Kernel-based Virtual Machine (KVM) security

Abstract

You can protect and secure the Kernel-based Virtual Machine (KVM) environment by deploying KVM security features, such as configuring network isolation, securing storage devices, configuring secure remote management, isolating virtual machines with the sVirt service, preventing denial-of-service situations with control groups (cgroups), and protecting data at rest with disk-image encryption.

- Host security
- Trusted computing base
- Host network isolation
- Securing storage devices in a customized local location or on a network file system
- Creating static sVirt labels, and verifying static and dynamic sVirt labels
- Control groups (cgroups)
- Maintaining virtual machine

Host security

Learn about the trusted computing base (TCB), how to configure the network to isolate the host and guest operating systems, and how to customize the storage location for storage devices.

Trusted computing base

The trusted computing base (TCB) is the combination of hardware and software in a computer system that enforces a unified security policy .The TCB usually contains components that are critical to the security of the system, such as hardware, firmware, a security policy, and other components. The TCB controls and authenticates access to system resources and verifies system integrity. In a KVM environment, the overall TCB includes the host TCB, KVM , and QEMU.

The type of the hypervisor does not influence the security quality of the hypervisor. A type 1 hypervisor can be more secure than a type 2 hypervisor and a type 2 hypervisor can be more secure than a type 1 hypervisor. Instead, the TCB of the hypervisor determines the security quality of the hypervisor. More specifically, the size, complexity, design, and implementation of the TCB determine the security quality of the hypervisor. For example, a large hypervisor with a quality design can be more secure than a small hypervisor with a poor design. However, as the size and complexity of the TCB increases, the difficulty of determining the quality of the design and implementation also increases. This difficulty increases exponentially with the amount of code that must be trusted. Therefore, to achieve maximum security, most operating systems reduce the size and complexity of the TCB as much as possible .The size of the TCB directly affects the security quality of the hypervisor. The larger the TCB, the more bugs the TCB likely has and the less secure the hypervisor. To further reduce the size of the TCB in KVM, you can minimize the amount of code that runs in the host operating system. For example, you can disable any network-facing daemons that run in the host operating system. Another reason to reduce the size and complexity of the TCB is to aid the feasibility of formal certification, such as the Common Criteria certification. The size of the TCB directly affects the cost of the TCB assurance process.

Host network configuration

Learn how to isolate the host from the guest operating systems and how use the network functions of KVM to isolate the guest operating systems from each other.

Host network isolation

You can increase network security by configuring one network interface for the host and a separate network interface for the guest operating systems. Typically , tasks that you perform from the host, like starting or stopping virtual machines, require a high security clearance. Generally, there are a small number of trusted users with a high security clearance. Tasks that you perform from guest operating systems require a lower security clearance. Generally, there are a large number of users with a lower security clearance. To increase the security of the host, configure one network interface for the host and a separate network interface for the guest operating systems. In this configuration, the network traffic for the host travels on a different subnet than the network traffic for the guest operating systems. This configuration increases security in the following ways:

- Helps isolate the host from the guest operating systems.

- Helps prevent malicious users with a lower security clearance from breaching a guest operating system and attacking the host or other guest operating systems.

Network isolation options

A standard network security practice is the use of firewalls to block unwanted traffic between two different networks. These firewalls are often applied at the borders of networks as a way to filter potential security threats coming from untrusted sources. However security threats can also originate from behind the firewall when a trusted host is compromised.

Securing storage devices

Learn about the default location for virtual disk images for CentOS 6 and how to change the default location to another directory on the system or to a directory in the network file system.

1. ### Default location for virtual disk images
   Learn about where SELinux stores virtual disk images in CentOS 6, and about the default permissions. SELinux automatically stores images in the /var/lib/libvirt/images directory by default. The root user owns the /var/lib/libvirt/images directory. Therefore, only the root user can create files, delete files, and view a listing of the files in the /var/lib/libvirt/images directory.
   The /var/lib/libvirt/images directory has the following permissions:
   drwx--x--x. root root system_u:object_r:virt_image_t:s0 .
   drwxr-xr-x. root root system_u:object_r:virt_var_lib_t:s0 ..
   -rw-------. root root system_u:object_r:virt_image_t:s0 filename
   where filename is the name of a virtual disk image. For example: f14-live.iso The SELinux label of the /var/lib/libvirt/images directory has the virt_image_t type. Images files not currently in use also have the virt_image_t type. The root user is the only user that can read and write to the virtual disk image. When you start a guest operating system by using the libvirt daemon, SELinux relabels the image to reflect the process labels of the guest operating system. For example:
   -rw-------. qemu qemu system_u:object_r:svirt_image_t:s0:c408,c776 /var/lib/libvirt/images/f14-live.iso
   This example shows that SELinux relabeled the f14-live.iso image as follows:

Relabeled the image with the svirt_image_t type because processes of guest operating systems run in the svirt_t domain.
Relabeled the image with the c408,c776 categories to match the process labels of the guest operating system.

2. Storing virtual disk images in a customized location

You can store virtual disk images in a location other than the /var/lib/libvirt/images directory. The following information applies to KVM environments that are running CentOS 6. To store virtual disk images in a customized location, complete the following steps:
Create the directory in which you want to store the virtual disk images by running the mkdir command as follows:

```
# mkdir directory
```

where directory is the directory in which you want to store the virtual disk images. For example:
/mnt/raid/images
Move the virtual disk images to the directory that you created in Step 1 by running the mv command as follows:

```
# mv /var/lib/libvirt/images/image directory
```

where:
image is a virtual image that you want to move from the /var/lib/libvirt/images directory to the directory that you created in Step 1. For example: rh-5.5.qcow2
directory is the directory that you created in Step 1. For example: /mnt/raid/images
Update the domain.xml file of the libvirt utility:

a. Edit the file by typing the following command:

```
$ virsh edit domainID
```

where domainID is the ID of the domain of the guest operating system whose virtual disk images you want to store.

b. Change the value of source file to reflect the directory that you created in Step 1. For example,

```
<source file='/mnt/raid/images/rh-5.5.qcow2'/>
```

Change the SELinux context for files by running the following command:

```
# semanage fcontext -a -t virt_image_t "directory(/.*)?"
```

KVM security 9where directory is the directory that you created in Step 1. For example:/mnt/raid/images/
Apply the context changes by running the following command:

```
# restorecon -R directory
```

where directory is the directory that you created in Step 1. For example: /mnt/raid/images/

Start the guest operating system.

3. Storing virtual disk images on an NFS mount

You can store virtual disk images on a Network File System (NFS) instead of the /var/lib/libvirt/ images directory. The following information applies to KVM environments that are running CentOS 6 with SELinux and the sVirt service enabled. Before you move the virtual disk images of a guest operating system to an NFS mount, consider the following limitations:

NFS does not support file labels so virtual disk images that are stored on an NFS mount cannot be labeled. Therefore, when using dynamic labeling, you must prevent the sVirt service from relabeling the virtual disk images. When using static labeling, you cannot manually label the virtual disk images.

When you prevent the sVirt service from relabeling the virtual disk images, you also disable dynamic labeling of the virtual disk images. This disablement removes the isolation of the virtual disk images , that are stored on NFS, of other guest operating systems.

To store virtual disk images on an NFS mount, complete the following steps:

Move the virtual disk images to an NFS mount.

Prevent the sVirt service from relabeling the virtual disk images that are stored in NFS by running the following command:

```
# setsebool -P virt_use_nfs on
```

Start the guest operating system.

Remote management

Like many other open source software bundles, KVM-based virtualization favors modularity, flexibility, and open-standard interfaces where the functionality is separated into smaller, self-contained packages. Modularization offers benefits in terms of code reusability, maintainability, and flexibility. However, in terms of security, this design creates more entry-points of concern compared to the monolithic approach. This section discusses how you can secure the KVM solution but still enjoy the benefits of having small, understandable, interconnected pieces. The following information applies to KVM environments where the host is running CentOS 6 with the virtualization software channel enabled. Because of the stock configuration provided with Red Hat Enterprise Linux 5.5 virtualization packages, management of KVM guests cannot be performed remotely by default. This default condition exists because the libvirtd daemon does not create any listening sockets, and the qemu-kvm VNC console only accepts connections from the local host. You have, however, several choices for secure remote management configuration.

Overview of remote management

Learn about the packages included in the VT software channel, the libvirtd deamon, and the management clients of the libvirtd deamon , including virsh, virt-viewer, and virt-manager. Typical Red Hat Enterprise Linux 5 installation source have the following packages (part of the VT software channel):

kvm, kvm-qemu-img

the userspace process that performs machine virtualization, namely Qemu or qemu-kvm (the executable name). The kvm-qemu-img utility is used to create virtual-machine disk images.

kmod-kvm

the Linux kernel module that enables efficient virtualization on hardware that supports it (processors enabled with Intel VT-x or AMD-V).

libvirt, python-virtinst

the userspace management layer, including the libvirtd daemon (a system service responsible for coordinating virtual-machine management), virsh (a full-featured, command-line virtual-machine management utility that connects to the libvirtd daemon), and virt-install (a command-line utility that facilitates virtual-machine installations).

virt-viewer

a tool that displays the graphical (VNC) console of a virtual machine.

virt-manager

a desktop tool that manages virtual machines. The libvirtd daemon is responsible for managing all things related to virtualization. It stores virtual-machine guest configuration, creates the user space qemu-kvm processes that are responsible for machine emulation, and assigns resources such as virtual networks, disk images, or pass-through devices to the guests. With the exception of the VNC console (which is provided by the qemu-kvm process directly), all management of the guests is done through libvirtd daemon clients such as virsh, virt-viewer, and virt-manager. These clients can be initiated remotely through various methods including local UNIX domain sockets, TCP sockets, SSH tunnels, or secure TLS (Transport Layer Security) connections. The latter two will be discussed. Authentication can also be handled using SASL, which is a library and a protocol designed to add pluggable authentication support for connection-based services. In the case of the VNC console, management can be partially performed outside of the libvirtd daemon's control because VNC clients connect directly to the Qemu process.

To allow incoming connections through the KVM host's firewall, you will be asked to open the following ports:

- 22 for SSH
- 16509 for SASL
- 16514 for TLS
- 5900 + VNC screen number for VNC connections

Remote management using SSH tunnels

The most direct (while still secure) way of performing remote management of the libvirtd daemon plus qemu-kvm pair is by using standard SSH sessions as tunnels for communicating libvirt management data. The libvirt API can perform management through an SSH tunnel, so most management software built over libvirt is also capable of using this feature. The following information applies to KVM environments where the host is running CentOS 6 with the virtualization software channel enabled.

No special configuration or setup is needed, provided that the root user (or any other user that is given permission to manage virtual machine guests) is allowed to log on the KVM host using a standard SSH session. This is because connections through an SSH tunnel are seen as local from the libvirtd daemon's point of view. Through the SSH tunnel, you can remotely manage KVM guests using virsh, or access your KVM guests' graphical consoles using the VNC viewer of your choice.

Managing KVM guests remotely with the virsh command

You can manage your remote KVM hosts by connecting to their local libvirtd daemon instance using the SSH method and running virsh commands. Assume you are already able to SSH from your management station to the remote KVM host.

Displaying the remote KVM's VNC console using any VNC client

You can tunnel the VNC console connection manually using a standard SSH session if needed (for example, when the virt-viewer utility is unavailable on the management station). KVM security 13Before you start, ensure that the host is running CentOS 6 with the virtualization software channel enabled.

Complete this procedure to tunnel the VNC console connection manually.

1. Use the virsh command to query the VNC graphical console screen number for the guest you would like to connect to:

```
# virsh -c qemu+ssh://root@kvmhost.company.org/system vncdisplay guest01

root@kvmhost.company.org’s password:

:1
```

2. Open an SSH session forwarding the remote VNC port to a local port. To find out which VNC port is being used by your remote KVM guest, add 5900 (the known base VNC port number) to the screen number from the previous step. In this example, the remote VNC port is 5901. The choice of local port is arbitrary. In this example, assume port 5910 (VNC Port for screen number is free in the management station. The following command therefore forwards port 5901 of kvmhost.company.org to port 5910 of the machine you are on (management station):

```
# ssh -L 5910:localhost:5901 root@kvmhost.company.org
```

```
root@kvmhost.company.org's password:

Last login: Tue Apr 6 15:28:14 2010 from otherclient.company.org
```

3. At another terminal, use your VNC client of choice to connect to the local port 5910, where the KVM host's 5901 port is forwarded to. SSH forwards the port and treats the remote port as if it were a local connection attempt:

```
# vncviewer localhost:10
```

If this configuration is successful, a VNC session of your remote KVM guest graphical console opens.

Remote management using SASL authentication and encryption

SASL provides secure authentication and data encryption while allowing integration with traditional or external authentication and authorization services. Before you start, ensure that the host is running Red Hat Enterprise Linux 5.5 with the virtualization software channel enabled. In its simplest form, SASL can be used to define a database of credentials for authorization. In more complex scenarios, it can work with external authentication services such as Kerberos or LDAP to authenticate users. Either way, the libvirtd daemon guarantees confidentiality by requiring GSSAPI or DIGEST-MD5 as the SASL method if they are not running on top of a secured TLS connection. These methods enable encryption of the data being pushed through.

Remote management using TLS

TLS (Transport Layer Security) connections are made secure by digital signature verification when peers exchange certificates that were previously signed by a recognized certificate authority (CA). The following information applies to KVM environments where the host is running CentOS 6 with the virtualization software channel enabled. In the most common scenario (for example when web browsers connect to web servers), the client application is configured to trust a certain list of CAs. Each server must prove its identity to the client by presenting a certificate signed by those trusted CAs with a matching Subject Name, usually the fully qualified domain name (FQDN) of the server. In other less common scenarios that need improved security, the server also requires the client to present a digitally signed certificate to prove its identity.

The example given in this section demonstrates how to create a local CA, use the local CA to digitallysign server and client certificates, and distribute the certificates for use. Mozilla's nss-util tools (modutiland certutil) are used to create private keys and certificates and the OpenSSL tool (openssl) is used toconvert the private keys and certificates into a usable format for libvirt consumption.

Step 1. Create a local CA in your KVM host

Create an NSS database and configure it to act as the local CA.Before you start, ensure that the host is running Red Hat Enterprise Linux 5.5 with the virtualizationsoftware channel enabled.

1. Log in to your KVM host.

2. In an empty directory, create an empty NSS database. This database will act as the local CA and hold

its keys and certificates:

```
# mkdir libvirt_nssdb

# modutil -create -dbdir libvirt_nssdb/ -force
```

3. Verify that the database was created by listing the configured modules in the database. If it was

created successfully, the command will display an output of modules like the following:

```
# modutil -list -dbdir libvirt_nssdb/

Listing of PKCS #11 Modules

-----------------------------------------------------------

1. NSS Internal PKCS #11 Module

slots: 2 slots attached

status: loaded

slot: NSS Internal Cryptographic Services
```

```
token: NSS Generic Crypto Services

slot: NSS User Private Key and Certificate Services

token: NSS Certificate DB

-----------------------------------------------------------
```

4. Initialize the certificate database. Add a password to protect this database and any information thatcan be used to bypass libvirtd daemon security. Never allow public access to this database.

```
# certutil -N -d libvirt_nssdb/

Enter a password which will be used to encrypt your keys.

The password should be at least 8 characters long,and should contain at least one non-alphabetic character.

Enter new password:

Re-enter password:
```

Step 2. Create the CA private key

Create the CA private key and self-signed certificate.

Note: Because certificates are only valid during a particular period of time, check the system date on thesystem you are using to generate the certificates before you generate any certificates.For x509 version 3 certificates (the default type created by the certutil tool), you must include the keyusage extension (enabled with the -1 flag) and the basic constraint extension (enabled with the -2 flag) todescribe a CA signing key and CA certificate.

1. Run the following certutil command and associated flags and arguments to create the private key:

```
# certutil -S -d libvirt_nssdb/ -n cacert \

-s "CN=My Example Certificate Authority,O=IBM,C=US" \

-t TC,u,u -x -v 12 -1 -2

A random seed must be generated that will be used in the creation of your key. One of the easiest ways to create a random seed is to use the timing of keystrokes on a keyboard.
```

```
To begin, type keys on the keyboard until this progress meter is full. DO NOT USE THE
AUTOREPEAT FUNCTION ON YOUR KEYBOARD!

Continue typing until the progress meter is full:

|***********************************************************|

Finished. Press enter to continue:

Generating key. This may take a few moments...

0 - Digital Signature

1 - Non-repudiation

2 - Key encipherment

3 - Data encipherment

4 - Key agreement

5 - Cert signing key

6 - CRL signing key

Other to finish

> 5

0 - Digital Signature

1 - Non-repudiation

2 - Key encipherment

3 - Data encipherment

4 - Key agreement

5 - Cert signing key

6 - CRL signing key

Other to finish

> 9
```

```
Is this a critical extension [y/N]?

y

Is this a CA certificate [y/N]?

y

Enter the path length constraint, enter to skip [<0 for unlimited path]: >

Is this a critical extension [y/N]?

y
```

2. Verify that the previous command was successful by listing the generated certificate. Both the Certificate Basic Constraints and Certificate Key Usage should be marked as Critical: True:

```
# certutil -L -d libvirt_nssdb/ -n cacertCertificate:

Data:

Version: 3 (0x2)

Serial Number:

00:8f:e9:f8:cd

Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Issuer: "CN=My Example Certificate Authority,O=IBM,C=US"

Validity:

Not Before: Thu Feb 11 09:28:41 2010

Not After : Fri Feb 11 09:28:41 2011

Subject: "CN=My Example Certificate Authority,O=IBM,C=US"

Subject Public Key Info:

Public Key Algorithm: PKCS #1 RSA Encryption

RSA Public Key:

Modulus:
```

```
9e:5a:50:37:af:ee:2b:d7:f0:de:90:fc:c1:c2:65:b2:

...

f4:3d:5b:28:76:a7:d6:58:57:ce:b4:b3:0e:20:5b:1d

Exponent: 65537 (0x10001)

Signed Extensions:

Name: Certificate Basic Constraints

Critical: True

Data: Is a CA with no maximum path length.

Name: Certificate Key Usage

Critical: True

Usages: Certificate Signing

Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Signature:

0f:83:37:2a:25:88:df:c0:1b:8a:16:a0:15:0e:54:de:

...

5e:12:90:4c:5d:4d:30:b2:c4:6c:32:53:a0:12:5a:7e

Fingerprint (MD5):

0C:F7:96:7E:1E:AF:2B:1C:5D:FB:6F:35:C0:4E:44:1C

Fingerprint (SHA1):

10:0A:73:59:8C:ED:1B:54:BE:68:13:5E:44:3A:5B:62:A2:75:EC:F4

Certificate Trust Flags:

SSL Flags:

Valid CA

Trusted CA
```

```
User

Trusted Client CA

Email Flags:

User

Object Signing Flags:

User
```

Step 3. Create server and client certificates

After the CA key and certificate are created, generate the server and client keys and certificates (signed by the CA).In the following instructions, replace kvmhost.company.org with the domain name of your KVM host.

1. Generate the server key and certificate for the KVM host with the following command:

Note: The Common Name (CN) of the subject must match the FQDN or IP address of the server.

```
# certutil -S -d libvirt_nssdb/ -n servercert \

-s "CN=kvmhost.company.org,O=IBM,C=US" -t u,u,u -c cacert -v 12

...
```

2. Verify the properties of the server certificate:

```
# certutil -L -d libvirt_nssdb/ -n servercert

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

00:8f:e9:fd:a6

Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Issuer: "CN=My Example Certificate Authority,O=IBM,C=US"

Validity:
```

```
Not Before: Thu Feb 11 09:39:31 2010

Not After : Fri Feb 11 09:39:31 2011

Subject: "CN=kvmhost.company.org,O=IBM,C=US"

Subject Public Key Info:

Public Key Algorithm: PKCS #1 RSA Encryption

RSA Public Key:

Modulus:

a8:78:68:7d:db:ba:8d:c1:86:9c:be:af:8c:b0:3c:d7:

...

d0:07:0b:6e:2b:8a:d8:dc:2d:8b:bc:fc:c7:18:b2:c7

Exponent: 65537 (0x10001)

...
```

3. Generate the client key and certificate for the management stations, signed by the CA. Unlike Step 1, the CN can be anything that helps you identify the clients that are allowed to connect (see the tls_allowed_dn_list directive in the libvirtd.conf file). But the value of OU will need to match in the rest of the steps. This example uses “CN=admin,OU=virtualization,O=IBM,C=US” as the subject name. Adapt the subject to your environment.

```
# certutil -S -d libvirt_nssdb/ -n clientcert \

-s "CN=admin,OU=virtualization,O=IBM,C=US" -t u,u,u -c cacert -v 12

...
```

4. Verify the generated client certificate:

```
# certutil -L -d libvirt_nssdb/ -n clientcert

Certificate:

Data:

Version: 3 (0x2)

Serial Number:
```

```
00:8f:ea:03:36

Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Issuer: "CN=My Example Certificate Authority,O=IBM,C=US"

Validity:

Not Before: Thu Feb 11 09:51:57 2010

Not After : Fri Feb 11 09:51:57 2011

Subject: "CN=admin,OU=virtualization,O=IBM,C=US"

Subject Public Key Info:

Public Key Algorithm: PKCS #1 RSA Encryption

RSA Public Key:

Modulus:

eb:fc:17:f9:6e:95:14:35:a6:33:06:ba:20:9c:e6:28:

...

7c:09:6d:3f:5b:0a:40:72:fb:49:cb:b7:7b:52:4e:ef

Exponent: 65537 (0x10001)
```

Step 4. Convert certificates to ASCII format

Export and convert the generated CA, server, and client keys and certificates to formats that are consumable by libvirt.

1. Convert the certificates for the CA (cacert.pem), the server (servercert.pem), and the client

(clientcert.pem) to ASCII (RFC1113) format:

```
# certutil -L -d libvirt_nssdb/ -n cacert -a > cacert.pem

# certutil -L -d libvirt_nssdb/ -n servercert -a > servercert.pem

# certutil -L -d libvirt_nssdb/ -n clientcert -a > clientcert.pem
```

2. Export the server private key and the client private key. First convert them to the PKCS#12 format using the pk12util tool, and then to the ASCII format using the openssl tool.

Note: No passwords are being used in this example because the p12 files are temporary containers that are removed at the end.

```
# pk12util -d libvirt_nssdb/ -n servercert -o serverkey.p12

Enter password for PKCS12 file:

Re-enter password:

pk12util: PKCS12 EXPORT SUCCESSFUL

# pk12util -d libvirt_nssdb/ -n clientcert -o clientkey.p12

Enter password for PKCS12 file:

Re-enter password:

pk12util: PKCS12 EXPORT SUCCESSFUL

# openssl pkcs12 -in serverkey.p12 -nodes -nocerts > serverkey.pem

Enter Import Password:

MAC verified OK

# openssl pkcs12 -in clientkey.p12 -nodes -nocerts > clientkey.pem

Enter Import Password:

MAC verified OK

# rm -f serverkey.p12 clientkey.p12

# ls

cacert.pem clientkey.pem servercert.pem

clientcert.pem libvirt_nssdb serverkey.pem
```

The server private key is serverkey.pem and the client private key is clientkey.pem. These key files are sensitive and must be protected from unauthorized access. Remove them from the current directory after they are distributed.

Step 5. Distribute keys and certificates to the server (KVM host)

When certificates and keys for both the server and the client are in a format readable by libvirt, distribute and configure them to be used by TLS.

1. Copy the CA certificate (cacert.pem) to the /etc/pki/CA/ directory. Do not change the file name.

```
# cp cacert.pem /etc/pki/CA/cacert.pem
```

2. Copy the server certificate (servercert.pem) to the /etc/pki/libvirt/ directory, and the server key (serverkey.pem) to the /etc/pki/libvirt/private/ directory. Use the default file names and make sure that only the root user is able to access the private key.

Note: If the keys or certificates are named incorrectly or copied to the wrong directories, the authorization will fail.

```
# mkdir /etc/pki/libvirt

# cp servercert.pem /etc/pki/libvirt/.

# mkdir /etc/pki/libvirt/private

# cp serverkey.pem /etc/pki/libvirt/private/.

# chmod -R o-rwx /etc/pki/libvirt/private
```

3. Verify that the files are placed correctly:

```
# find /etc/pki/CA/*|xargs ls -l

-rw-r--r-- 1 root root 821 Apr 9 15:10 /etc/pki/CA/cacert.pem

# ls -lR /etc/pki/libvirt

/etc/pki/libvirt:

total 16

drwxr-x--- 2 root root 4096 Apr 9 16:35 private

-rw-r--r-- 1 root root 751 Apr 9 15:11 servercert.pem

/etc/pki/libvirt/private:

total 8

-rw-r----- 1 root root 1040 Apr 9 15:11 serverkey.pem
```

Step 6. Distribute keys and certificates to clients (management stations)

For every configured management station, repeat the following procedure. Place a copy of the client

certificate (clientcert.pem) in the /etc/pki/libvirt/ directory and the key (clientkey.pem) in the /etc/pki/libvirt/private/ directory. As usual, restrict access to the client key to the root user. Restricting the client key to root access results in root-only access to the server using this certificate/key pair. This situation is acceptable if the management console requires root access to manage remote environments. Otherwise you can disable client certificate verification in the server configuration and stack another layer of authentication (using SASL) on top of TLS.

1. Log in to the management station.

2. Copy the CA certificate (cacert.pem) from the KVM host to the management station's /etc/pki/CA/

directory. Do not change the file name.

```
# scp kvmhost.company.org:/tmp/cacert.pem /etc/pki/CA/
```

3. Copy the client certificate (clientcert.pem) to the /etc/pki/libvirt/ directory, and the client key (clientkey.pem) to the /etc/pki/libvirt/private/ directory. Use the default file names and make sure that only the root user is able to access the private key.

Note: If the keys or certificates are named incorrectly or copied to the wrong directories, the

authorization will fail.

```
# cp clientcert.pem /etc/pki/libvirt/.

# mkdir /etc/pki/libvirt/private

# cp clientkey.pem /etc/pki/libvirt/private/.

# chmod -R o-rwx /etc/pki/libvirt/private
```

4. Verify that the files are placed correctly:

```
# ls -lR /etc/pki/libvirt/

/etc/pki/libvirt/:

total 8
```

```
-rw-r--r-- 1 root root 767 2010-04-09 13:54 clientcert.pem

drwxr-xr-- 2 root root 4096 2010-04-09 14:00 private

/etc/pki/libvirt/private:

total 4

-rw-r--r-- 1 root root 1044 2010-04-09 13:55 clientkey.pem
```

Step 7. Edit the libvirtd daemon configuration

Make sure that the libvirtd daemon is listening to network connections and that the libvirtd.conf file specifies allowed subjects and client certificates.

1. Log in to the KVM host.

2. If you have not already done so, make a copy of the /etc/sysconfig/libvirtd file and the

/etc/libvirt/libvirtd.conf file.

3. Edit the /etc/sysconfig/libvirtd file and ensure that the --listen argument is passed to the libvirtd daemon. This step ensures that the libvirtd daemon is listening to network connections. The following example shows the changes from the original file:

```
--- libvirtd.orig 2010-02-04 11:41:37.000000000 -0600

+++ libvirtd 2010-02-04 11:31:33.000000000 -0600

@@ -3,7 +3,7 @@

# Listen for TCP/IP connections

# NB. must set up TLS/SSL keys prior to using this

-#LIBVIRTD_ARGS="--listen"

+LIBVIRTD_ARGS="--listen"

...
```

4. Edit the /etc/libvirt/libvirtd.conf file and configure a set of allowed subjects using the tls_allowed_dn_list directive in the libvirtd.conf file. The following example shows the changes from the original file. It restricts acceptable client certificates to those with the "O=IBM,OU=virtualization" values, while the country (C) and common name (CN) might be assigned any value:

```
--- libvirtd.conf.orig 2010-02-04 11:28:32.000000000 -0600

+++ libvirtd.conf 2010-02-10 11:33:02.000000000 -0600

@@ -210,7 +210,7 @@

##

By default, no DN's are checked

#tls_allowed_dn_list = ["DN1", "DN2"]

-

+tls_allowed_dn_list = ["C=*,O=IBM,OU=virtualization,CN=*"]

...
```

5. Restart the libvirtd daemon service for changes to take effect:

```
# service libvirtd restart

Stopping libvirtd daemon: [ OK ]

Starting libvirtd daemon: [ OK ]
```

Step 8. Change the firewall configuration

Allow TLS-authenticated connections through the firewall of the KVM host by opening its TCP port 16514.

1. Access the security level configuration:

```
# system-config-securitylevel
```

2. Click Other Ports and add TCP port 16514 as a trusted port.

Step 9. Verify that remote management is working

Clients with a valid certificate and subject should be able to connect to the KVM host libvirtd daemon to perform virtual-machine guest administration. To verify that clients are able to connect, run the following commands in your management station (client):

```
# sudo virsh -c qemu+tls://kvmhost.company.org/system list --all

Id Name State
```

```
----------------------------------

- guest01 shut off

- guest02 running

# sudo virsh -c qemu+tls://kvmhost.company.org/system start guest01

Domain guest01 started
```

Administration tasks

- Automatically starting guests
- Using qemu-img
- Verifying virtualization extensions
- Setting KVM processor affinities
- Generating a new unique MAC address
- Improving guest response time
- Disable SMART disk monitoring for guests
- Configuring a VNC Server
- Gracefully shutting down guests
- Virtual machine timer management with libvirt

1. Automatically starting guests

This section covers how to make guests start automatically during the host system's boot phase. This example uses virsh to set a guest, TestServer, to automatically start when the host boots.

```
# virsh autostart TestServer
Domain TestServer marked as autostarted
```

The guest now automatically starts with the host.To stop a guest automatically booting use the --disable parameter

```
# virsh autostart --disable TestServer
Domain TestServer unmarked as autostarted
```

The guest no longer automatically starts with the host.

2. Using qemu-img

The qemu-img command line tool is used for formatting, modifying and verifying various file systems used by KVM. qemu-img options and usages are listed below.

Check Perform a consistency check on the disk image filename.

```
# qemu-img check [-f format] filename
```

Commit

Commit any changes recorded in the specified file (filename) to the file's base image with the qemu-img commit command. Optionally, specify the file's format type (format).

```
# qemu-img commit [-f format] filename
```

Convert

The convert option is used to convert one recognized image format to another image format.

Command format:

```
# qemu-img convert [-c] [-f format] [-o options] [-O output_format] filename output_filename
```

Convert the disk image filename to disk image output_filename using format output_format. The disk image can be optionally compressed with the -c option, or encrypted with the -o option by setting -o encryption. Note that the options available with the -o parameter differ with the selected format. Only the qcow2 format supports encryption or compression. qcow2 encryption uses the AES format with secure 128-bit keys. qcow2 compression is read-only, so if a compressed sector is converted from qcow2 format, it is written to the new format as uncompressed data.Image conversion is also useful to get a smaller image when using a format which can grow, such as qcow or cow. The empty sectors are detected and suppressed from the destination image.

Create

Create the new disk image filename of size size and format format.

```
# qemu-img create [-f format] [-o options] filename [size]
```

If a base image is specified with -o backing_file=filename, the image will only record differences between itself and the base image. The backing file will not be modified unless you use the commit command. No size needs to be specified in this case.

Info

The info parameter displays information about a disk image filename. The format for the info option is as follows:

```
# qemu-img info [-f format] filename
```

This command is often used to discover the size reserved on disk which can be different from the displayed size. If snapshots are stored in the disk image, they are displayed also.

Rebase

Changes the backing file of an image.

```
# qemu-img rebase [-f format] [-u] -b backing_file [-F backing_format] filename
```

The backing file is changed to backing_file and (if the format of filename supports the feature), the backing file format is changed to backing_format.

There are two different modes in which rebase can operate: Safe and Unsafe. Safe mode is used by default and performs a real rebase operation. The new backing file may differ from the old one and the qemu-img rebase command will take care of keeping the guest-visible content of filename unchanged. In order to achieve this, any clusters that differ between backing_file and old backing file of filename are merged into filename before making any changes to the backing file. Note that safe mode is an expensive operation, comparable to converting an image. The old backing file is required for it to complete successfully. Unsafe mode is used if the -u option is passed to qemu-img rebase. In this mode, only the backing file name and format of filename is changed, without any checks taking place on the file contents. Make sure the new backing file is specified correctly or the guest-visible content of the image will be corrupted. This mode is useful for renaming or moving the backing file. It can be used without an accessible old backing file. For instance, it can be used to fix an image whose backing file has already been moved or renamed.

Resize

Change the disk image filename as if it had been created with size size. Only images in raw format can be resized regardless of version. Red Hat Enterprise Linux 6.1 and later adds the ability to grow (but not shrink) images in qcow2 format. Use the following to set the size of the disk image filename to size bytes:

```
# qemu-img resize filename size
```

You can also resize relative to the current size of the disk image. To give a size relative to the current size, prefix the number of bytes with + to grow, or - to reduce the size of the disk image by that number of bytes. Adding a unit suffix allows you to set the image size in kilobytes (K), megabytes (M), gigabytes (G) or terabytes (T).

```
# qemu-img resize filename [+|-]size[K|M|G|T]
```

Snapshot

List, apply, create, or delete an existing snapshot (snapshot) of an image (filename).

```
# qemu-img snapshot [ -l | -a snapshot | -c snapshot | -d snapshot ] filename
```

-l lists all snapshots associated with the specified disk image. The apply option, -a, reverts the disk image (filename) to the state of a previously saved snapshot. -c creates a snapshot (snapshot) of an image (filename). -d deletes the specified snapshot.

3. Verifying virtualization extensions

Use this section to determine whether your system has the hardware virtualization extensions. Virtualization extensions (Intel VT-x or AMD-V) are required for full virtualization.

1:Run the following command to verify the CPU virtualization extensions are available:

```
$ grep -E 'svm|vmx' /proc/cpuinfo
```

2: Analyze the output.

The following output contains a vmx entry indicating an Intel processor with the Intel VT-x extension:

```
flags   : fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
dts acpi mmx fxsr sse sse2 ss ht  tm syscall lm constant_tsc pni
monitor ds_cpl
vmx est tm2 cx16 xtpr lahf_lm
```

The following output contains an svm entry indicating an AMD processor with the AMD-V extensions:

```
flags   :  fpu tsc msr pae mce cx8 apic mtrr mca cmov pat pse36 clflush
mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt lm 3dnowext 3dnow pni
cx16
lahf_lm cmp_legacy svm cr8legacy ts fid vid ttp tm stc
```

If any output is received, the processor has the hardware virtualization extensions. However in some circumstances manufacturers disable the virtualization extensions in BIOS. The "flags:" output content may appear multiple times, once for each hyper thread, core or CPU on the system.
The virtualization extensions may be disabled in the BIOS. If the extensions do not appear or full virtualization does not work refer to Procedure 22.1, “Enabling virtualization extensions in BIOS”.
3: Ensure KVM subsystem is loaded
As an additional check, verify that the kvm modules are loaded in the kernel:

```
# lsmod | grep kvm
```

If the output includes kvm_intel or kvm_amd then the kvm hardware virtualization modules are loaded and your system meets requirements.

4. Setting KVM processor affinities

By default, libvirt provisions guests using the hypervisor's default policy. For most hypervisors, the policy is to run guests on any available processing core or CPU. There are times when an explicit policy may be better, particularly for systems with a NUMA (Non-Uniform Memory Access) architecture. A guest on a NUMA system can be pinned to a processing core so that its memory allocations are always local to the node it is running on. This avoids cross-node memory transports which have less bandwidth and can significantly degrade performance. On non-NUMA systems some form of explicit placement across the hosts’ sockets, cores and hyperthreads may be more efficient.
Identifying CPU and NUMA topology
The first step in deciding which policy to apply is to determine the host’s memory and CPU topology. The virsh nodeinfo command provides information about how many sockets, cores and hyperthreads are attached to a host.

```
# virsh nodeinfo
CPU model:          x86_64
CPU(s):             8
CPU frequency:      1000 MHz
CPU socket(s):      2
Core(s) per socket:  4
Thread(s) per core:  1
```

```
NUMA cell(s):        2
Memory size:         8179176 kB
```

This output shows that the system has eight CPU cores and two sockets. Each CPU socket has four cores. This splitting of CPU cores across multiple sockets suggests that the system has Non-Uniform Memory Access (NUMA) architecture.

5. Generating a new unique MAC address

In some case you will need to generate a new and unique MAC address for a guest. There is no command line tool available to generate a new MAC address at the time of writing. The script provided below can generate a new MAC address for your guests. Save the script to your guest as macgen.py. Now from that directory you can run the script using ./macgen.py and it will generate a new MAC address. A sample output would look like the following:

```
$ ./macgen.py
00:16:3e:20:b0:11
#!/usr/bin/python
# macgen.py script to generate a MAC address for guests
#
import random
#
def randomMAC():
mac = [ 0x00, 0x16, 0x3e,
random.randint(0x00, 0x7f),
random.randint(0x00, 0xff),
random.randint(0x00, 0xff) ]
return ':'.join(map(lambda x: "%02x" % x, mac))
#
print randomMAC()
```

6. Improving guest response time

Guests can sometimes be slow to respond with certain workloads and usage patterns. Examples of situations which may cause slow or unresponsive guests:

Severely overcommitted memory.
Overcommitted memory with high processor usage
Other (not qemu-kvm processes) busy or stalled processes on the host.

These types of workload may cause guests to appear slow or unresponsive. Usually, the guest's memory is eventually fully loaded into the host's main memory from swap. Once the guest is loaded in main memory, the guest will perform normally. Note, the process of loading a guest from swap to main memory may take several seconds per gigabyte of RAM assigned to the guest, depending on the type of storage used for swap and the performance of the components.

KVM guests function as Linux processes. Linux processes are not permanently kept in main memory (physical RAM). The kernel scheduler swaps process memory into virtual memory (swap). Swap, with conventional hard disk drives, is thousands of times slower than main memory in modern computers. If a guest is inactive for long periods of time, the guest may be placed into swap by the kernel.

KVM guests processes may be moved to swap regardless of whether memory is overcommitted or overall memory usage. Using unsafe overcommit levels or overcommitting with swap turned off guest processes or other critical processes may be killed by the pdflush kernel function. pdflush automatically kills processes to keep the system from crashing and to free up memory. Always ensure the host has sufficient swap space when overcommitting memory.

Turning off swap

Swap usage can be completely turned off to prevent guests from being unresponsive while they are moved back to main memory. Swap may also not be desired for guests as it can be resource-intensive on some systems .The swapoff command can disable all swap partitions and swap files on a system.

```
# swapoff -a
```

To make this change permanent, remove swap lines from the /etc/fstab file and restart the host system.

Using SSDs for swap

Using Solid State Drives (SSDs) for swap storage may improve the performance of guests. Using RAID arrays, faster disks or separate drives dedicated to swap may also improve performance.

Disable SMART disk monitoring for guests

SMART disk monitoring can be safely disabled as virtual disks and the physical storage devices are managed by the host.

```
# service smartd stop
# chkconfig --del smartd
```

7. Configuring a VNC Server

To configure a VNC server use the Remote Desktop application in System > Preferences. Alternatively, you can run the vino-preferences command. The following steps set up a dedicated VNC server session:

1: Edit the ~/.vnc/xstartup file to start a GNOME session whenever vncserver is started. The first time you run the vncserver script it will ask you for a password you want to use for your VNC session.

2: A sample xstartup file:

```
#!/bin/sh
# Uncomment the following two lines for normal desktop:
# unset SESSION_MANAGER
# exec /etc/X11/xinit/xinitrc
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
[ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
#xsetroot -solid grey
#vncconfig -iconic &
#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &
#twm &
if test -z "$DBUS_SESSION_BUS_ADDRESS" ; then
eval `dbus-launch --sh-syntax –exit-with-session`
echo "D-BUS per-session daemon address is: \
$DBUS_SESSION_BUS_ADDRESS"
fi
exec  gnome-session
```

8. Gracefully shutting down guests

   Installing virtualized Red Hat Enterprise Linux 6 guests with the Minimal installation installation option will not install the acpid package. Without the acpid package, the Red Hat Enterprise Linux 6 guest does not shut down when the virsh shutdown command is executed. The virsh shutdown command is designed to gracefully shut down guests. Using virsh shutdown is easier and safer for system administration. Without graceful shut down with the virsh shutdown command a system administrator must log into a guest manually or send the Ctrl-Alt-Del key combination to each guest.

Virtual machine timer management with libvirt

Accurate time keeping on guests is a key challenge for virtualization platforms. Different hypervisors attempt to handle the problem of time keeping in a variety of ways. Libvirt provides hypervisor independent configuration settings for time management, using the <clock> and <timer> elements in the domain XML. The domain XML can be edited using the virsh edit command. See Editing a guest's configuration file for details.

Storage concepts

Local storage

Local storage is directly attached to the host server. Local storage includes local directories, directly attached disks, and LVM volume groups on local storage devices.

Networked storage

Networked storage covers storage devices shared over a network using standard protocols. Networked storage includes shared storage devices using Fibre Channel, iSCSI, NFS, GFS2, and SCSI RDMA protocols. Networked storage is a requirement for migrating guest virtualized guests between hosts.

Storage pools

A storage pool is a file, directory, or storage device managed by libvirt for the purpose of providing storage to virtualized guests. Storage pools are divided into storage volumes that store virtualized guest images or are attached to virtualized guests as additional storage. libvirt uses a directory-based storage pool, the /var/lib/libvirt/images/ directory, as the default storage pool. The default storage pool can be changed to another storage pool.

- Local storage pools - Local storage pools are directly attached to the host server. Local storage pools include local directories, directly attached disks, physical partitions, and LVM volume groups on local devices. Local storage pools are useful for development, testing and small deployments that do not require migration or large numbers of virtualized guests. Local storage pools are not suitable for many production environments as local storage pools do not support live migration.
- Networked (shared) storage pools - Networked storage pools cover storage devices shared over a network using standard protocols. Networked storage is required for migrating guest virtualized guests between hosts. Networked storage pools are managed by libvirt. Storage pools are divided into storage volumes. Storage volumes are an abstraction of physical partitions, LVM logical volumes, file-based disk images and other storage types handled by libvirt. Networked storage pools cover storage devices

shared over a network using standard protocols. Supported protocols for networked storage pools:

- ü Fibre Channel-based LUNs
- ü iSCSI
- ü NFS
- ü GFS2
- ü SCSI RDMA protocols (SCSI RCP), the block export protocol used in Infiniband and 10GbE iWARP adapters.

Networked storage is a requirement for migrating guest virtualized guests between hosts. Networked storage pools are managed by libvirt.

Storage volumes

Storage pools are divided into storage volumes. Storage volumes are an abstraction of physical partitions , LVM logical volumes, file-based disk images and other storage types handled by libvirt. Storage volumes are presented to virtualized guests as local storage devices regardless of the underlying hardware.

Storage pools

Partition-based storage pools

This section covers using a pre-formatted block device, a partition, as a storage pool.For the following examples, a host has a 500GB hard drive (/dev/sdc) partitioned into one 500GB,ext4 formatted partition (/dev/sdc1). We set up a storage pool for it using the procedure below.

Creating a partition-based storage pool using virt-manager

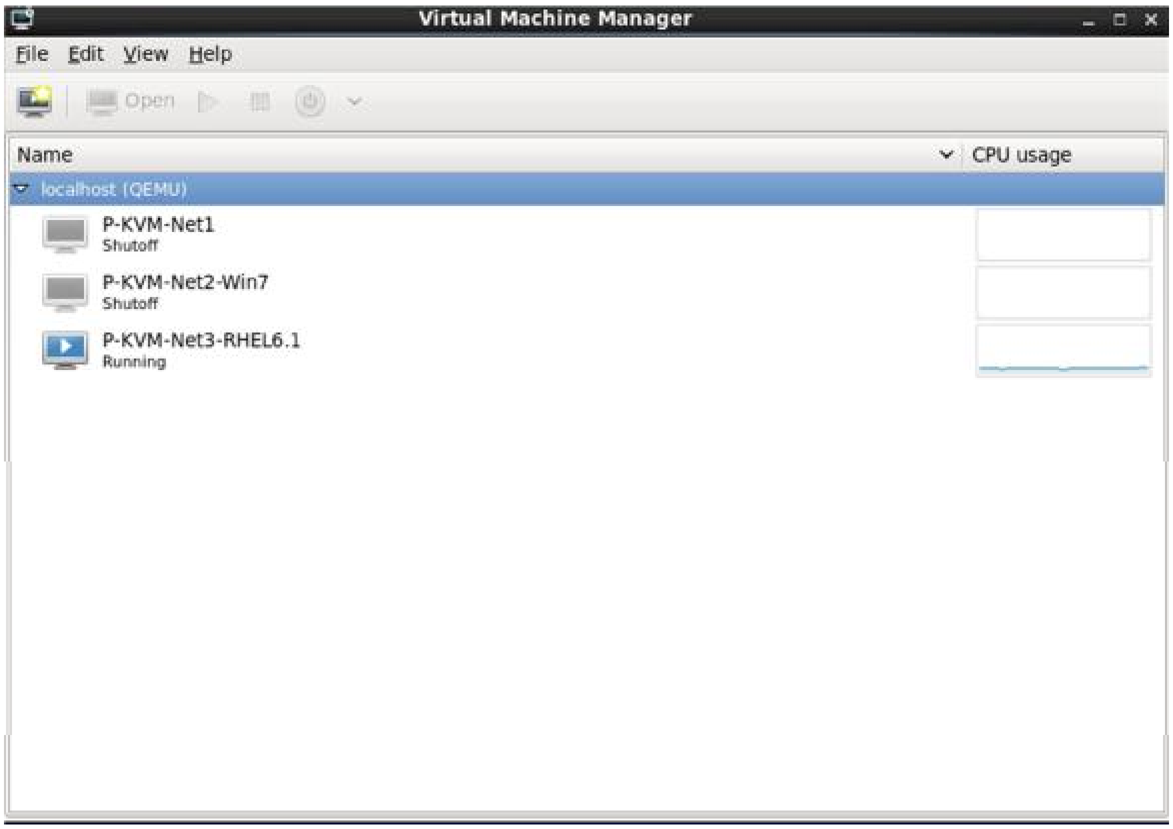
This procedure creates a new storage pool using a partition of a storage device.

Creating a partition-based storage pool with virt-manager

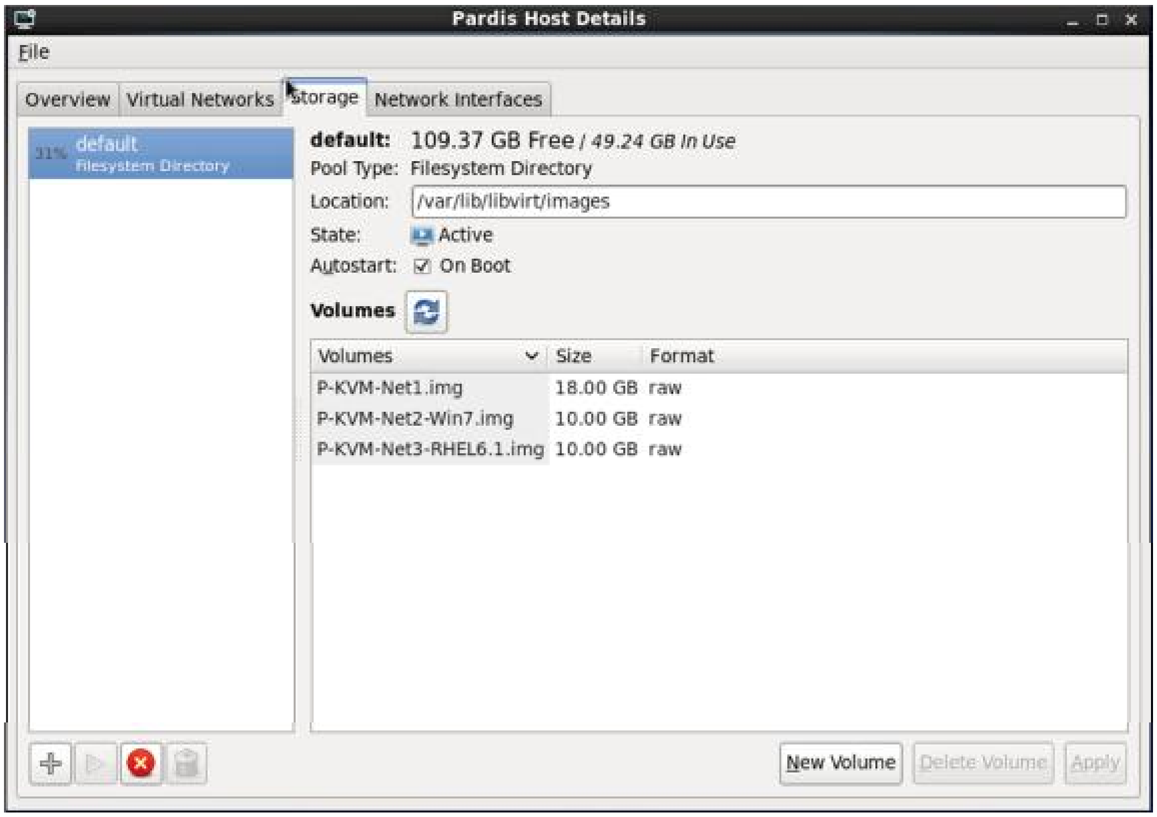
1. Open the storage pool settings

a. In the virt-manager graphical interface, select the host from the main window.

Open the Edit menu and select Host Details



b. Click on the Storage tab of the Host Details window.



default: 109.37 GB Free / 49.24 GB In Use
Pool Type: Filesystem Directory
Location: /var/lib/libvirt/images
State: Active
Autostart: ☑ On Boot

Volumes	Size	Format
P-KVM-Net1.img	18.00 GB	raw
P-KVM-Net2-Win7.img	10.00 GB	raw
P-KVM-Net3-RHEL6.1.img	10.00 GB	raw

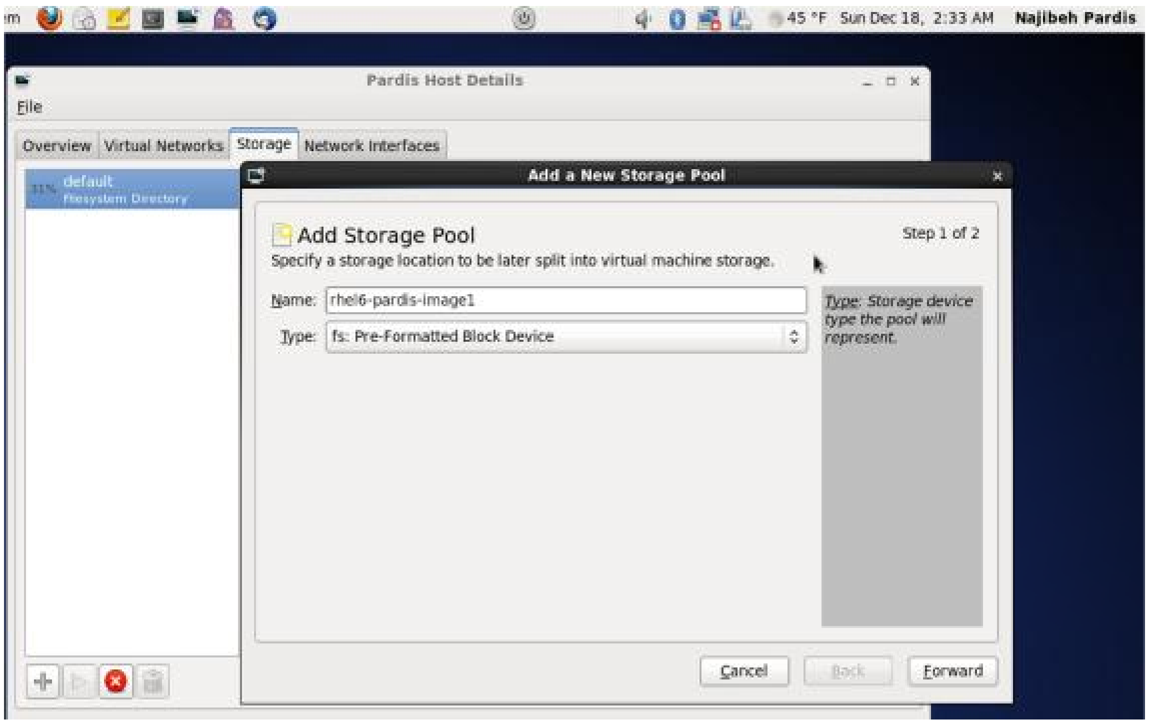
2. Create the new storage pool

a. Add a new pool (part 1)

Press the + button (the add pool button). The Add a New Storage Pool wizard appears.

Choose a Name for the storage pool. This example uses the name guest_images_fs.

Change the Type to fs: Pre-Formatted Block Device.



Add Storage Pool

Step 1 of 2

Specify a storage location to be later split into virtual machine storage.

Name: rhel6-pardis-image1

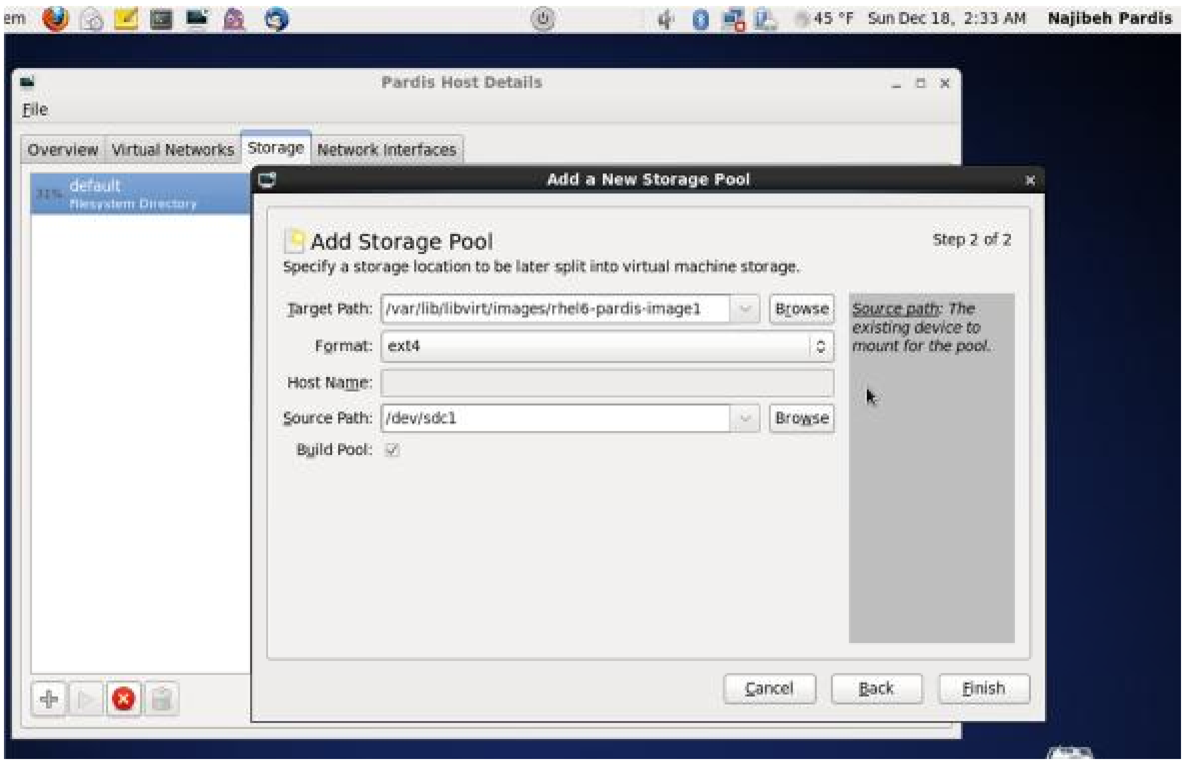
Type: fs: Pre-Formatted Block Device

Type: Storage device type the pool will represent.

Press the Forward button to continue.

b. Add a new pool (part 2)

Change the Target Path, Format, and Source Path fields.



Add Storage Pool

Step 2 of 2

Specify a storage location to be later split into virtual machine storage.

Target Path: /var/lib/libvirt/images/rhel6-pardis-image1

Format: ext4

Host Name:

Source Path: /dev/sdc1

Build Pool: ✓

Source path: The existing device to mount for the pool.

Target Path

Enter the location to mount the source device for the storage pool in the Target Path field. If the location does does not already exist, virt-manager will create the directory.

Format

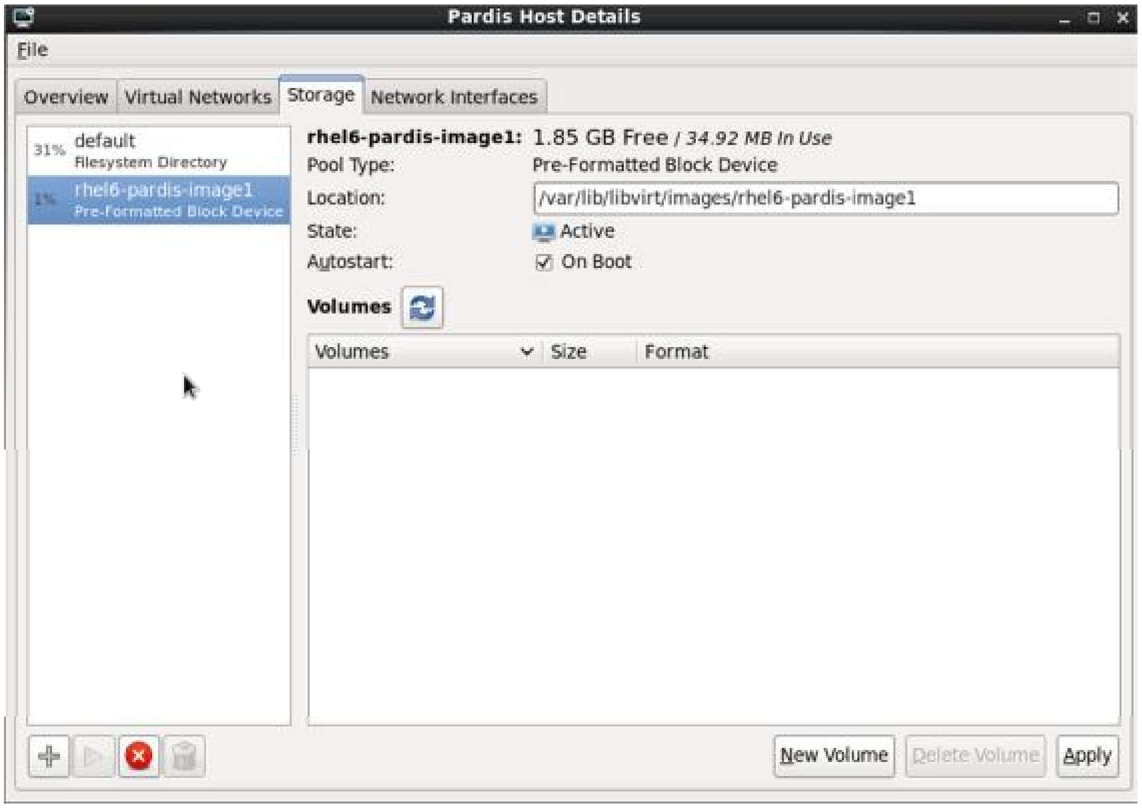
Select a format from the Format list. The device is formatted with the selected format. This example uses the ext4 file system, the default Red Hat Enterprise Linux file system.

Source Path

Enter the device in the Source Path field. This example uses the /dev/sdc1 device. Verify the details and press the Finish button to create the storage pool.

3. Verify the new storage pool

The new storage pool appears in the storage list on the left after a few seconds. Verify the size is reported as expected, 458.20 GB Free in this example. Verify the State field reports the new storage pool as Active. Select the storage pool. In the Autostart field, click the On Boot checkbox. This will make sure the storage device starts whenever the libvirtd service starts.



rhel6-pardis-image1: 1.85 GB Free / *34.92 MB In Use*
Pool Type: Pre-Formatted Block Device
Location: /var/lib/libvirt/images/rhel6-pardis-image1
State: Active
Autostart: ☑ On Boot

The storage pool is now created, close the Host Details window.

Directory-based storage pools

This section covers storing virtualized guests in a directory on the host . Directory-based storage pools can be created with virt-manager or the virsh command line tools.

Creating a directory-based storage pool with virt-manager

1. Create the local directory

a. Optional: Create a new directory for the storage pool

Create the directory on the host for the storage pool. An existing directory can be used if permissions and SELinux are configured correctly. This example uses a directory named /guest_images.

```
# mkdir /guest_images
```

b. Set directory ownership

Change the user and group ownership of the directory. The directory must be owned by the root user.

```
# chown root:root /guest_images
```

c. Set directory permissions

Change the file permissions of the directory.

```
# chmod 700 /guest_images
```

d. Verify the changes

Verify the permissions were modified. The output shows a correctly configured empty directory.

```
# ls -la /guest_images
total 8
```

```
drwx------.  2 root root 4096 May 28 13:57 .
dr-xr-xr-x. 26 root root 4096 May 28 13:57 ..
```

2. Configure SELinux file contexts

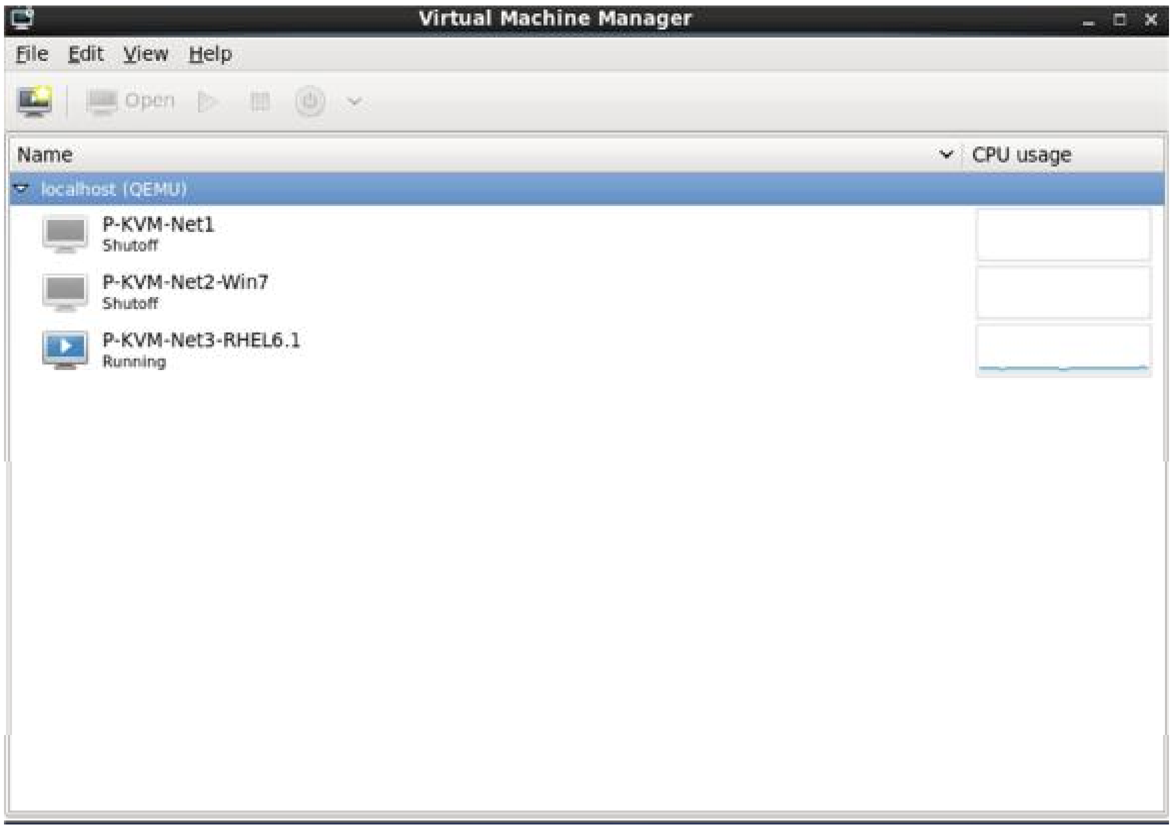
Configure the correct SELinux context for the new directory.

```
# semanage fcontext -a -t virt_image_t /guest_images
```

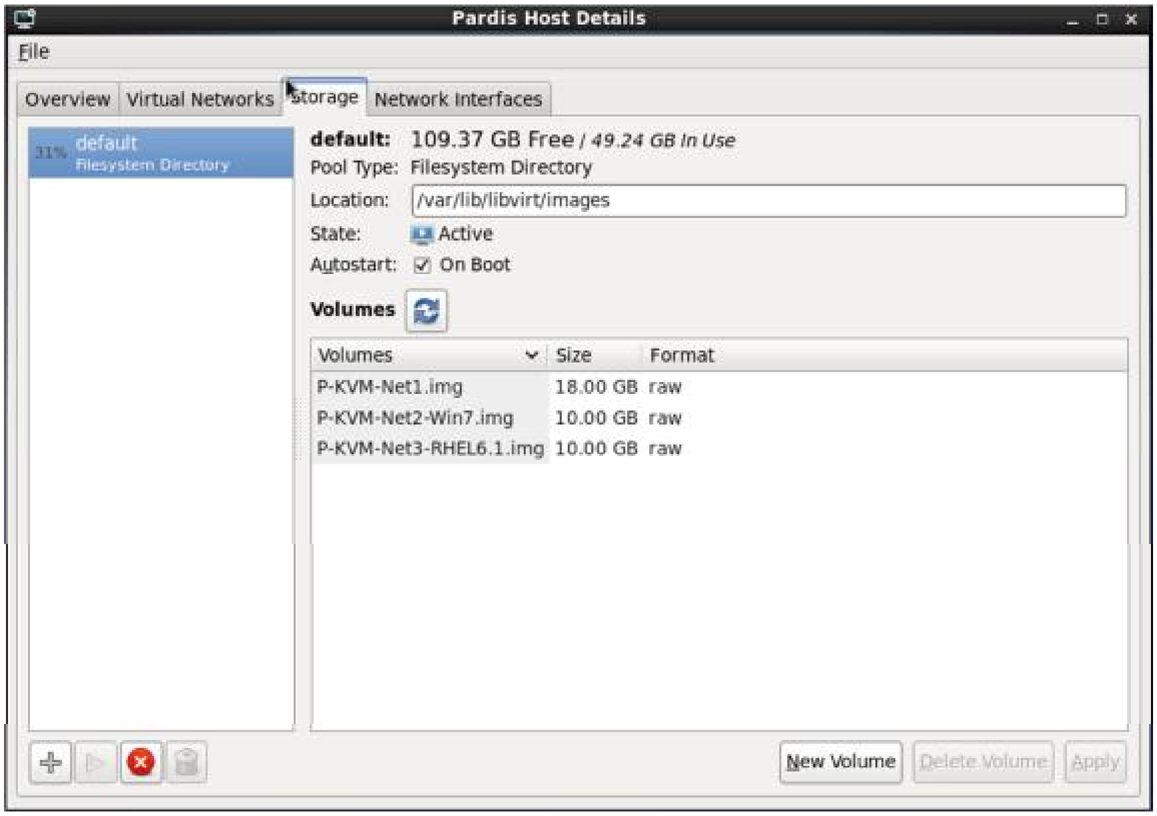
3. Open the storage pool settings

a. In the virt-manager graphical interface, select the host from the main window.

Open the Edit menu and select Host Details



b. Click on the Storage tab of the Host Details window.



default: 109.37 GB Free / *49.24 GB In Use*

Pool Type: Filesystem Directory

Location: /var/lib/libvirt/images

State: Active

Autostart: ✓ On Boot

Volumes	Size	Format
P-KVM-Net1.img	18.00 GB	raw
P-KVM-Net2-Win7.img	10.00 GB	raw
P-KVM-Net3-RHEL6.1.img	10.00 GB	raw

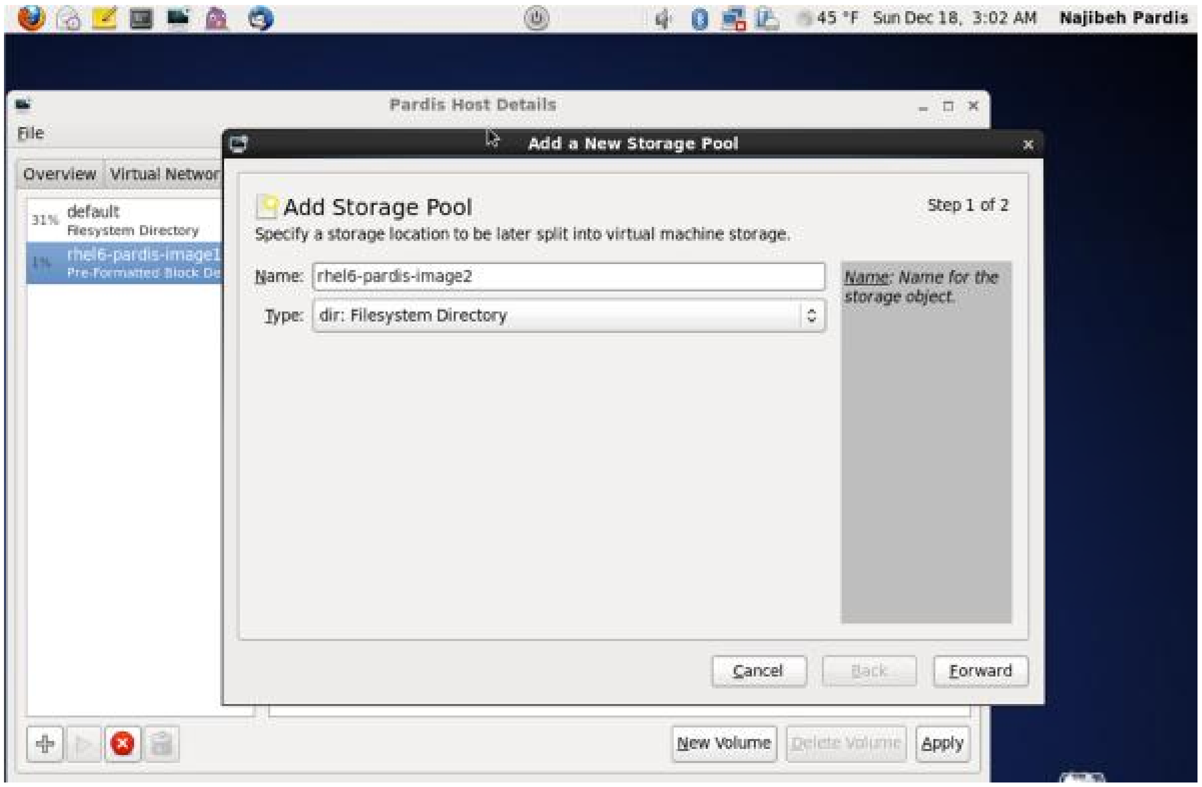
4. Create the new storage pool

a. Add a new pool (part 1)

Press the + button (the add pool button). The Add a New Storage Pool wizard appears.

Choose a Name for the storage pool. This example uses the name guest_images_dir.

Change the Type to dir: Filesystem Directory.


45 °F Sun Dec 18, 3:02 AM Najibeh Pardis
Pardis Host Details
File
Overview
Virtual Networ
31% default Filesystem Directory
1% rhel6-pardis-image1 Pre-Formatted Block De
Add a New Storage Pool
Cancel
Back
Forward
New Volume
Delete Volume
Apply

Add Storage Pool

Step 1 of 2

Specify a storage location to be later split into virtual machine storage.

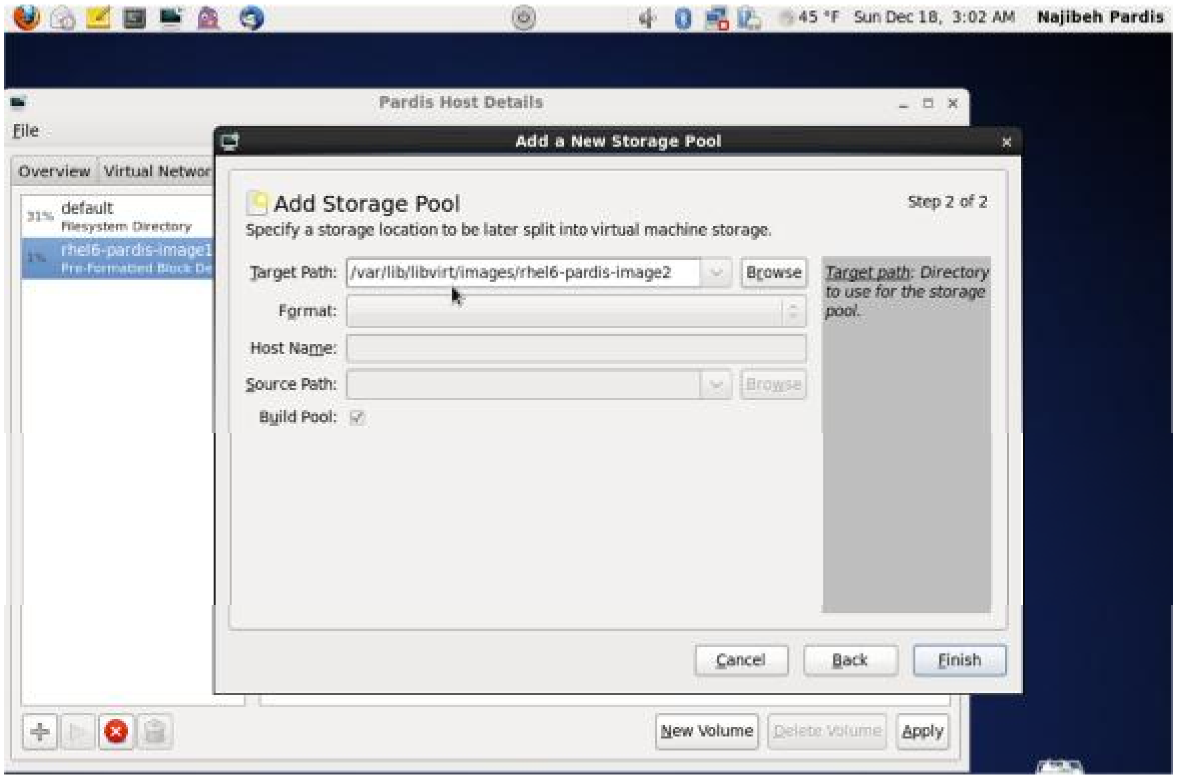
Name: rhel6-pardis-image2

Type: dir: Filesystem Directory

Name: Name for the storage object.

Press the Forward button to continue.

b. Add a new pool (part 2) and Change the Target Path field. This example uses /guest_images.


45 °F Sun Dec 18, 3:02 AM Najibeh Pardis
Pardis Host Details
File
Overview
Virtual Networ
31% default Filesystem Directory
1% rhel6-pardis-image1 Pre-Formatted Block De
Add a New Storage Pool
Browse
Cancel
Back
Finish
New Volume
Delete Volume
Apply

Add Storage Pool

Step 2 of 2

Specify a storage location to be later split into virtual machine storage.

Target Path: /var/lib/libvirt/images/rhel6-pardis-image2

Format:

Host Name:

Source Path:

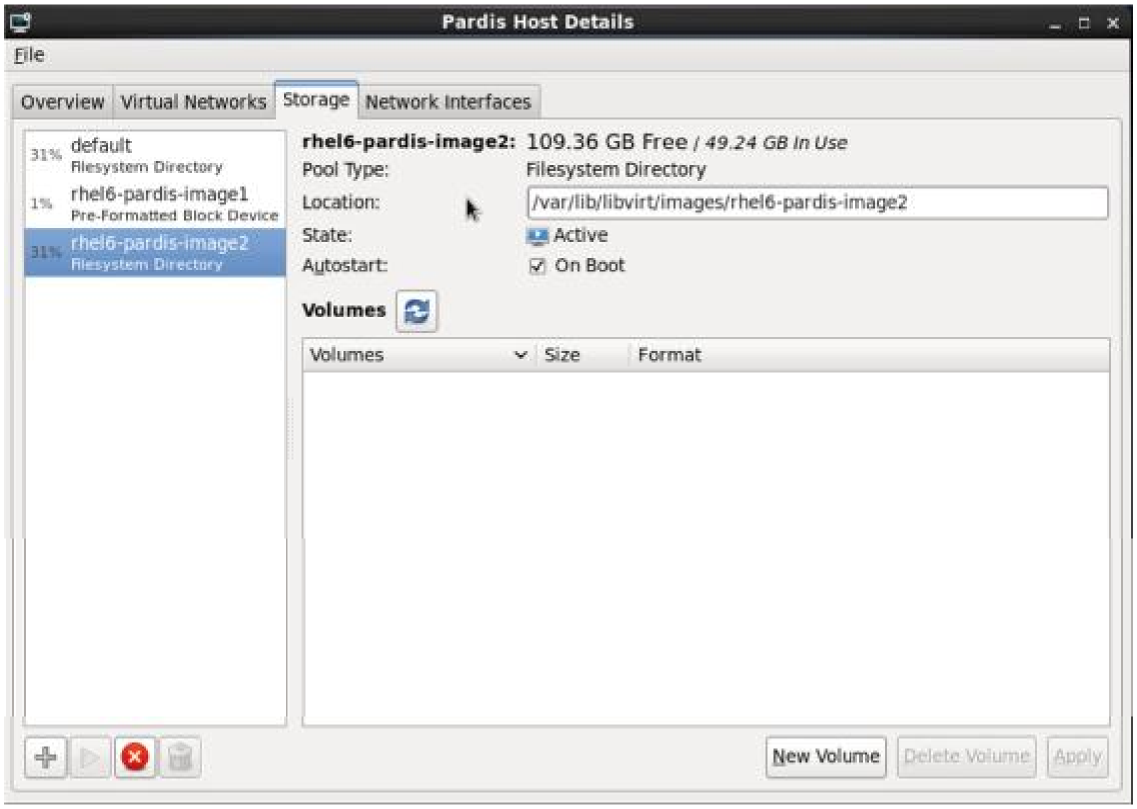
Build Pool: ✓

Target path: Directory to use for the storage pool.

Verify the details and press the Finish button to create the storage pool.

5. Verify the new storage pool

The new storage pool appears in the storage list on the left after a few seconds. Verify the size is reported as expected, 36.41 GB Free in this example. Verify the State field reports the new storage pool as Active. Select the storage pool. In the Autostart field, click the On Boot checkbox. This will make sure the storage pool starts whenever the libvirtd service sarts.


Pardis Host Details

31% default
Filesystem Directory
1% rhel6-pardis-image1
Pre-Formatted Block Device
31% rhel6-pardis-image2
Filesystem Directory

rhel6-pardis-image2: 109.36 GB Free / 49.24 GB In Use
Pool Type: Filesystem Directory
Location: /var/lib/libvirt/images/rhel6-pardis-image2
State: Active
Autostart: ☑ On Boot

Volumes

Volumes	Size	Format

The storage pool is now created, close the Host Details window.

LVM-based storage pools

This chapter covers using LVM volume groups as storage pools. LVM-based storage groups provide the full flexibility of LVM.

Creating an LVM-based storage pool with virt-manager

LVM-based storage pools can use existing LVM volume groups or create new LVM volume groups on a blank partition.

1. Optional: Create new partition for LVM volumes

These steps describe how to create a new partition and LVM volume group on a new hard disk drive.

a. Create a new partition

Use the fdisk command to create a new disk partition from the command line. The following example creates a new partition that uses the entire disk on the storage device /dev/sdb.

```
# fdisk /dev/sdb

Command (m for help):
```

Press n for a new partition.

b. Press p for a primary partition.

```
Command action

   e   extended

   p   primary partition (1-4)
```

c. Choose an available partition number. In this example the first partition is chosen by entering

```
Partition number (1-4): 1
```

d. Enter the default first cylinder by pressing Enter.

```
First cylinder (1-400, default 1):
```

e. Select the size of the partition. In this example the entire disk is allocated by pressing Enter.

```
Last cylinder or +size or +sizeM or +sizeK (2-400, default 400):
```

f. Set the type of partition by pressing t.

```
Command (m for help): t
```

g. Choose the partition you created in the previous steps. In this example, the partition number is 1.

```
Partition number (1-4): 1
```

h. Enter 8e for a Linux LVM partition.

```
Hex code (type L to list codes): 8e
```

i. write changes to disk and quit.

```
Command (m for help): w

Command (m for help): q
```

j. Create a new LVM volume group

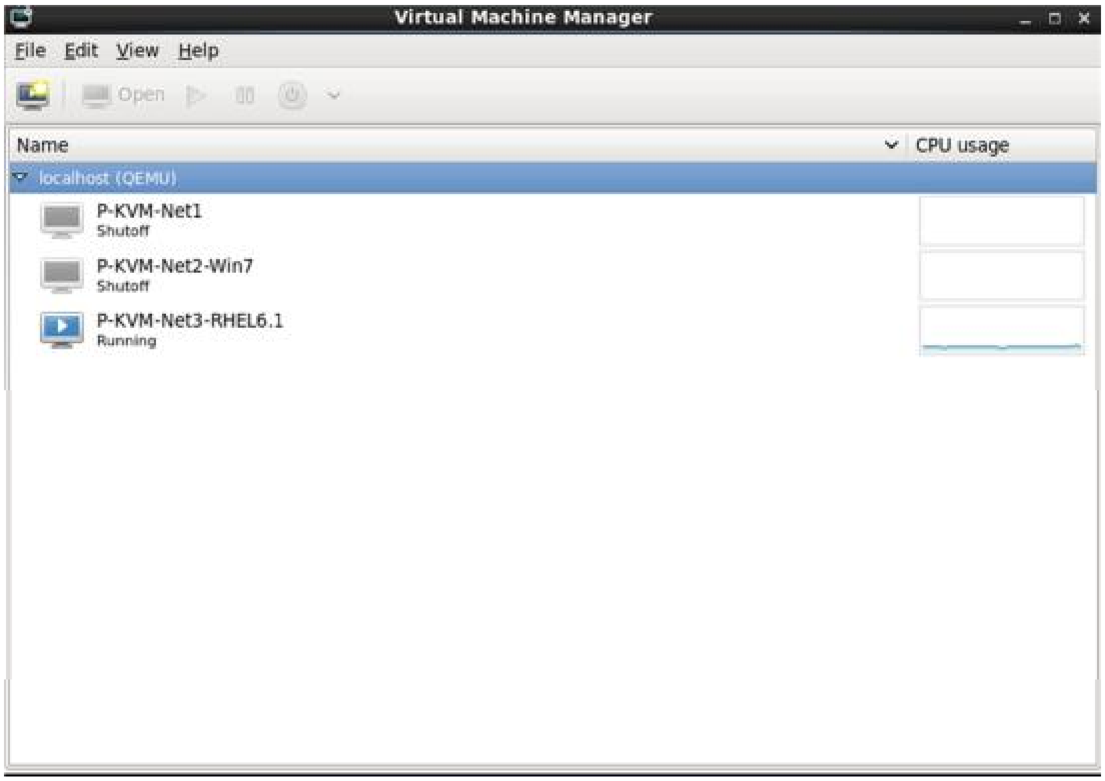
Create a new LVM volume group with the vgcreate command. This example creates a volume group named guest_images_lvm.

```
# vgcreate guest_images_lvm /dev/sdb1

Physical volmue "/dev/vdb1" successfully created

Volume group "guest_images_lvm" successfully created
```

The new LVM volume group, guest_images_lvm, can now be used for an LVM-based storage pool.

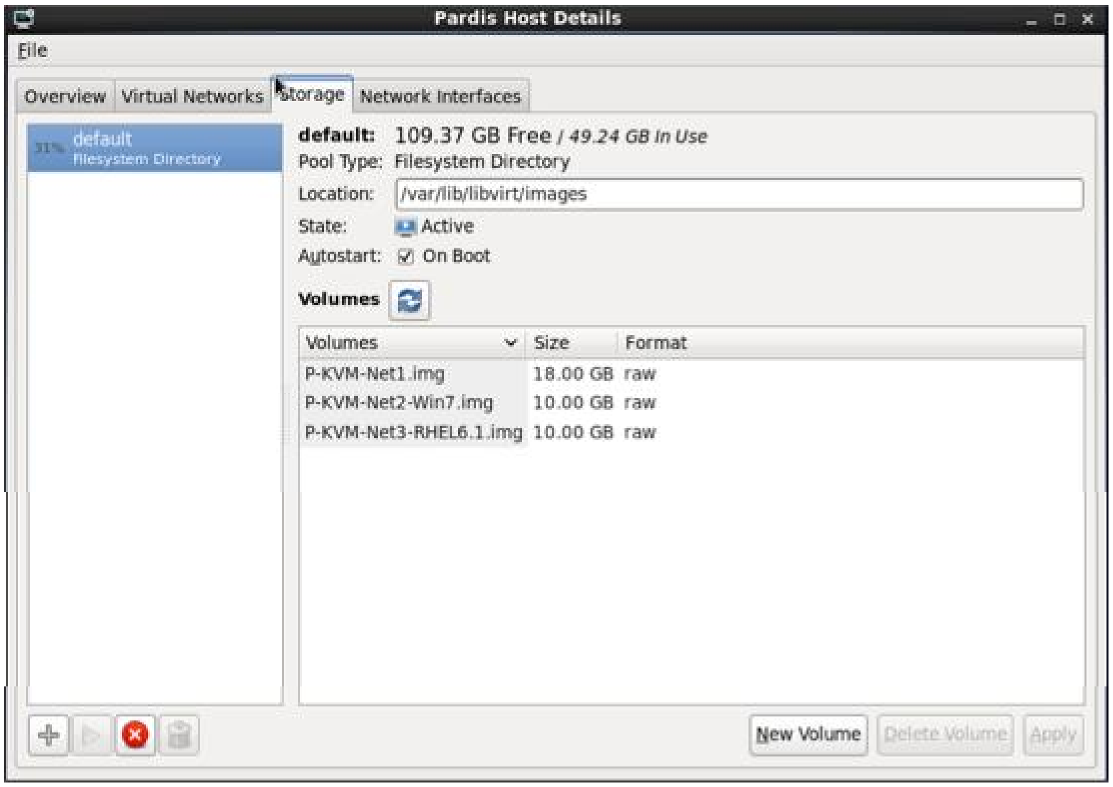
2. Open the storage pool settings

a. In the virt-manager graphical interface, select the host from the main window. Open the Edit menu and select Host Details



localhost (QEMU)
P-KVM-Net1 Shutoff
P-KVM-Net2-Win7 Shutoff
P-KVM-Net3-RHEL6.1 Running

b. Click on the Storage tab of the Host Details window.



default Filesystem Directory

default: 109.37 GB Free / *49.24 GB In Use*
Pool Type: Filesystem Directory
Location: /var/lib/libvirt/images
State: Active
Autostart: ☑ On Boot

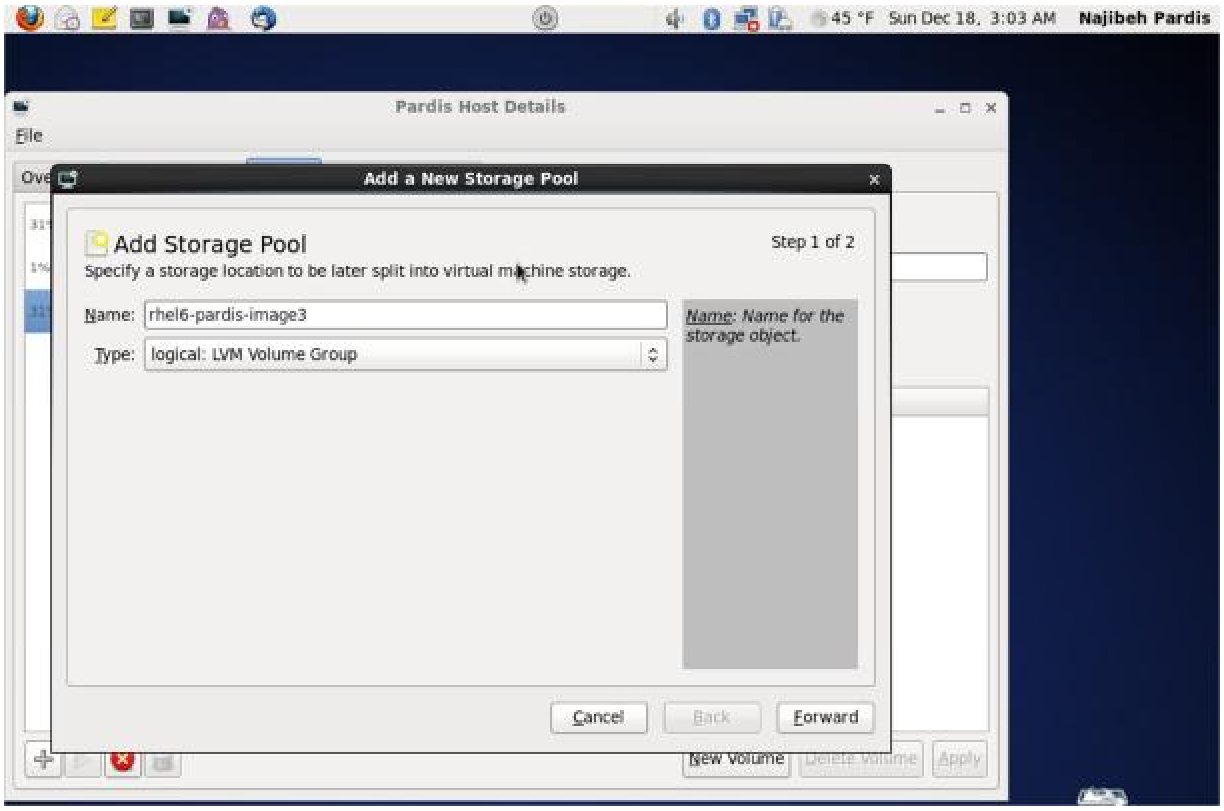
Volumes

Volumes	Size	Format
P-KVM-Net1.img	18.00 GB	raw
P-KVM-Net2-Win7.img	10.00 GB	raw
P-KVM-Net3-RHEL6.1.img	10.00 GB	raw

3. Create the new storage pool

a. Start the Wizard

Press the + button (the add pool button). The Add a New Storage Pool wizard appears. Choose a Name for the storage pool. We use guest_images_lvm for this example. Then change the Type to logical: LVM Volume Group, and



Add Storage Pool

Step 1 of 2

Specify a storage location to be later split into virtual machine storage.

Name: rhel6-pardis-image3

Type: logical: LVM Volume Group

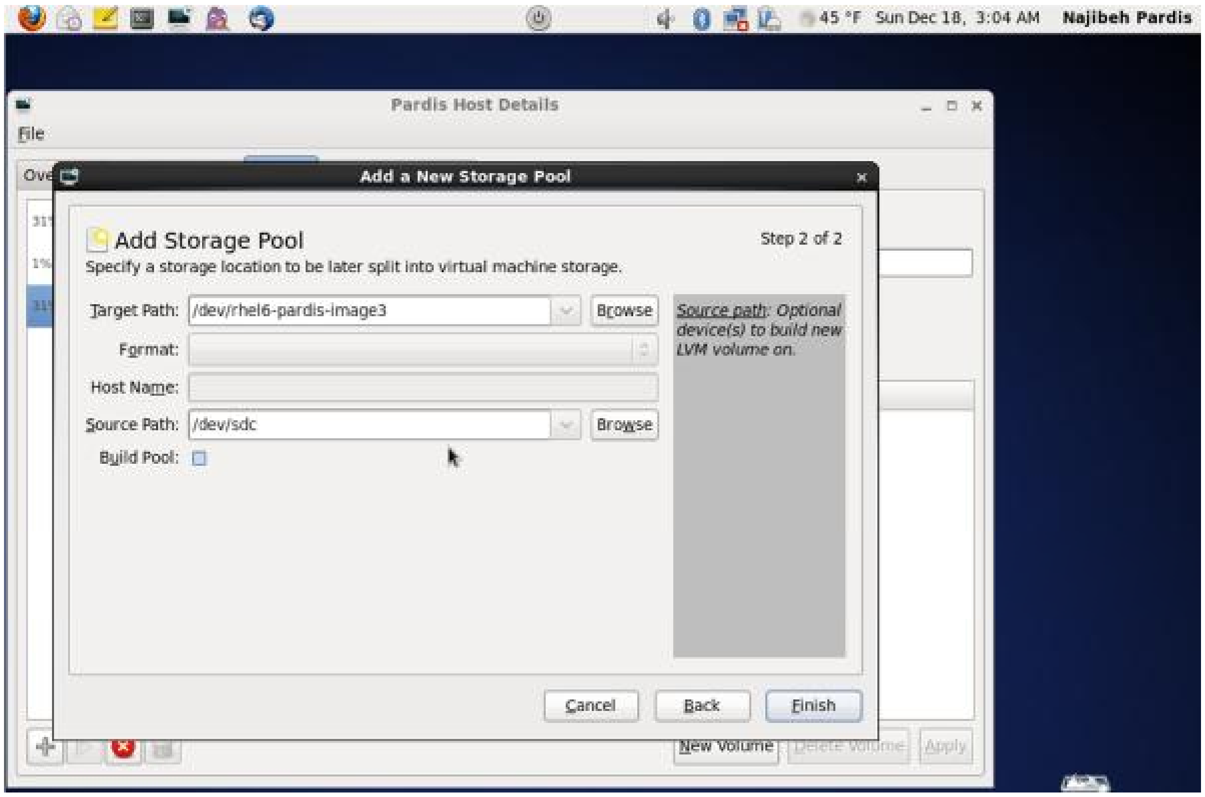
Name: Name for the storage object.

Press the Forward button to continue.

b. Add a new pool (part 2)

Change the Target Path field. This example uses /guest_images. Now fill in the Target Path and Source Path fields, then tick the Build Pool check box.

- Use the Target Path field to either select an existing LVM volume group or as the name for a new volume group. The default format is /dev/storage_pool_name. This example uses a new volume group named /dev/guest_images_lvm.
- The Source Path field is optional if an existing LVM volume group is used in the Target Path. For new LVM volume groups, input the location of a storage device in the Source Path field. This example uses a blank partition /dev/sdc.
- The Build Pool checkbox instructs virt-manager to create a new LVM volume group. If you are using an existing volume group you should not select the Build Pool checkbox. This example is using a blank partition to create a new volume group so the Build Pool checkbox must be selected.



Add Storage Pool

Step 2 of 2

Specify a storage location to be later split into virtual machine storage.

Target Path: /dev/rhel6-pardis-image3 Browse

Format:

Host Name:

Source Path: /dev/sdc Browse

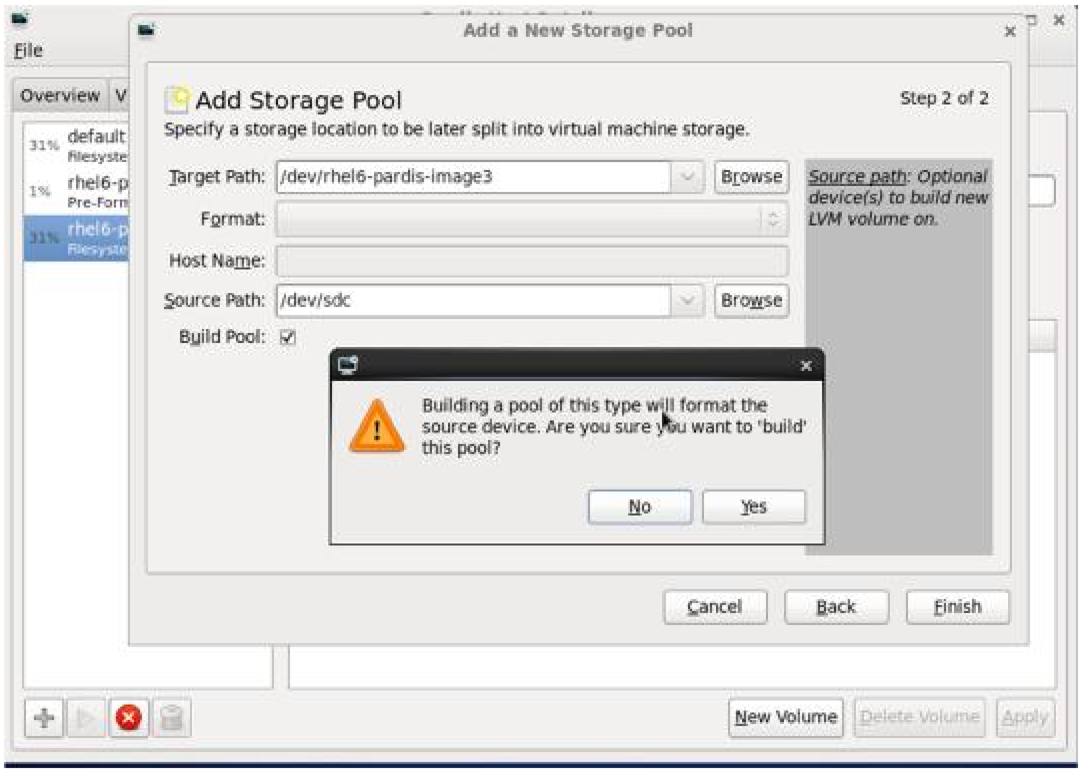
Build Pool: ☐

Source path: Optional device(s) to build new LVM volume on.

Verify the details and press the Finish button format the LVM volume group and create the storage pool.

c. Confirm the device to be formatted

A warning message appears.



Specify a storage location to be later split into virtual machine storage.

Target Path: /dev/rhel6-pardis-image3

Format:

Host Name:

Source Path: /dev/sdc

Build Pool: ✓

Source path: Optional device(s) to build new LVM volume on.

Building a pool of this type will format the source device. Are you sure you want to 'build' this pool?

Press the Yes button to proceed to erase all data on the storage device and create the storage pool.

4. Verify the new storage pool

The new storage pool will appear in the list on the left after a few seconds. Verify the details are what you expect, 465.76 GB Free in our example. Also verify the State field reports the new storage pool as Active. It is generally a good idea to have the Auto start check box enabled, to ensure the storage pool starts automatically with libvirtd.

Close the Host Details dialog, as the task is now complete.

ISCSI-based storage pools

This section covers using iSCSI-based devices to store virtualized guests. iSCSI (Internet Small Computer System Interface) is a network protocol for sharing storage devices. iSCSI connects initiators (storage clients) to targets (storage servers) using SCSI instructions over the IP layer. For more information and background on the iSCSI protocol refer to wikipedia's iSCSI article1.

Configuring a software iSCSI target

The scsi-target-utils package provides a tool for creating software-backed iSCSI targets.

1. Install the required packages

Install the scsi-target-utils package and all dependencies

```
# yum install scsi-target-utils
```

2. Start the tgtd service

The tgtd service hosts SCSI targets and uses the iSCSI protocol to host targets. Start the tgtd service and make the service persistent after restarting with the chkconfig command.

```
# service tgtd start

# chkconfig tgtd on
```

3. Optional: Create LVM volumes

LVM volumes are useful for iSCSI backing images. LVM snapshots and resizing can be beneficial for virtualized guests. This example creates an LVM image named virtimage1 on a new volume group named virtstore on a RAID5 array for hosting virtualized guests with iSCSI.

a. Create the RAID array

Creating software RAID5 arrays is covered by the Red Hat Enterprise Linux Deployment Guide.

b. Create the LVM volume group

Create a volume group named virtstore with the vgcreate command.

```
# vgcreate virtstore /dev/md1
```

c. Create a LVM logical volume

Create a logical volume group named virtimage1 on the virtstore volume group with a size of 20GB using the lvcreate command.

```
# lvcreate --size 20G -n virtimage1

virtstore
```

The new logical volume, virtimage1, is ready to use for iSCSI.

4. Optional: Create file-based images

File-based storage is sufficient for testing but is not recommended for production environments or any significant I/O activity. This optional procedure creates a file based imaged named virtimage2.img for an iSCSI target.

a. Create a new directory for the image

Create a new directory to store the image. The directory must have the correct SELinux contexts.

```
# mkdir -p /var/lib/tgtd/virtualization
```

b. Create the image file

Create an image named virtimage2.img with a size of 10GB.

```
# dd if=/dev/zero of=/var/lib/tgtd/virtualization/virtimage2.img bs=1M seek=10000  count=0
```

c. Configure SELinux file contexts

Configure the correct SELinux context for the new image and directory.

```
# restorecon -R /var/lib/tgtd
```

The new file-based image, virtimage2.img, is ready to use for iSCSI.

5. Create targets

Targets can be created by adding a XML entry to the /etc/tgt/targets.conf file. The target attribute requires an iSCSI Qualified Name (IQN). The IQN is in the format:

```
iqn.yyyy-mm.reversed domain name:optional identifier text
```

6. Restart the tgtd service

Restart the tgtd service to reload the configuration changes.

```
# service tgtd restart
```

7. iptables configuration

Open port 3260 for iSCSI access with iptables.

```
# iptables -I INPUT -p tcp -m tcp --dport 3260 -j ACCEPT

# service iptables save

# service iptables restart
```

8. Verify the new targets

View the new targets to ensure the setup was success with the tgt-admin --show command.

```
# tgt-admin –show
```

9. Optional: Test discovery

Test whether the new iSCSI device is discoverable.

```
# iscsiadm --mode discovery --type sendtargets --portal server1.example.com 127.0.0.1:3260,1
iqn.2010-05.com.example.server1:trial1
```

10. Optional: Test attaching the device

Attach the new device to determine whether the device can be attached.

```
# iscsiadm -d2 -m node --login
```

```
scsiadm: Max file limits 1024 1024
```

```
Logging in to [iface: default, target: iqn.2010-05.com.example.server1:trial1, portal: 10.0.0.1,3260]
Login to [iface: default, target: iqn.2010-05.com.example.server1:trial1, portal: 10.0.0.1,3260] successful.
```

Detach the device.

```
# iscsiadm -d2 -m node –logout
```

An iSCSI device is now ready to use for virtualization.

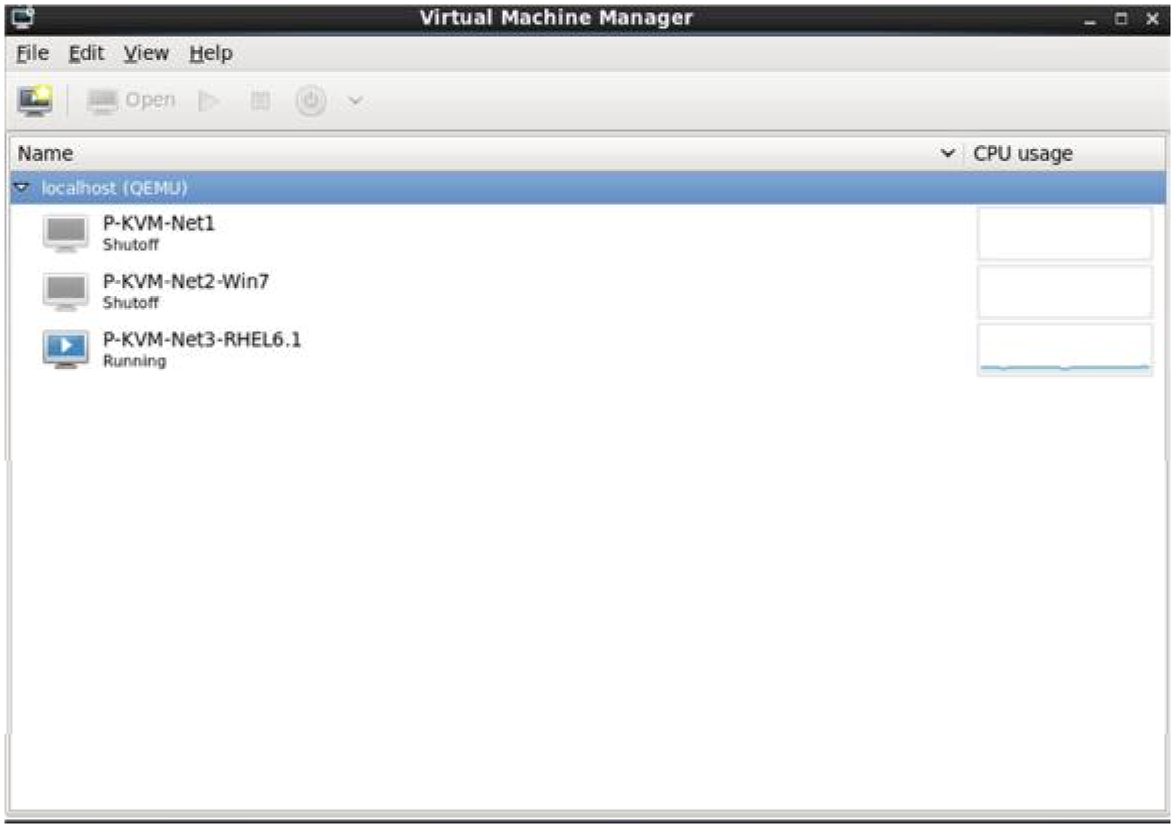
Adding an iSCSI target to virt-manager

1. Open the host storage tab

Open the Storage tab in the Host Details window.

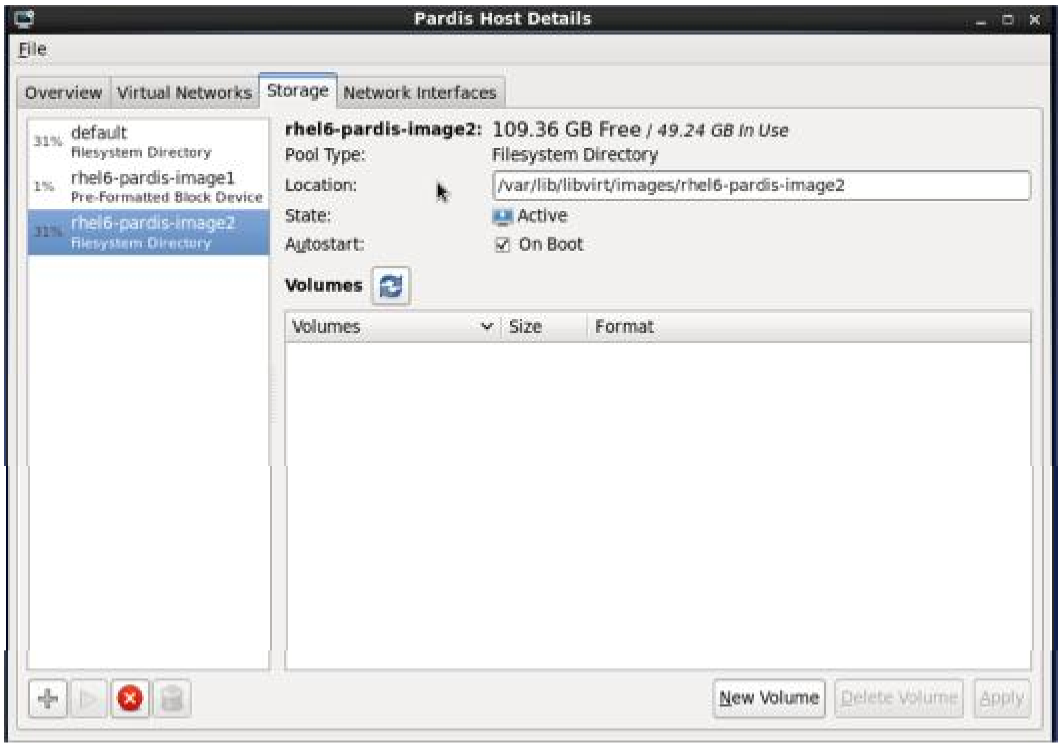
a. Open virt-manager.

b. Select a host from the main virt-manager window.



c. Open the Edit menu and select Host Details.

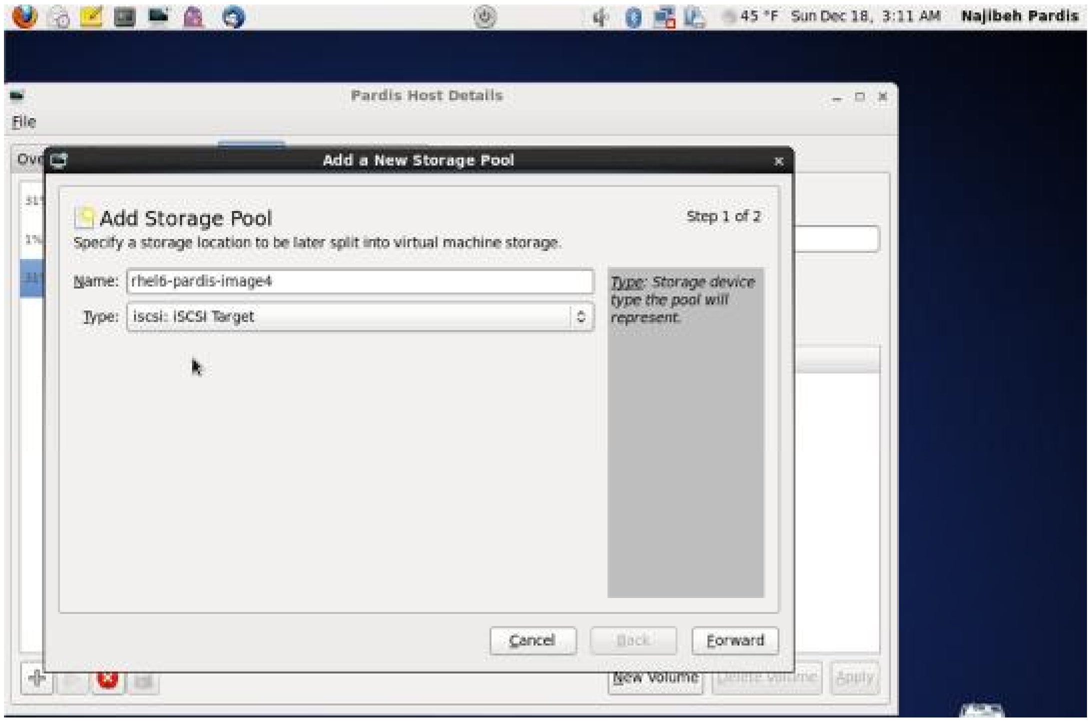
d. Click on the Storage tab of the Host Details window.



rhel6-pardis-image2: 109.36 GB Free / *49.24 GB In Use*
Pool Type: Filesystem Directory
Location: /var/lib/libvirt/images/rhel6-pardis-image2
State: Active
Autostart: ☑ On Boot
Volumes

2. Add a new pool (part 1)

Press the + button (the add pool button). The Add a New Storage Pool wizard appears.



Add Storage Pool
Specify a storage location to be later split into virtual machine storage.
Name: rhel6-pardis-image4
Type: iscsi: iSCSI Target
Type: Storage device type the pool will represent.

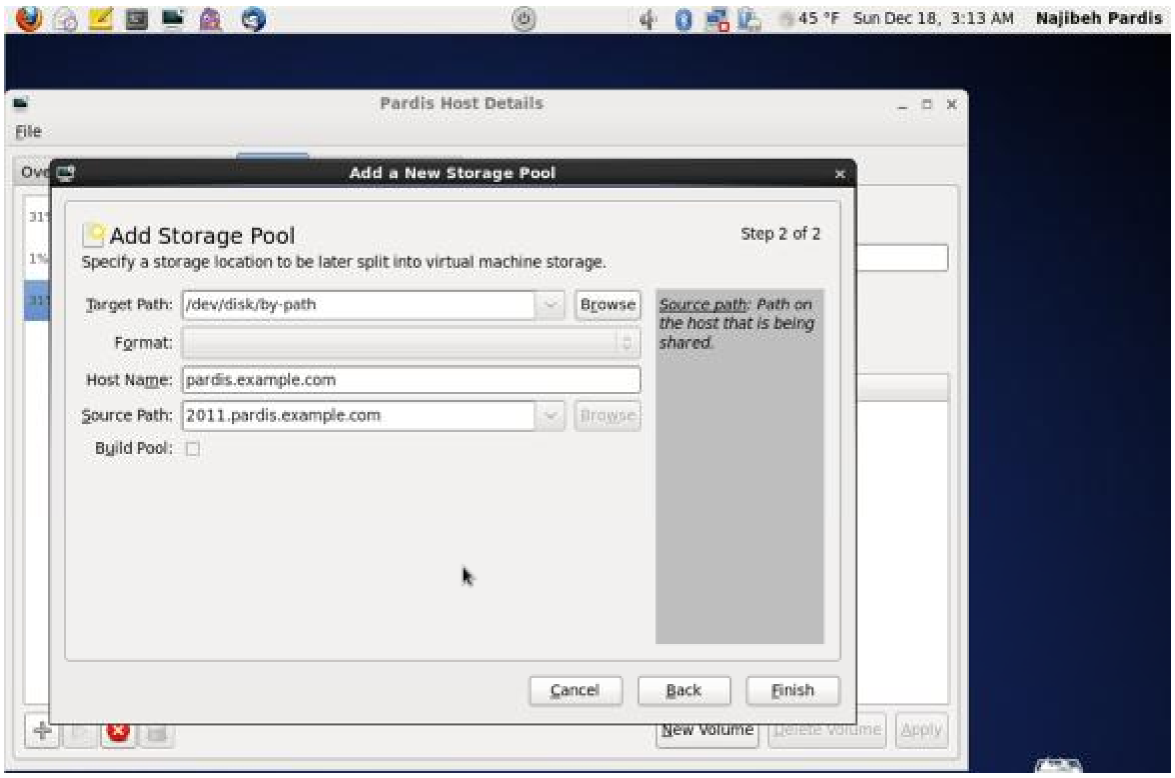
Choose a name for the storage pool, change the Type to iscsi, and press Forward to continue.

3. Add a new pool (part 2)

Enter the target path for the device, the host name of the target and the source path (the IQN). The Format option is not available as formatting is handled by the guests. It is not advised to edit the Target Path. The default target path value, /dev/disk/by-path/, adds the drive path to that directory. The target path should be the same on all hosts for migration.

Enter the hostname or IP address of the iSCSI target. This example uses server1.example.com.

Enter the source path, the IQN for the iSCSI target. This example uses iqn.2010-05.com.example.server1:trial1.



Add a New Storage Pool

Add Storage Pool

Step 2 of 2

Specify a storage location to be later split into virtual machine storage.

Target Path: /dev/disk/by-path

Format:

Host Name: pardis.example.com

Source Path: 2011.pardis.example.com

Build Pool: ☐

Source path: Path on the host that is being shared.

Press Finish to create the new storage pool.

NFS-based storage pools

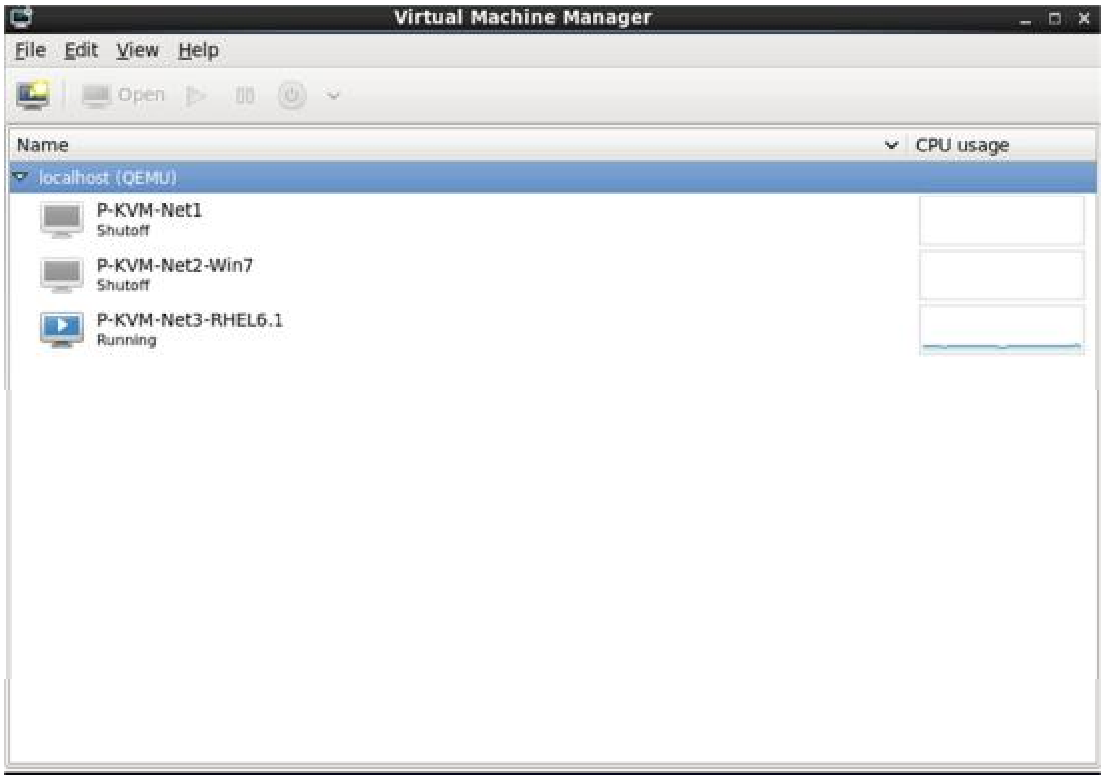
Creating a NFS-based storage pool with virt-manager

1. Open the host storage tab

Open the Storage tab in the Host Details window.

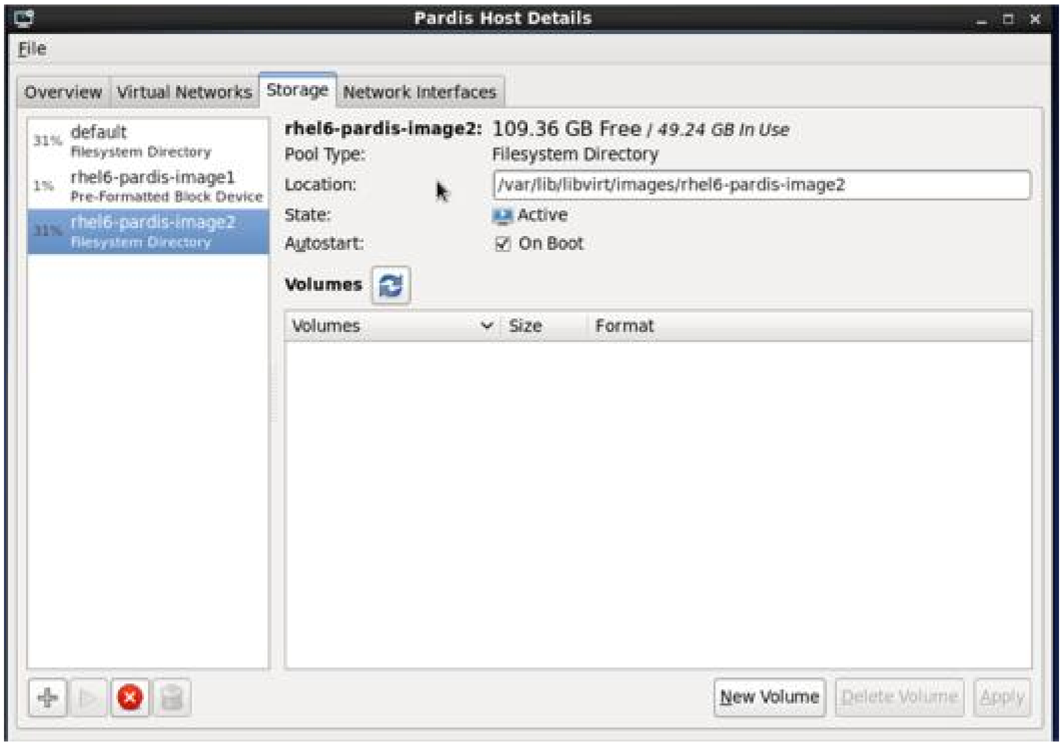
a. Open virt-manager.

b. Select a host from the main virt-manager window.



c. Open the Edit menu and select Host Details.

d. Click on the Storage tab of the Host Details window.



31%	default Filesystem Directory
1%	rhel6-pardis-image1 Pre-Formatted Block Device
31%	rhel6-pardis-image2 Filesystem Directory

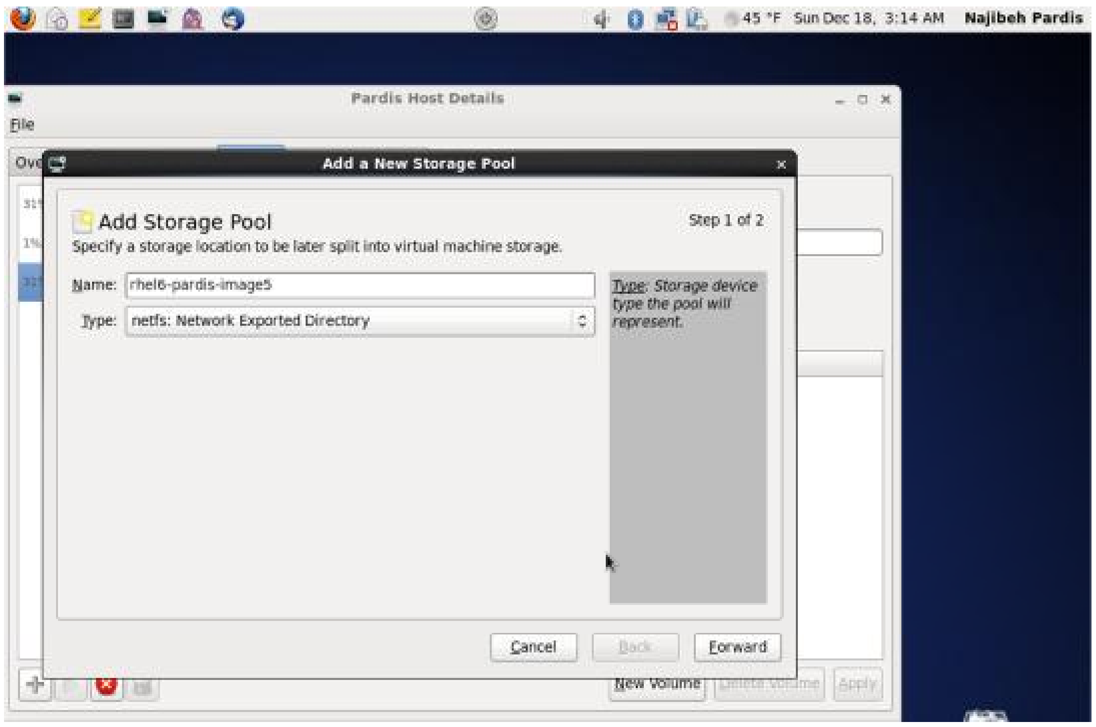
rhel6-pardis-image2: 109.36 GB Free / *49.24 GB In Use*
Pool Type: Filesystem Directory
Location: /var/lib/libvirt/images/rhel6-pardis-image2
State: Active
Autostart: ☑ On Boot

Volumes

Volumes	Size	Format

2. Create a new pool (part 1)

Press the + button (the add pool button). The Add a New Storage Pool wizard appears.



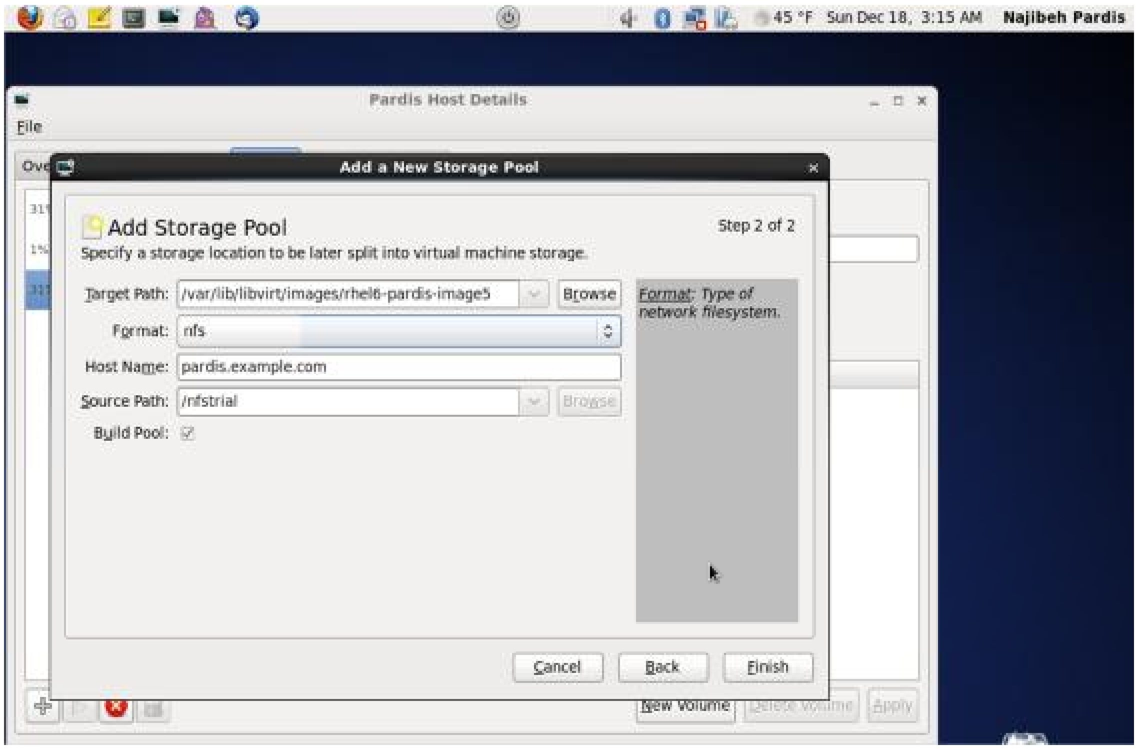
Add Storage Pool
Step 1 of 2
Specify a storage location to be later split into virtual machine storage.
Name: rhel6-pardis-image5
Type: netfs: Network Exported Directory
Type: Storage device type the pool will represent.

Choose a name for the storage pool and press Forward to continue.

3. Create a new pool (part 2)

Enter the target path for the device, the hostname and the NFS share path. Set the Format option to NFS or auto (to detect the type). The target path must be identical on all hosts for migration. Enter the hostname or IP address of the NFS server. This example uses server1.example.com.

Enter the NFS path. This example uses /nfstrial.



Specify a storage location to be later split into virtual machine storage.

Target Path: /var/lib/libvirt/images/rhel6-pardis-image5
Format: nfs
Host Name: pardis.example.com
Source Path: /nfstrial
Build Pool: ☑

Format: Type of network filesystem.

Press Finish to create the new storage pool.

Troubleshooting

This chapter covers common problems and solutions for CentOS Linux 6 virtualization issues . Read this chapter to develop an understanding of some of the common problems associated with virtualization technologies. Troubleshooting takes practice and experience which are difficult to learn from a book.

Debugging and troubleshooting tools

This section summarizes the System Administrator applications, the networking utilities, and debugging tools. You can employ these standard System administration tools and logs to assist with troubleshooting:

- kvm_stat
- trace-cmd
- vmstat
- iostat
- lsof
- systemtap
- crash
- sysrq
- sysrq t
- sysrq w

These networking tools can assist with troubleshooting virtualization networking problems:

- ifconfig
- tcpdump

The tcpdump command 'sniffs' network packets. tcpdump is useful for finding network abnormalities and problems with network authentication. There is a graphical version of tcpdump named wireshark.

- brctl

brctl is a networking tool that inspects and configures the Ethernet bridge configuration in the Virtualization linux kernel. You must have root access before performing these commands .

- Strace : is a command which traces system calls and events received and used by another process.
- Vncviewer : connect to a VNC server running on your server or a virtual machine. Install vncviwer using the yum install vnc command.
- Vncserver : start a remote desktop on your server. Gives you the ability to run graphical user interfaces such as virt-manager via a remote session. Install vncserver using the yum install vnc-server command.

1. kvm_stat

The kvm_stat command is a python script which retrieves runtime statistics from the kvm kernel module. The kvm_stat command can be used to diagnose guest behavior visible to kvm. In particular, performance related issues with guests. Currently, the reported statistics are for the entire system; the behavior of all running guests is reported.
The kvm_stat command requires that the kvm kernel module is loaded and debugfs is mounted. If either of these features are not enabled, the command will output the required steps to enable debugfs or the kvm module.

2. Log files

KVM uses various log files. All the log files are standard ASCII files, and accessible with a text editor.
The default directory for all file-based images is the /var/lib/libvirt/images directory.
qemu-kvm.[PID].log is the log file created by the qemu-kvm process for each fully virtualized guest. When using this log file, you must retrieve the given qemu-kvm process PID, by using the ps command to examine process arguments to isolate the qemu-kvm process on the virtual machine. Note that you must replace the [PID] symbol with the actual PID qemu-kvm process.

If you encounter any errors with the Virtual Machine Manager, you can review the generated data in the virt-manager.log file that resides in the /.virt-manager directory. Note that every time you start the Virtual Machine Manager, it overwrites the existing log file contents. Make sure to backup the virt-manager . log file , before you restart the Virtual Machine manager after a system error.

3. Troubleshooting with serial consoles

Linux kernels can output information to serial ports. This is useful for debugging kernel panics and hardware issues with video devices or headless servers. The subsections in this section cover setting up serial console output for machines running CentOS Linux 6 virtualization kernels and their virtualized guests.

This section covers how to enable serial console output for fully virtualized guests. Fully virtualized guest serial console output can be viewed with the virsh console command .

Be aware fully virtualized guest serial consoles have some limitations. Present limitations include:

output data may be dropped or scrambled.

The serial port is called ttyS0 on Linux or COM1 on Windows.

You must configure the virtualized operating system to output information to the virtual serial port. To output kernel information from a fully virtualized Linux guest into the domain modify the /boot/grub/grub.conf file by inserting the line console=tty0 console=ttyS0,115200.

```
title CentOS Linux Server (2.6.32-36.x86-64)
root (hd0,0)
kernel /vmlinuz-2.6.32-36.x86-64 ro root=/dev/volgroup00/logvol00
console=tty0 console=ttyS0,115200
initrd /initrd-2.6.32-36.x86-64.img
```

Reboot the guest.

On the host, access the serial console with the following command:

```
# virsh console
```

You can also use virt-manager to display the virtual text console. In the guest console window, select Serial Console from the View menu.

4. Virtualization log files

/var/log/libvirt/qemu/GuestName.log

If you encounter any errors with the Virtual Machine Manager, you can review the generated data in the virt-manager.log file that resides in the /.virt-manager directory. Note that every time you start the Virtual Machine Manager, it overwrites the existing log file contents. Make sure to backup the virt-manager.log file, before you restart the Virtual Machine manager after a system error.

5. Loop device errors

If file-based guest images are used you may have to increase the number of configured loop devices. The default configuration allows up to eight active loop devices. If more than eight file-based guests or loop devices are needed the number of loop devices configured can be adjusted in /etc/modprobe.conf. Edit /etc/modprobe.conf and add the following line to it:

```
options loop max_loop=64
```

This example uses 64 but you can specify another number to set the maximum loop value. You may also have to implement loop device backed guests on your system. To use a loop device backed guests for a full virtualized system, use the phy: device or file: file commands.

6. KVM networking performance

By default, KVM virtual machines are assigned a virtual Realtek 8139 (rtl8139) NIC (network interface controller) if they are Windows guests or the guest type is not specified. CentOS 6 Linux guests are assigned a virtio NIC by default.

The rtl8139 virtualized NIC works fine in most environments. However, this device can suffer from performance degradation problems on some networks, for example, a 10 Gigabit Ethernet network.

To improve performance switch to the para-virtualized network driver.

Switching to the virtio driver

- Shutdown the guest operating system.
- Edit the guest's configuration file with the virsh command (where GUEST is the guest's name):

  ```
  # virsh edit GUEST
  ```

  The virsh edit command uses the $EDITOR shell variable to determine which editor to use.
- Find the network interface section of the configuration. This section resembles the snippet below:

  ```
  <interface type='network'>
  [output truncated]
  <model type='rtl8139' />
  </interface>
  ```
- Change the type attribute of the model element from 'rtl8139' to 'virtio'. This will change the driver from the rtl8139 driver to the e1000 driver.

  ```
  <interface type='network'>
  [output truncated]
  <model type='virtio' />
  </interface>
  ```

- Save the changes and exit the text editor
- Restart the guest operating system.

Kvm Management tools

There are a several options available to manage kvm virtual machines for example:

Name/URL	Description	UI Type	Last Updated	Notes	License
UCS / UVMM	Univention Virtual Machine Manager is a high-performance management system for KVM and XEN.	Web-based	active	Uses libvirt	Commercial, Free for personal use edition available
Red Hat Enterprise Virtualization / RHEV	Commercial management solution for RHEL / KVM.	Web-based	active		Commercial
oVirt	oVirt is a virtualization management framework constisting of a small host image, the oVirt Node, that provides the libvirt service to host virtual machines, and a robust vm management software stack, controlled by a web-based management interface, the oVirt Server.	web	active	uses libvirt	
openQRM	openQRM is the next generation, open-source Data-center management platform.	web	active	KVM, Xen, VMware and Linux V-Server support	
Abiquo	Abiquo is an open source infrastructure software for the creation and integral management of Public & Private Clouds based on heterogeneous	web	active	KVM, Xen & Virtual Box support; uses libvirt	

	environments.				
SolusVM	The most popular control panel for commercial use.	Web	active	KVM, Xen & OpenVZ support	

References

[1] J. Campbell, “Speaker recognition: a tutorial,” *Proc. IEEE*, vol. 85, pp. 1437–1462, Sept. 1997.

[2] D. A. Reynolds, T. Quatieri, and R. Dunn, “Speaker verification using adapted Gaussian mixture models,” *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000.

[3] D. A. Reynolds, “Comparison of background normalization methods for text-independent speaker verification,” in *Proc. Eurospeech*, 1997.

[4] D. A. Reynolds and R. C. Rose, “Robust text-independent speaker identification using Gaussian mixture speaker models,” *IEEE Trans. Speech Audio Processing*, vol. 3, no. 1, pp. 72–83, 1995.

[5] J. L. Gauvain and C.-H. Lee, “Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains,” *IEEE Trans. Speech Audio Processing*, vol. 2, pp. 291–298, Apr. 1994.

[6] E. Bocchieri, “Vector quantization for the efficient computation of continuous density likelihoods,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1993, pp. 692–695.

[7] K. M. Knill, M. J. F. Gales, and S. J. Young, “Use of Gaussian selection in large vocabulary continuous speech recognition using HMMs,” in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[8] D. B. Paul, “An investigation of Gaussian shortlists,” in *Proc. Automatic Speech Recognition and Understanding Workshop*, 1999.

[9] T. Watanabe, K. Shinoda, K. Takagi, and K.-I. Iso, “High speed speech recognition using tree-structured probability density function,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1995.

[10] J. Simonin, L. Delphin-Poulat, and G. Damnati, “Gaussian density tree structure in a multi-Gaussian HMM-based speech recognition system,”

[11] T. J. Hanzen and A. K. Halberstadt, “Using aggregation to improve the performance of mixture Gaussian acoustic models,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1998.

[12] M. Padmanabhan, L. R. ahl, and D. Nahamoo, “Partitioning the feature space of a classifier with linear hyperplanes,” *IEEE Trans. Speech Audio* in *Proc. Int. Conf. Spoken Language Processing*, 1998. *Processing*, vol. 7, no. 3, pp. 282–288, 1999.

[13] R. Auckenthaler and J. Mason, “Gaussian selection applied to text-independent speaker verification,” in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[14] J. McLaughlin, D. Reynolds, and T. Gleason, “A study of computation speed-ups of the GMM-UBM speaker recognition system,” in *Proc. Eurospeech*, 1999.

[15] S. van Vuuren and H. Hermansky, “On the importance of components of the modulation spectrum of speaker verification,” in *Proc. Int. Conf. Spoken Language Processing*, 1998.

[16] B. L. Pellom and J. H. L. Hansen, “An efficient scoring algorithm for Gaussian mixture model based speaker identification,” *IEEE Signal Processing Lett.*, vol. 5, no. 11, pp. 281–284, 1998.

[17] J. Oglesby and J. S. Mason, “Optimization of neural models for speaker identification,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 1990, pp. 261–264.

[18] Y. Bengio, R. De Mori, G. Flammia, and R. Kompe, “Global optimization of a neural network—hidden Markov model hybrid,” *IEEE Trans. Neural Networks*, vol. 3, no. 2, pp. 252–259, 1992.

[19] H. Bourlard and C. J. Wellekins, “Links between Markov models and multilayer perceptrons,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, pp. 1167–1178, Dec. 1990.

[20] J. Navrátil, U. V. Chaudhari, and G. N. Ramaswamy, “Speaker verification using target and background dependent linear transforms and multi-system fusion,” in *Proc. Eurospeech*, 2001.

[21] L. P. Heck, Y. Konig, M. K. Sonmez, and M. Weintraub, “Robustness to telephone handset distortion in speaker recognition by discriminative feature design,” *Speech Commun.*, vol. 31, pp. 181–192, 2000.

[22] A. Dempster, N. Laird, and D. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,” *J. R. Statist. Soc.*, vol. 39, pp. 1–38, 1977.

[23] K. Shinoda and C. H. Lee, “A structural Bayes approach to speaker adaptation,” *IEEE Trans. Speech Audio Processing*, vol. 9, no. 3, pp. 276–287, 2001.

[24] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. New York: Academic, 1990.

[25] J. C. Junqua, *Robust Speech Recogntion in Embedded Systems and PC*

[26] U. V. Chaudhari, J. Navrátil, S. H. Maes, and R. A. Gopinath, “Transformation enhanced multi-grained modeling for text-independent speaker recognition,” in *Proc. Int. Conf. Spoken Language Processing*, 2000.

[27] Q. Lin, E.-E. Jan, C. W. Che, D.-S. Yuk, and J. Flanagan, “Selective use of the speech spectrum and a VQGMM method for speaker identification,” in *Proc. Int. Conf. Spoken Language Processing*, 1996.

[28] S. Raudys, *Statistical and Neural Classifiers: An Integrated Approach to Design*. New York: Springer, 2001.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing*. Cambridge, MA: MIT Press, 1986, pp. 318–364.

[30] [Online] Available: http://www.nist.gov/speech/tests/spk/index.htm.

[31] J. Pelecanos and S. Sridharan, “Feature warping for robust speaker verification,” in *Proc. A Speaker Odyssey—Speaker Recognition Workshop*, 2001.

[32] B. Xiang, U. V. Chaudhari, J. Navrátil, N. Ramaswamy, and R. A. Gopinath, “Short-time Gaussianization for robust speaker verification,” in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2002.

[33] G. R. Doddington, M. A. Przybocki, A. F. Martin, and D. A. Reynolds, “The NIST speaker recognition evaluation—overview, methodology, systems, results, perspective,” *Speech Communication*, vol. 31, pp. 225–254, 2000.

Bing Xiang (M’03) was born in 1973 in China. He received the B.S. degree in radio and electronics and M.E. degree in signal and information processing from Peking University in 1995 and 1998, respectively. In January, 2003, he received the Ph.D. degree in electrical engineering from Cornell University, Ithaca, NY.

From 1995 to 1998, he worked on speaker recognition and auditory modeling in National Laboratory on Machine Perception, Peking University. Then he entered Cornell University and worked on speaker recognition and speech recognition in DISCOVER Lab as a Research Assistant. He also worked in the Human Language Technology Department of IBM Thomas J. Watson Research Center as a summer intern in both 2000 and 2001. He was a selected remote member of the SuperSID Group in the 2002 Johns Hopkins CLSP summer workshop in which he worked on speaker verification with high-lelvel information. In January, 2003, he joined the Speech and Language Processing Department of BBN Technologies where he is presently a Senior Staff Consultant-Technology. His research interests include large vocabulary speech recognition, speaker recognition, speech synthesis, keyword spotting, neural networks and statistical pattern recognition.

Toby Berger (S’60–M’66–SM’74–F’78) was born in New York, NY, on September 4, 1940. He received the B.E. degree in electrical engineering from Yale University, New Haven, CT in 1962, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA in 1964 and 1966, respectively.

From 1962 to 1968 he was a Senior Scientist at Raytheon Company, Wayland, MA, specializing in communication theory, information theory, and coherent signal processing. In 1968 he joined the faculty of Cornell University, Ithaca, NY where he is presently the Irwin and Joan Jacobs Professor of Engineering. His research interests include information theory, random fields, communication networks, wireless communications, video compression, voice and signature compression and verification, neuroinformation theory, quantum information theory, and coherent signal processing. He is the author/co-author of Rate Distortion Theory: A Mathematical Basis for Data Compression, Digital Compression for Multimedia: Principles and Standards, and Information Measures for Discrete Random Fields.

Dr. Berger has served as editor-in-chief of the IEEE Transactions on Information Theory and as president of the IEEE Information Theory