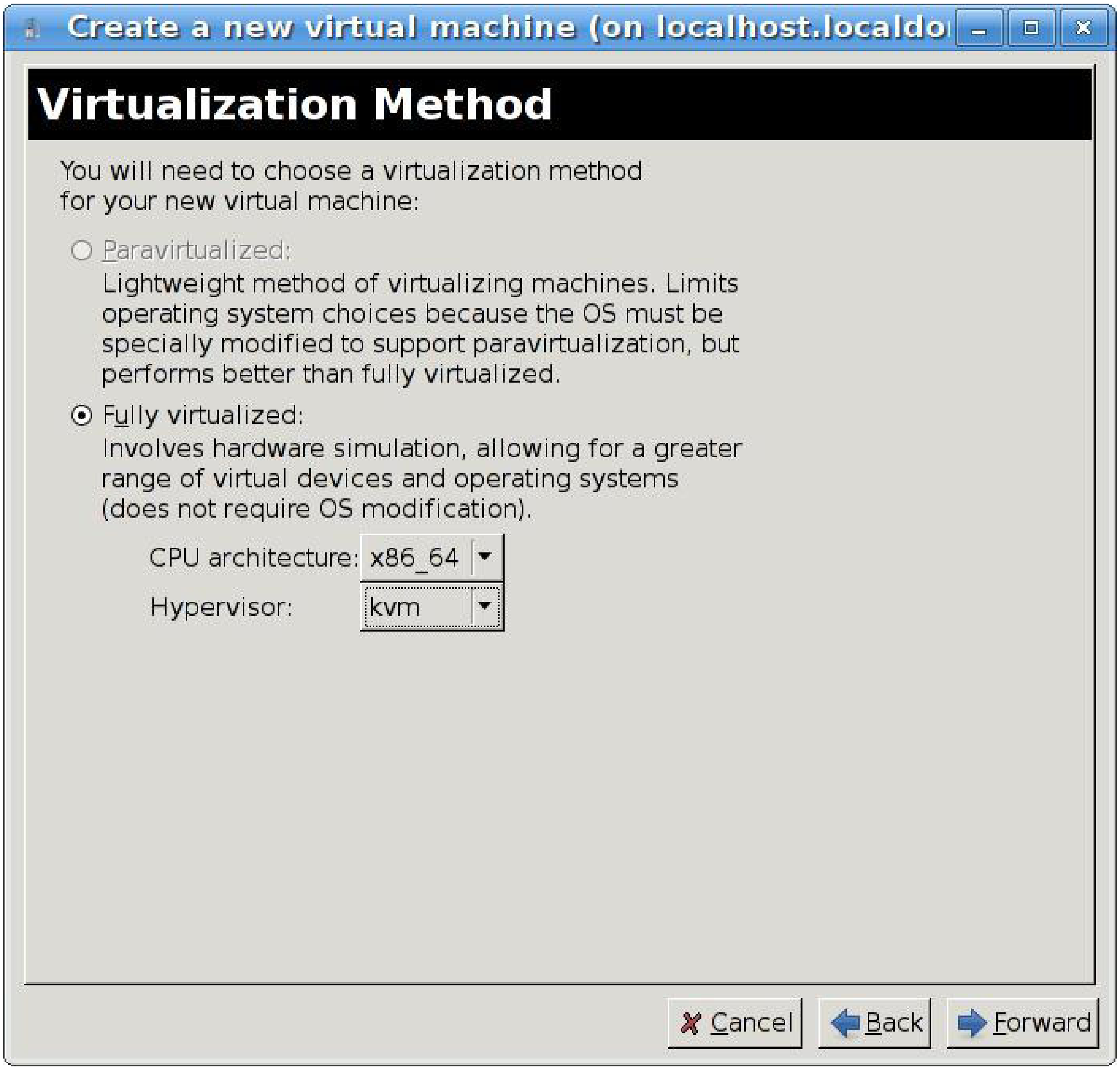
Virtual Machine Name

Please choose a name for your virtual machine:

Name: MY_VM1_SLES11

Example: system1

6. On the Virtualization method dialog, select the `Fully virtualized` option (paravirtualization is not supported in KVM). Choose the CPU architecture you want to simulate. Note that if you are running KVM on an i686 machine, you cannot simulate the x86_64 architecture. This example uses `x86_64`. Accept the default of `kvm` as the choice of Hypervisor. Click **Forward**.



Virtualization Method

You will need to choose a virtualization method for your new virtual machine:

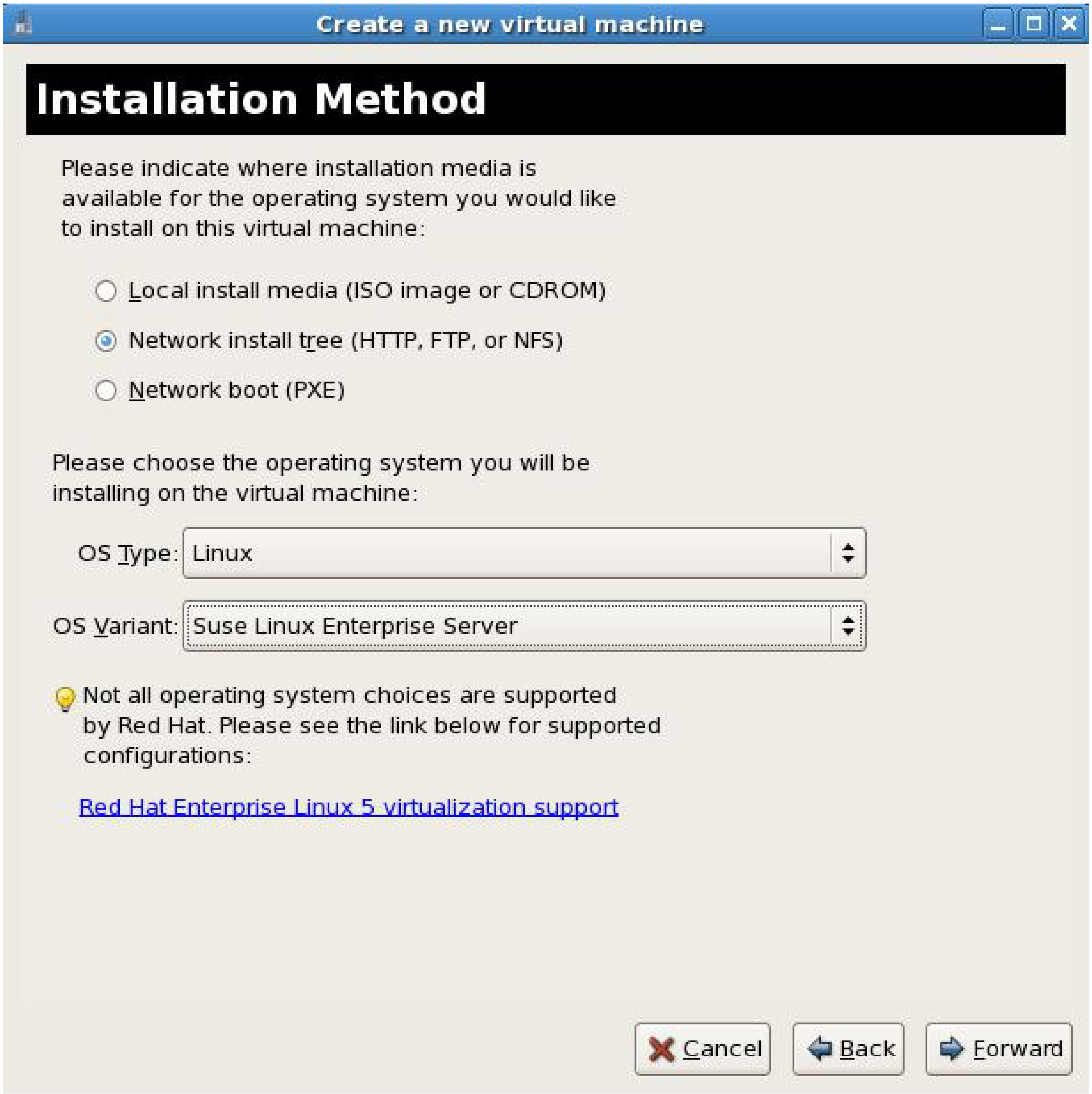
○ Paravirtualized:
Lightweight method of virtualizing machines. Limits operating system choices because the OS must be specially modified to support paravirtualization, but performs better than fully virtualized.

● Fully virtualized:
Involves hardware simulation, allowing for a greater range of virtual devices and operating systems (does not require OS modification).

CPU architecture: x86_64

Hypervisor: kvm

7. On the Installation Method dialog, select your installation method. You also need to select your OS type and OS variant. In this example, an install source made available through a NFS mount is used for installation. Therefore choose `Network Install tree (HTTP, FTP, or NFS)` as the installation media, `Linux` as the OS type, and `Suse Linux Enterprise Server` as the OS variant. Click **Forward**.



Installation Method

Please indicate where installation media is available for the operating system you would like to install on this virtual machine:

○ Local install media (ISO image or CDROM)

● Network install tree (HTTP, FTP, or NFS)

○ Network boot (PXE)

Please choose the operating system you will be installing on the virtual machine:

OS Type: Linux

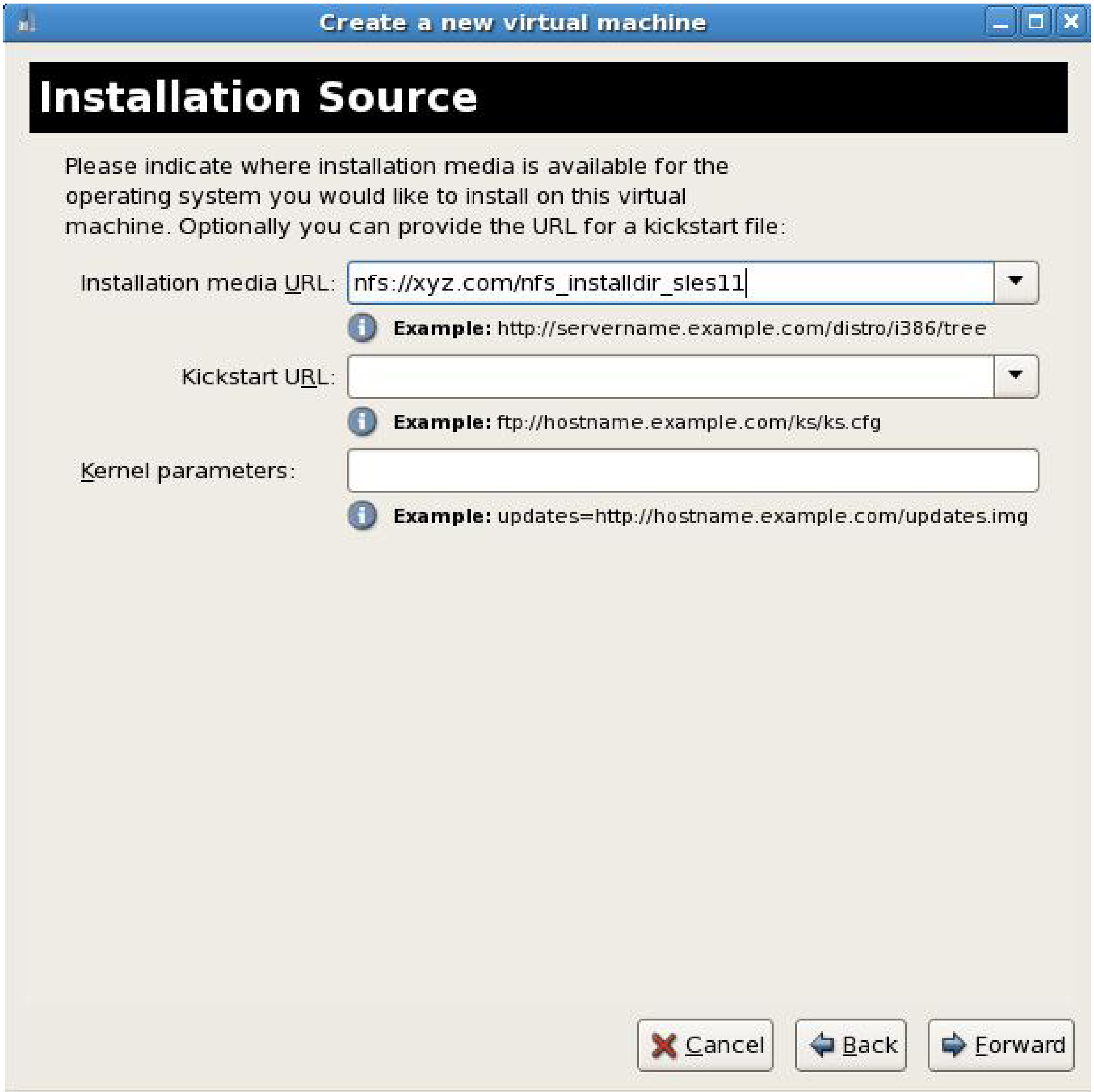
OS Variant: Suse Linux Enterprise Server

Not all operating system choices are supported by Red Hat. Please see the link below for supported configurations:

Red Hat Enterprise Linux 5 virtualization support

8. Depending on which installation media option you chose in the last step, the next step varies. If you are using a Local install or a Network Install, follow this step. For the Network PXE boot option, skip this step and go to step 9.

   If you are using the ISO/CD or network media installation methods, specify the location of your installation media. In this example, the location of the NFS install tree (`nfs://xyz.com/nfs_installdir_sles11`) was specified in the Installation media URL field using the syntax suggested by the tool. Click **Forward**.



Create a new virtual machine

Installation Source

Please indicate where installation media is available for the operating system you would like to install on this virtual machine. Optionally you can provide the URL for a kickstart file:

Installation media URL: nfs://xyz.com/nfs_installdir_sles11

Example: http://servername.example.com/distro/i386/tree

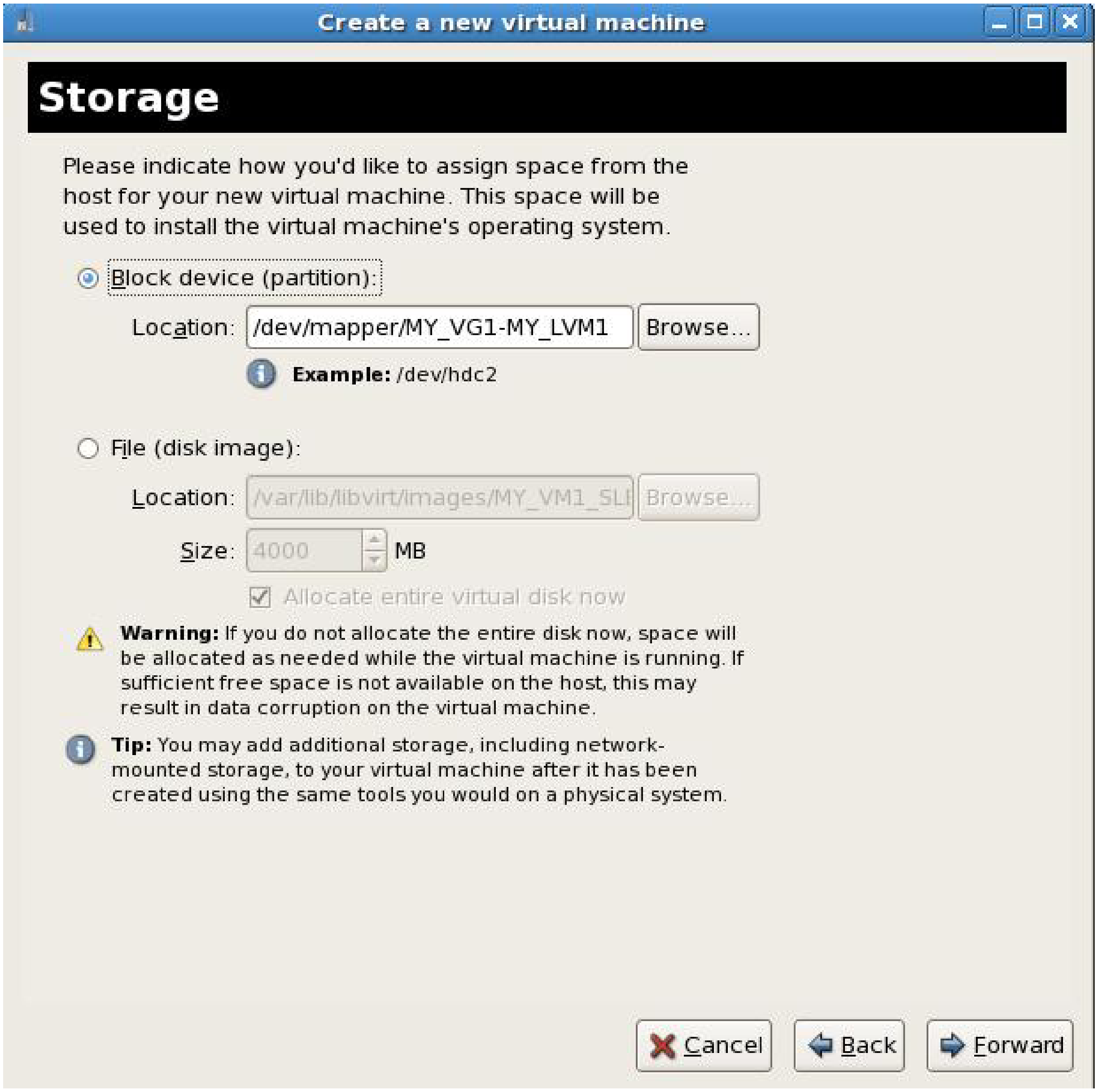
Kickstart URL:

Example: ftp://hostname.example.com/ks/ks.cfg

Kernel parameters:

Example: updates=http://hostname.example.com/updates.img

9. Assign storage for the guest. You can assign an existing **Block device** or a to-be-created **File** (disk image of extension .img). In this example, an existing LVM partition **/dev/mapper/MY_VG1-MY_LVM1** from a local disk was assigned. If you choose to assign a **File**, you will also have to specify its size so that virt-manager can create the file for you. Be reminded to allocate enough disk size for your operating system. Consult the operating system's documentation for minimum amount of disk space needed. Click **Forward**.



Storage

Please indicate how you'd like to assign space from the host for your new virtual machine. This space will be used to install the virtual machine's operating system.

● Block device (partition):

Location: /dev/mapper/MY_VG1-MY_LVM1 Browse...

Example: /dev/hdc2

○ File (disk image):

Location: /var/lib/libvirt/images/MY_VM1_SLI Browse...

Size: 4000 MB

☑ Allocate entire virtual disk now

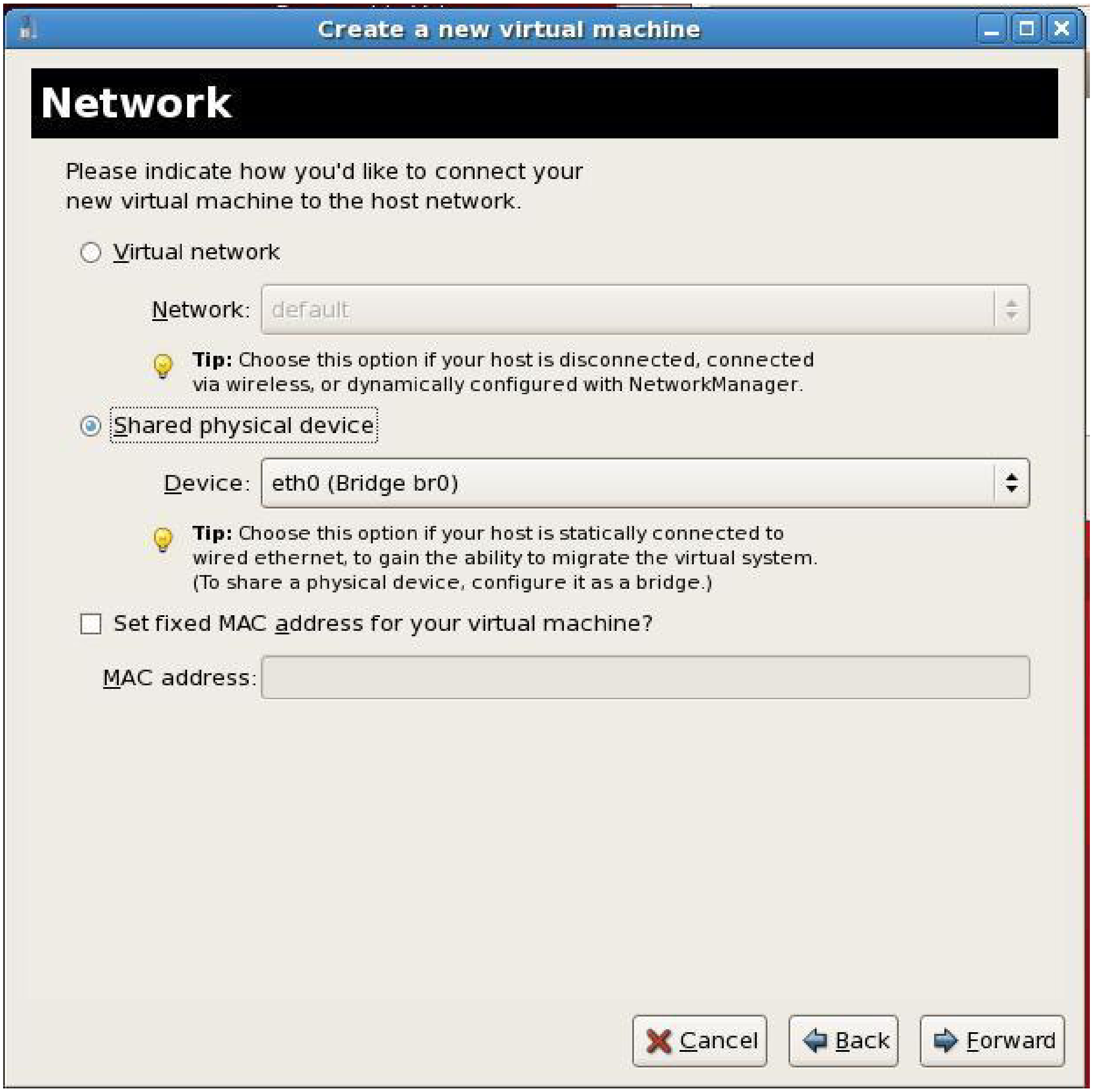
Warning: If you do not allocate the entire disk now, space will be allocated as needed while the virtual machine is running. If sufficient free space is not available on the host, this may result in data corruption on the virtual machine.

Tip: You may add additional storage, including network-mounted storage, to your virtual machine after it has been created using the same tools you would on a physical system.

10. On the Network dialog, you can choose **Virtual network** or **Shared physical device**.

    If you have not set up a Linux bridge, choose the default **Virtual network**.

    In this example, **Shared physical device** was used and `br0` (Bridge to **eth0**) is specified. This bridge was configured earlier in the section “Optional: Setting up a network bridge in the host” on page 6. Click **Forward**.



Network

Please indicate how you'd like to connect your new virtual machine to the host network.

○ Virtual network

Network: default

Tip: Choose this option if your host is disconnected, connected via wireless, or dynamically configured with NetworkManager.

● Shared physical device

Device: eth0 (Bridge br0)

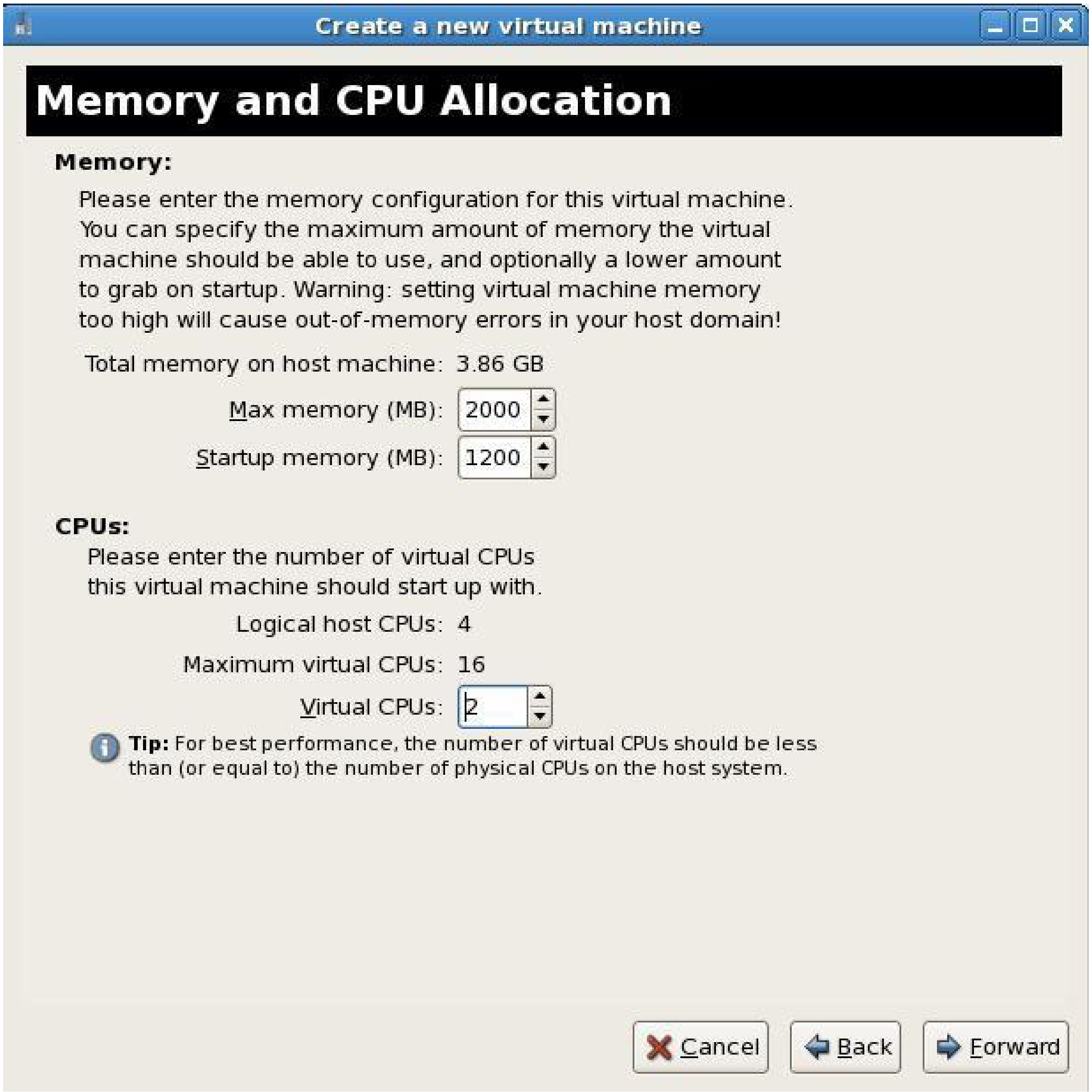
Tip: Choose this option if your host is statically connected to wired ethernet, to gain the ability to migrate the virtual system. (To share a physical device, configure it as a bridge.)

□ Set fixed MAC address for your virtual machine?

MAC address:

11. Assign Memory and CPUs to the guest. The default **Maximum** and **Startup memory** for the guest is 512 MB, and the default number of CPUs is 1. You can edit the resources as per your requirements. Note that you can dynamically update the two resources while the guest is still running if the new assignment is smaller or equal to the maximum amount allocated. In this example, the guest is given `2000 MB` Max memory, `1200 MB` of Startup memory and `2` Virtual CPUs.

    Click **Forward**.



Memory and CPU Allocation

Memory:

Please enter the memory configuration for this virtual machine. You can specify the maximum amount of memory the virtual machine should be able to use, and optionally a lower amount to grab on startup. Warning: setting virtual machine memory too high will cause out-of-memory errors in your host domain!

Total memory on host machine: 3.86 GB

Max memory (MB): 2000

Startup memory (MB): 1200

CPUs:

Please enter the number of virtual CPUs this virtual machine should start up with.

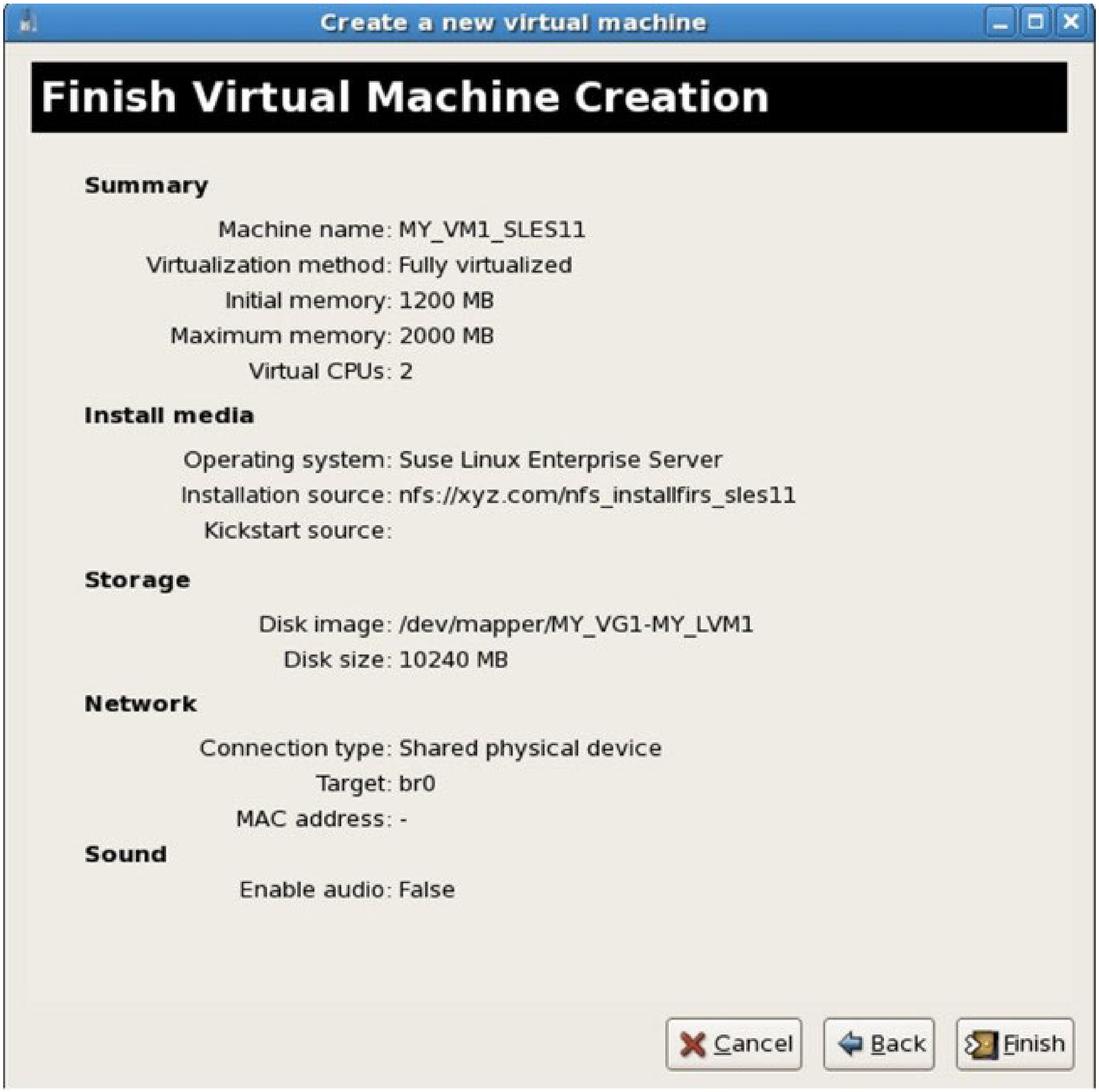
Logical host CPUs: 4

Maximum virtual CPUs: 16

Virtual CPUs: 2

Tip: For best performance, the number of virtual CPUs should be less than (or equal to) the number of physical CPUs on the host system.

12. On the Final Summary page, verify that you have all your setting correct for creating the guest. When you are satisfied, click **Finish**.



Finish Virtual Machine Creation

Summary

Machine name: MY_VM1_SLES11
Virtualization method: Fully virtualized
Initial memory: 1200 MB
Maximum memory: 2000 MB
Virtual CPUs: 2

Install media

Operating system: Suse Linux Enterprise Server
Installation source: nfs://xyz.com/nfs_installfirs_sles11
Kickstart source:

Storage

Disk image: /dev/mapper/MY_VG1-MY_LVM1
Disk size: 10240 MB

Network

Connection type: Shared physical device
Target: br0
MAC address: -

Sound

Enable audio: False

What to do next

After you click **Finish**, the operating system installation dialog opens. Install the operating system just like you would on any other machine. Remember to use *br0* as your network device. For more information about installing your guest, see “Start installing the guest operating system” on page 20.

Creating KVM guest using virt-install

The **virt-install** tool is a command line tool to install KVM guests. Two examples are shown in this section to demonstrate how this tool can be used to create KVMs. For more details about this tool, see its man page.

The following command creates a KVM and opens a SLES 11 installation screen in graphical mode after the SLES 11 installation source is found. This command requires the **virt-viewer** tool that you installed previously, and requires an X-windows environment (for example, you accessed your host using SSH with the -X option).

```
# virt-install \
       --name kvm1 \
       --ram 500 \
       --disk path=/var/lib/libvirt/images/kvm1.img,size=5 \
       --network network:default \
       --accelerate \
       --vnc \
       -c /tmp/SLES11-x86_64-DVD.iso
```

The preceding **virt-install** command creates a KVM named **kvm1** that has 500 MB of memory allocated to it. A local 5 GB file, **/var/lib/libvirt/images/kvm1.img**, is created as this KVM's storage (note that **/var/lib/libvirt/images/** is the default directory for KVM guest images). The default virtual network is used. During installation, KVM kernel acceleration capabilities are used if they are available. A VNC server is made available for connection, and the installation source is an ISO located at **/tmp/SLES11-x86_64-DVD.iso**.

An XML file is created in the **/etc/libvirt/qemu/** directory every time a new KVM guest is created. The XML file is the guest's definition file and is named <Name of guest>.xml. After you run the preceding command, the **/etc/libvirt/qemu/kvm1.xml** file is created. If you are interested, you can examine this file's content. The convenience of using this tool to create a KVM guest is that you do not have to write the XML file to define it. The trade-off is that you have less freedom in customizing your KVM guest. If you want more flexibility in creating your KVM, see *The developer's approach to installing and managing KVMs* at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvmadv/kvmadvstart.htm

The following command creates a KVM and opens an RHEL 5.4 installation screen in text mode (provided by **virt-viewer**) after the RHEL 5.4 installation source is found. This command does not require an X-windows environment.

```
# virt-install \
       --name kvm2 \
       --vcpus 2 \
       --ram 1000 \
       --disk path=/dev/mapper/VolGroup00-LogVol03  \
       --network bridge:br0 \
       --arch i686 \
       -l nfs://10.1.1.212/nfsexport/rhel5.4-server-i386-is/
```

The preceding command creates a KVM named kvm2. Two virtual processors are assigned to this guest, and 1 GB of memory is allocated. An existing LVM partition, **/dev/mapper/VolGroup00-LogVol03**, is used for storage. The Linux bridge, **br0**, that was set up earlier is used. The guest is of i686 architecture. An NFS mount point of host IP 10.1.1.212 and directory **/nfsexport/rhel5.4-server-i386-is/** is specified as the installation source.

Note that because the **--vnc** option is not specified, no VNC server is made available for connection, and the installation screen is provided in text mode only. Note also that KVM kernel acceleration capabilities are not used even if they are available because the **--acelerate** option is not used.

Start installing the guest operating system

Now your guest is created and ready for you to complete the operating system install.

If you used **virt-manager**, you should see an installation screen. However, you have the option to use **virt-viewer** to continue the installation or to recover the installation screen should you lose it.

To install **virt-viewer**, run this command:

```
# yum install virt-viewer
```

To start using **virt-viewer**, enter the following command:

```
# virt-viewer <guest name>
```

After your installation screen appears, you can start installing your KVM guest's operating system as you would on any other physical machine. If you set up the Linux bridge, remember to select it when you set up the network for your guest. If you are using the default virtual network, note that choosing to use DHCP will give the guest a 192.168.122.x address. If you have a preference as to which 192.168.122.x address the guest is assigned, choose manual configuration and use any unused address from this subnet. Specify your subnet mask as 255.255.255.0, and your gateway as 192.168.122.1, the host's address.

To regain control of the mouse when you are using **virt-viewer** or **virt-manager**, press the key combination `Alt-Ctrl`.

Be sure to allow **ssh** connection during the installation if you are planning to access your guest through **ssh**.

The followings are links to how to install RHEL 5, SLES 11, and Windows XP for your reference:

- *Red Hat Enterprise Linux 5 Installation Guide* at http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Installation_Guide/index.html
- *SLES 11 Installation Quick Start* at http://www.novell.com/documentation/sles11/.
- Windows XP at http://publib.boulder.ibm.com/infocenter/lnxinfo/v3r0m0/topic/liaai/kvminstall/KVMinstallXP_pdf.pdf.

After the guest operating system installation

You will be able to continue to connect to your guest using **virt-viewer**. If you have allowed **ssh** connection during the guest OS installation, you should be able to use **ssh** to connect to your guest using either the virtual DHCP assigned address (if you did not set up Linux bridge) or the physical address (if you set up a Linux bridge on the host).

Related information

You can find additional information about the processes and tools described in these procedures.

- Kernel Based Virtual Machine

  http://www.linux-kvm.org/page/Main_Page
- Red Hat Enterprise Linux 5 Virtualization guide

  http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Virtualization_Guide/
- KVM Guest Support Status

  http://www.linux-kvm.org/page/Guest_Support_Status
- KVM Guest Support Status

  http://www.redhat.com/docs/manuals/enterprise/RHEL-5-manual/Deployment_Guide-en-US/s2-networkscripts-interfaces-eth0.html.
- x86 virtualization

  http://en.wikipedia.org/wiki/X86_virtualization
- Red Hat Enterprise Linux 5.4 Installation guide

  http://www.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/5.4/html/Installation_Guide/index.html
- developerWorks Virtualization Blueprint Community Forum

  http://www.ibm.com/developerworks/forums/forum.jspa?forumID=1272

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Dept. LRAS/Bldg. 903
11501 Burnet Road
Austin, TX 78758-3400
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® and ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at Copyright and trademark information at www.ibm.com/legal/copytrade.shtml

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Java and all Java-based trademarks and logos are registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, or service names may be trademarks or service marks of others.

Printed in USA