# Fedora 22

# Virtualization Getting Started Guide

Virtualization Documentation

Fedora Documentation Project

# Fedora 22 Virtualization Getting Started Guide
# Virtualization Documentation
# Edition 22

Author Fedora Documentation Project

The Fedora Virtualization Getting Started Guide describes the basics of virtualization and the virtualization products and technologies that are available with Fedora.

# Preface

## 1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

In PDF and paper editions, this manual uses typefaces drawn from the *Liberation Fonts*[1] set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

### 1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

**`Mono-spaced Bold`**

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keycaps and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a keycap, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from keycaps by the hyphen connecting each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to the first virtual terminal. Press **`Ctrl`**+**`Alt`**+**`F1`** to return to your X-Windows session.

The first paragraph highlights the particular keycap to press. The second highlights two key combinations (each a set of three keycaps with each set pressed simultaneously).

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

**Proportional Bold**

This denotes words or phrases encountered on a system, including application names; dialog box text; labeled buttons; check-box and radio button labels; menu titles and sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click the **Left-handed mouse** check box and click

[1] https://fedorahosted.org/liberation-fonts/

> **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the **Character Table**. Double-click this highlighted character to place it in the **Text to copy** field and then click the **Copy** button. Now switch back to your document and choose **Edit** → **Paste** from the **gedit** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

***`Mono-spaced Bold Italic`*** or ***`Proportional Bold Italic`***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above — username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

## 1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
```

```
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

## 1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

**Note**

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

**Important**

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled 'Important' will not cause data loss but may cause irritation and frustration.

**Warning**

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

## 2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: *http://bugzilla.redhat.com/bugzilla/* against the product **Fedora 22.**

When submitting a bug report, be sure to mention the manual's identifier: *virtualization-getting-started-guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

# Introduction

The *Fedora Virtualization Getting Started Guide* introduces the basics of virtualization and assists with the navigation of other virtualization documentation and products that Fedora provides.

This guide also explains the advantages of virtualization and dispels some common myths that exist regarding virtualization.

## 1.1. Who should read this guide?

This guide is designed for anyone wishing to understand the basics of virtualization, but may be of particular interest to:

- Those who are new to virtualization and seeking knowledge about the benefits offered.
- Those considering deployment of virtualized machines in their environment.
- Those looking for an overview of the virtualization technologies that Fedora produces and supports.

## 1.2. Virtualization in Fedora Linux

Fedora contains packages and tools to support a variety of virtualized environments.

Virtualization in Fedora is carried out by KVM (Kernel-based Virtual Machine). KVM is a full virtualization solution built into Fedora.

Refer to *Chapter 4, Introduction to Fedora virtualization products* for more about the virtualization products available in Fedora.

## 1.3. Virtualization resources

Fedora contains packages and tools to support a variety of virtualized environments. Fedora virtualization provides the upstream development for virtualization in Red Hat Enterprise Linux. Refer to *Chapter 4, Introduction to Fedora virtualization products* for more information about the virtualization products available in Fedora.

# What is virtualization and migration?

This chapter discusses terms related to virtualization and migration.

## 2.1. What is virtualization?

*Virtualization* is a broad computing term used for running software, usually multiple operating systems, concurrently and in isolation from other programs on a single system. Most existing implementations of virtualization use a *hypervisor*, a software layer or subsystem that controls hardware and provides *guest operating systems* with access to underlying hardware. The hypervisor allows multiple operating systems, called *guests*, to run on the same physical system by offering virtualized hardware to the guest operating system. There are several methods for virtualizing operating systems.

Virtualization methods

**Full virtualization**

Full virtualization uses the hardware features of the processor to provide guests with total abstraction of the underlying physical system. This creates a new virtual system, called a *virtual machine*, that allows guest operating systems to run without modifications. The guest operating system and any applications on the guest virtual machine are unaware of their virtualized environment and run normally. Hardware-assisted virtualization is the technique used for full virtualization with KVM (Kernel-based Virtual Machine) in Fedora.

**Para-virtualization**

Para-virtualization employs a collection of software and data structures that are presented to the virtualized guest, requiring software modifications in the guest to use the para-virtualized environment. Para-virtualization can encompass the entire kernel, as is the case for Xen para-virtualized guests, or drivers that virtualize I/O devices.

**Software virtualization (or emulation)**

Software virtualization uses slower binary translation and other emulation techniques to run unmodified operating systems.

**Note**

For more information and detailed instructions on guest installation, refer to the *Fedora Virtualization Deployment and Administration Guide*.

## 2.2. What is migration?

*Migration* describes the process of moving a guest virtual machine from one host to another. This is possible because guests are running in a virtualized environment instead of directly on the hardware. There are two ways to migrate a virtual machine: live and offline.

Migration types

**Offline migration**

An offline migration suspends the guest virtual machine, and then moves an image of the virtual machine's memory to the destination host. The virtual machine is then resumed on the destination host and the memory used by the virtual machine on the source host is freed.

**Live migration**
: Live migration is the process of migrating an active virtual machine from one physical host to another.

It is important to understand that the migration process moves the virtual machine's memory, and the disk volume associated with the virtual machine is also migrated. This process is done using live block migration.

In Fedora 19, shared storage is not necessary for storing guest images to be migrated. With live storage migration, a running virtual machine can be migrated from one host to another with no downtime. This capability can be used to optimize performance of virtual machines.

### 2.2.1. Benefits of migrating virtual machines

Migration is useful for:

**Load balancing**
: When a host machine is overloaded, one or many of its virtual machines could be migrated to other hosts using live migration.

**Upgrading or making changes to the host**
: When the need arises to upgrade, add, or remove hardware devices on one host, virtual machines can be safely relocated to other hosts. This means that guests do not experience any downtime due to changes that are made to any of the hosts.

**Energy saving**
: Virtual machines can be redistributed to other hosts and the unloaded host systems can be powered off to save energy and cut costs in low usage periods.

**Geographic migration**
: Virtual machines can be moved to another physical location for lower latency or for other special circumstances.

Shared, networked storage must be used for storing guest images to be migrated. Without shared storage, migration is not possible. It is recommended to use **libvirt**-managed storage pools for shared storage.

**Note**

For more information on migration, refer to the *Fedora Virtualization Deployment and Administration Guide*.

# Advantages and misconceptions of virtualization

There are many advantages to virtualization and perhaps an equal amount of misconceptions surrounding it. This chapter explores these points.

## 3.1. Virtualization costs

A common misconception is that virtualization is too expensive to justify the change. Virtualization can be expensive to introduce but often it saves money in the long term. It is important to perform a Return on Investment (ROI) analysis to determine the best use of virtualization in your environment. Consider the following benefits:

Less power

> Using virtualization negates much of the need for multiple physical platforms. This equates to less power being drawn for machine operation and cooling, resulting in reduced energy costs. The initial cost of purchasing multiple physical platforms, combined with the machines' power consumption and required cooling, is drastically cut by using virtualization.

Less maintenance

> Provided adequate planning is performed before migrating physical systems to virtualized ones, less time is spent maintaining them. This means less money being spent on parts and labor.

Extended life for installed software

> Older versions of software may not run on newer, bare metal machines directly. However, by running the older software virtually on a larger, faster system, the life of the software may be extended while taking advantage of the performance from the newer system.

Smaller footprint

> Consolidating servers onto fewer machines means less physical space is required. This means the space normally occupied by server hardware can be used for other purposes.

## 3.2. Virtualization learning curve

A misconception exists that virtualization is difficult to learn. In truth, virtualization is no more difficult or easy to learn than any new process. The skills required for managing and supporting a physical environment are easily transferable to a virtual one. Virtual environments function similarly to their physical counterparts, ensuring the learning curve remains a slight one.

## 3.3. Performance

On older virtualization versions that supported only a single CPU, virtual machines experienced noticeable performance limitations. This created a long-lasting misconception that virtualization solutions are slow. This is no longer the case; advances in technology allow virtual machines to run at much faster speeds than previously.

## 3.4. Flexibility

Virtualization provides greater flexibility for managing systems. Virtual machines can be copied or moved to test software updates and validate configuration changes, without impacting other systems. Because each of the virtualized systems are completely separate to each other, one system's downtime will not affect any others.

## 3.5. Disaster recovery

Disaster recovery is quicker and easier when the systems are virtualized. On a physical system, if something serious goes wrong, a complete re-install of the operating system is usually required, resulting in hours of recovery time. However, if the systems are virtualized this is much faster due to migration ability. If the requirements for live migration are followed, virtual machines can be restarted on another host, and the longest possible delay would be in restoring guest data.

## 3.6. Security

A virtual machine uses SELinux and sVirt to improve security in virtualization. This section includes an overview of the security options available.

### 3.6.1. Virtualization security features

#### SELinux

SELinux was developed by the US National Security Agency and others to provide Mandatory Access Control (MAC) for Linux. Under control of SELinux, all processes and files are given what is known as a *type*, and access is limited by fine-grained controls. SELinux limits the abilities of an attacker and works to prevent many common security exploits such as buffer overflow attacks and privilege escalation.

SELinux strengthens the security model of Fedora hosts and virtualized Fedora guests. SELinux is configured and tested to work, by default, with all virtualization tools shipped with Fedora.

#### sVirt

sVirt is a technology included in Fedora that integrates SELinux and virtualization. It applies Mandatory Access Control (MAC) to improve security when using virtual machines, and improves security and hardens the system against bugs in the hypervisor that might be used as an attack vector for the host or to another virtual machine.

## 3.7. Virtualization for servers and individuals

Virtualization is not just for servers; it can be useful for individuals as well. Desktop virtualization offers centralized management, an improved desktop solution, and better disaster recovery. By using connection software, it is possible to connect to a desktop remotely.

For servers, virtualization is not only for larger networks, but for any situation with two or more servers. It provides live migration, high availability, fault tolerance, and streamlined backups.

# Introduction to Fedora virtualization products

This chapter introduces the various virtualization products available in Fedora.

## 4.1. KVM and virtualization in Fedora

What is KVM?

KVM (Kernel-based Virtual Machine) is a full virtualization solution for Linux on AMD64 and Intel 64 hardware that is built into the standard Fedora kernel. It can run multiple, unmodified Windows and Linux guest operating systems. The KVM hypervisor in Fedora is managed with the **libvirt** API and tools built for **libvirt** (such as **`virt-manager`** and **`virsh`**). Virtual machines are executed and run as multi-threaded Linux processes controlled by these tools.

Overcommitting

KVM hypervisor supports *overcommitting* of system resources. Overcommitting means allocating more virtualized CPUs or memory than the available resources on the system. Memory overcommitting allows hosts to utilize memory and virtual memory to increase guest densities.

**Important**

Overcommitting involves possible risks to system stability.

Thin provisioning

Thin provisioning allows the allocation of flexible storage and optimizes the available space for every guest. It gives the appearance that there is more physical storage on the guest than is actually available. This is not the same as overcommitting as this only pertains to storage and not CPUs or memory allocations. However, like overcommitting, the same warning applies.

**Important**

Thin provisioning involves possible risks to system stability.

KSM

*Kernel SamePage Merging (KSM)*, used by the KVM hypervisor, allows KVM guests to share identical memory pages. These shared pages are usually common libraries or other identical, high-use data. KSM allows for greater guest density of identical or similar guest operating systems by avoiding memory duplication.

QEMU guest agent

The *QEMU guest agent* runs on the guest operating system and allows the host machine to issue commands to the guest operating system.

KVM guest virtual machine compatibility

KVM requires a CPU with virtualization extensions, found on most modern consumer CPUs. These extensions are called Intel VT or AMD-V.

## 4.2. libvirt and libvirt tools

The *libvirt* package is a hypervisor-independent virtualization API that is able to interact with the virtualization capabilities of a range of operating systems.

The *libvirt* package provides:

- A common, generic, and stable layer to securely manage virtual machines on a host.
- A common interface for managing local systems and networked hosts.
- All of the APIs required to provision, create, modify, monitor, control, migrate, and stop virtual machines, but only if the hypervisor supports these operations. Although multiple hosts may be accessed with **libvirt** simultaneously, the APIs are limited to single node operations.

The *libvirt* package is designed as a building block for higher level management tools and applications, for example, **`virt-manager`** and the **`virsh`** command-line management tools. With the exception of migration capabilities, **libvirt** focuses on managing single hosts and provides APIs to enumerate, monitor and use the resources available on the managed node, including CPUs, memory, storage, networking and Non-Uniform Memory Access (NUMA) partitions. The management tools can be located on separate physical machines from the host using secure protocols. **libvirt** is the the foundation of the Gnome application: **gnome-boxes** is built apon.

Fedora supports **libvirt** and included **libvirt**-based tools as its default method for virtualization management.

The *libvirt* package is available as free software under the GNU Lesser General Public License. The *libvirt* project aims to provide a long term stable C API to virtualization management tools, running on top of varying hypervisor technologies.

virsh
: The **`virsh`** command-line tool is built on the **libvirt** management API and operates as an alternative to the graphical **`virt-manager`** application. The **`virsh`** command can be used in read-only mode by unprivileged users or, with root access, full administration functionality. The **`virsh`** command is ideal for scripting virtualization administration.

virt-manager
: **`virt-manager`** is a graphical desktop tool for managing virtual machines. It allows access to graphical guest consoles and can be used to perform virtualization administration, virtual machine creation, migration, and configuration tasks. The ability to view virtual machines, host statistics, device information and performance graphs is also provided. The local hypervisor and remote hypervisors can be managed through a single interface.

virt-install
: **`virt-install`** is a command line tool to provision new virtual machines. It supports both text-based and graphical installations, using serial console, SDL, SPICE, or VNC client/server pair graphics. Installation media can be local, or exist remotely on an NFS, HTTP, or FTP server. The tool can also be configured to run unattended and kickstart the guest when installation is complete, allowing for easy automation of installation.

## 4.3. Boxes

*Boxes* is a lightweight graphical desktop virtualization tool used to view and access virtual machines and remote systems. It provides a way to test different operating systems and applications from the desktop with minimal configuration. **Boxes** is based on QEMU and is included in Fedora Workstation.

## 4.4. Storage

Storage for virtual machines is abstracted from the physical storage used by the virtual machine. It is attached to the virtual machine using the para-virtualized or emulated block device drivers.

### 4.4.1. Storage pools

A *storage pool* is a file, directory, or storage device managed by **libvirt** for the purpose of providing storage to virtual machines. Storage pools are divided into storage *volumes* that store virtual machine images or are attached to virtual machines as additional storage. Multiple guests can share the same storage pool, allowing for better allocation of storage resources.

Local storage pools

Local storage pools are directly attached to the host server. They 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 virtual machines. Local storage pools may not be suitable for many production environments as they do not support live migration.

Networked (shared) storage pools

Networked storage pools include storage devices shared over a network using standard protocols. Networked storage is required when migrating virtual machines between hosts with **virt-manager**, but is optional when migrating with **`virsh`**. Networked storage pools are managed by **libvirt**.

### 4.4.2. Storage volumes

Storage pools are further 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 virtual machines as local storage devices regardless of the underlying hardware.

# Introduction to Boxes

Boxes is a simple graphical interface for managing and using virtual machines. Boxes can also connect to computers via VNC, SPICE, and Quemu.

Boxes uses **qemu-kvm**, **libvirt-glib**, and **spice-gtk** to allow users to easily manage virtual machines and connect to remote machines.

## 5.1. Features of Boxes

- Create, access, and manage, local virtual machines.
- Connect to remote machines (called connections) via SPICE, Quemu, or VNC protocols.
- Select and create favorite virtual machines or connections.

## 5.2. How do I create a virtual machine in Boxes?

Procedure 5.1. Create a Virtual Machine In Boxes

1. Download an ISO image of an operating system that will be used in the virtual machine.
2. Launch Boxes from the application launcher, super key, or terminal.
3. Click the **New** Button.
4. Read the introduction and click **continue** in the upper right hand corner.
5. Select the ISO Image file that was previously downloaded to the Downloads folder, otherwise click **select a file** to find an ISO image file located somewhere else. *Boxes will try to auto-detect ISO files in your Downloads folder, but it may not always be successful.*
6. Boxes will then auto create settings, they can be edited by clicking the **customize** button.
7. Memory and Disk allocation can be changed via the customize menu.
8. Click **Create**.
9. The new virtual machine will now boot.

## 5.3. How do I connect to other computers in Boxes?

**Note**

Boxes Supports three protocols: SPICE, Qemu, and VNC.

Procedure 5.2. To connect to another computer using Boxes:

1. Open Boxes.
2. Click **Continue** while the introduction is displayed.
3. Click **Enter URL**.

4. Enter the IP address or hostname of the remote machine you want to connect to. Remember to specify the protocol in the field. For example to connect to a computer via VNC at IP Address 192.168.1.115, you would type in *vnc://192.168.1.115*.

5. Click **Continue**.

6. The Review screen will now show you the type (protocol) and host (either hostname or IP address) of the remote connection.

7. You can click **customize** at this screen to access different options that are available per protocol, for example USB redirection is available for the spice protocol. Or Read only is available for VNC.

8. Click **Create**.

9. The Remote connection is now available in the selection screen.

## 5.4. How do I change the settings of a machine in boxes?

- On the listing of machines, right click on the machine and then choose **Properties**.

- If the machine is running, click on the **screwdriver and wrench icon**.

Once you have accessed the settings menu you can change various items of the virtual machine:

- Sharing the clipboard

- Resize Guest

- Redirect New USB Devices to the Virtual Machine

- Redirect currently plugged in USB devices to the Virtual machine

- Create Snapshots of the Virtual Machine from the snapshot heading

- Force Shutdown, by clicking "Force Shutdown" in the lower left hand corner

To exit the settings menu, *press the arrow pointing **back** or to the left on the top left hand side of the menu bar*. This will return you to your virtual machine console.

## 5.5. How do I move a Virtual Machine between computers?

**Note**

$USER is a variable or place holder for whatever your username is on your system. For example: if your username is glen, /home/$USER would mean /home/glen

Moving a virtual machine has three basic steps:

1. Move or copy the hard drive file that is kept in:**`/home/$USER/.local/share/gnome-boxes/images`**

2. Export of the XML file via **`VIRSH dumpxml`**.

3. Import of the XML file via **`VIRSH create`**.

**Procedure 5.3. Move a Virtual Machine Hard drive "image" file between Computers**

1. Open your home folder via files
2. Press CTRL+H to view hidden files.
3. Open the *.local* folder.
4. Open the *share* folder.
5. Open the *gnome-boxes* folder.
6. Open the *images* folder.
7. If You only have one virtual machine it will be called *boxes-unknown* by default. *Right Click* on the file of the virtual machine and copy it to the USB drive, network drive, or what ever media you will be using to copy your Virtual Machine to the other computer.
8. On the receiving computer, do the steps 1-6 again, but this time you will be copying/pasting the drive image you found in step 7 into the folder: **`/home/$USER/.local/share/gnome-boxes/images`**

**Procedure 5.4. Copy the Virtual Machine "Config" XML file between Computers**

1. Open a terminal window.
2. type in **`virsh list --all`**
3. If you only have one Virtual machine the default will be called *boxes-unknown*.
4. To export the *boxes-unknown* machine configuration, issue the command: **`virsh dumpxml boxes-unknown >~/boxes-unknown.xml`** This will place the xml config file in the root of your home folder. So it would be located at: **`/home/$USER/boxes-unknown.xml`**
5. Now that the configuration XML file "boxes-unknown.xml" is in the root of your home directory, copy it to the USB drive, network drive, or what ever media you will be using to copy your Virtual Machine to the other/receiving computer.
6. On the receiving computer, copy the boxes-unknown.xml into the root of your home folder: **`/home/$USER/boxes-unknown.xml`**
7. On the receiving computer, open a terminal window.
8. In the terminal window issue the command: **`virsh create boxes-unknown.xml`**

Now Your virtual machine should be ready, open Boxes on the receiving / destination computer and turn it on

## 5.6. How do I delete a box?

Deleting a box is simple:

**Procedure 5.5. Delete a Box**

1. Open Boxes.
2. Right Click the Box or Boxes you wish to delete.
3. After the Boxes have a check mark on them that you wish to delete, click the **Delete** button.

4. Click the **X** on the confirmation notification if you are sure you have deleted the correct boxes. Or click **UNDO** if you made a mistake.

## 5.7. Boxes Tips and Tricks

- Boxes uses **libvertd** and many **libvertd** commands can be used for Boxes virtual machines.
- Boxes keeps the virtual machine disks or images in:**`/home/$USER/.local/share/gnome-boxes/images`**

## 5.8. Advanced Commands in Boxes

You can start and stop virtual machines from the command line as well as using the boxes interface .

- **`virsh shutdown`** or **`virsh reboot`** will use ACPI to shutdown or reboot the virtual machine.
- **`virsh destroy`** will mimic if you pulled the power from a running virtual machine.
- **`virsh start`** will power on or start the virtual machine.

**Warning**

Do not use the **`virsh destory`** often or corruption and data loss will occur!

# Creating and Managing Guests with Virt-Manager

This chapter describes how to install the virtualization packages for virt-manager and the KVM hypervisor, and how to create guest virtual machines.

## 6.1. System Requirements

The common system requirements for virtualization on Fedora are:

- 10GB of hard disk storage per guest
- 1GB of RAM per guest
- 1GHz or faster processor

KVM requires a CPU with virtualization extensions, found on most modern consumer CPUs. These extensions are called Intel VT or AMD-V. To check whether you have proper CPU support, run the command:

```
     $ egrep '^flags.*(vmx|svm)' /proc/cpuinfo
```

If nothing is printed, your system does not support the relevant extensions. You can still use the QEMU/KVM, but the emulator will fall back to software virtualization, which is much slower.

## 6.2. Installing Virtualization package groups

You can install virtualization packages from package groups with the following command:

```
# dnf group install Virtualization
```

**Note**

Note that the **`qemu-img`** package is installed as a dependency of the **`Virtualization`** package group if it is not already installed on the system.

The following describes the Virtualization package:

```
$ dnf group info "Virtualization"

Group: Virtualization
 Description: These packages provide a virtualization environment.
 Mandatory Packages:
   virt-install
 Default Packages:
   libvirt-daemon-config-network
   libvirt-daemon-kvm
   qemu-kvm
   virt-manager
```

```
    virt-viewer
  Optional Packages:
    guestfs-browser
    libguestfs-tools
    python-libguestfs
    virt-top
```

The following list describes some of these packages.

Recommended virtualization packages

*qemu-kvm*
: This package provide the user-level KVM emulator on the host Fedora system.

*virt-install*
: Provides the **`virt-install`** command line tool for creating virtual machines.

*virt-manager*
: **`virt-manager`**, also known as **Virtual Machine Manager**, provides a graphical tool for administering virtual machines.

## 6.3. Network Support

By default libvirt creates a private network for your guests on the host machine. This private network will use a 192.168.x.x subnet and not be reachable directly from the network the host machine is on, but virtual guests can use the host machine as a gateway and can connect out with it. If you need to provide services on your guests that are reachable through other machines on your host network you can use firewalld to forward in specific ports, or you can setup a Bridged env. See the *Networking Guide*[1] for more details.

## 6.4. Creating guests with virt-manager

**`virt-manager`**, also known as Virtual Machine Manager, is a graphical tool for creating and managing guest virtual machines.

Procedure 6.1. Creating a guest virtual machine with **virt-manager**

1. **Open virt-manager**
   Start **`virt-manager`**. Launch the **Virtual Machine Manager** application from the **Applications** menu and **System Tools** submenu. Alternatively, run the **`virt-manager`** command as root.

2. **Optional: Open a remote hypervisor**
   Select the hypervisor and click the **Connect** button to connect to the remote hypervisor.

3. **Create a new virtual machine**
   The **virt-manager** window allows you to create a new virtual machine. Click the **Create a new virtual machine** button (*Figure 6.1, “Virtual Machine Manager window”*) to open the **New VM** wizard.

[1] http://docs.fedoraproject.org/en-US/Fedora/21/html/Networking_Guide/index.html

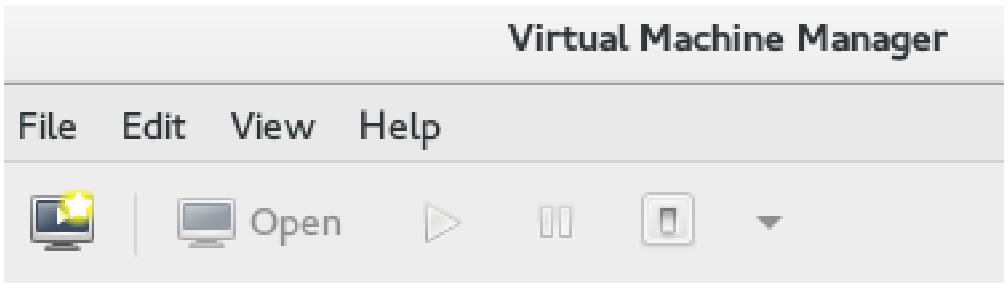


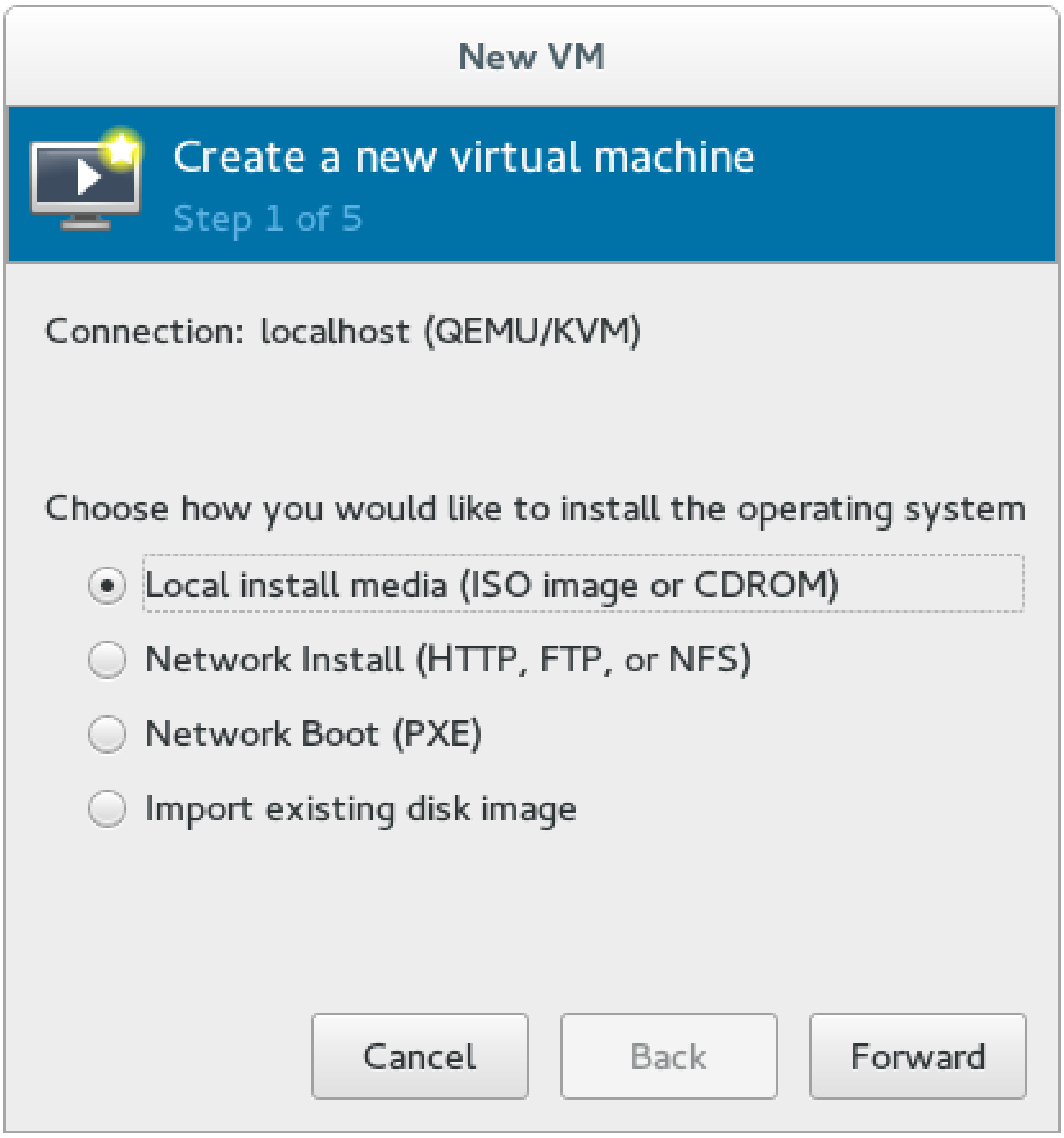
Figure 6.1. Virtual Machine Manager window

The **New VM** wizard breaks down the virtual machine creation process into five steps:

1. Choosing the installation type
2. Naming the guest virtual machine and the installation media
3. Configuring memory and CPU options
4. Configuring the virtual machine's storage
5. Configuring networking, architecture, and other hardware settings

Ensure that **`virt-manager`** can access the installation media (whether locally or over the network) before you continue.

4. **Specify the installation type**
   The guest virtual machine creation process starts with the selection of the installation type.



Connection: localhost (QEMU/KVM)

Choose how you would like to install the operating system

[✓ Local install media (ISO image or CDROM)]
Network Install (HTTP, FTP, or NFS)
Network Boot (PXE)
Import existing disk image

Figure 6.2. Name virtual machine and select installation method

Type in a virtual machine name and choose an installation type:

Local install media (ISO image or CDROM)
: This method uses a CD-ROM, DVD, or image of an installation disk (for example, `.iso`).

Network Install (HTTP, FTP, or NFS)
: This method involves the use of a mirrored Fedora installation tree to install a guest. The installation tree must be accessible through either HTTP, FTP, or NFS.

Network Boot (PXE)
: This method uses a Preboot eXecution Environment (PXE) server to install the guest virtual machine. Setting up a PXE server is covered in the *Virtualization Deployment and Administration Guide*[2]. To install via network boot, the guest must have a routable IP address

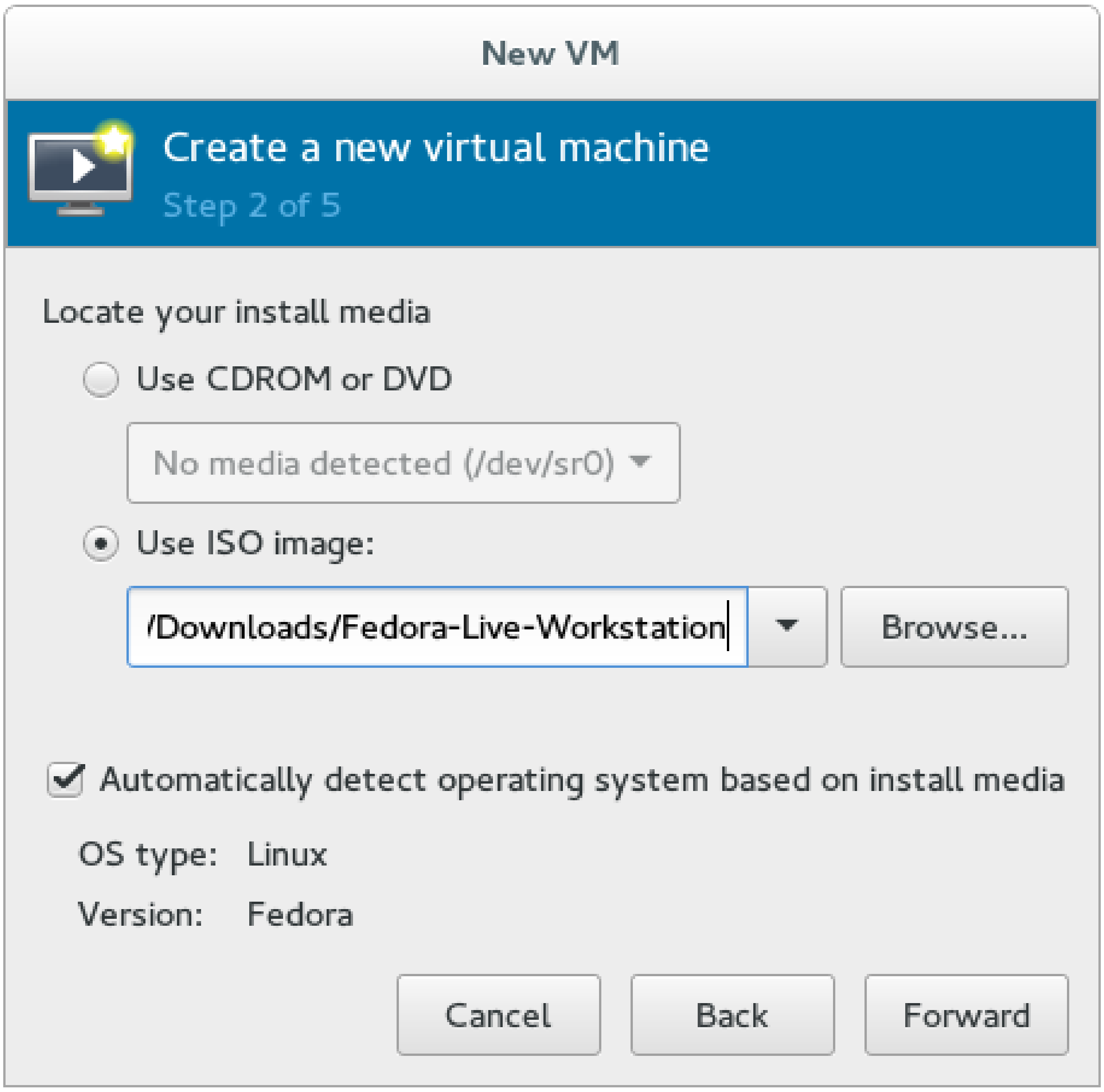
[2] http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/Virtualization_Deployment_and_Administration_Guide/index.html

or shared network device. For information on the required networking configuration for PXE installation, refer to the *Virtualization Deployment and Administration Guide*[3].

Import existing disk image
: This method allows you to create a new guest virtual machine and import a disk image (containing a pre-installed, bootable operating system) to it.

Click **Forward** to continue.

5. **Name the VM and locate the installation media**
   Next, name the virtual machine. Virtual machine names can have underscores (_), periods (`.`), and hyphens (`-`). Virt-manager will automatically detect the OS and version. Ensure that virt-manager selected the appropriate OS type for your virtual machine. Depending on the method of installation, provide the install drive or existing storage path.



Locate your install media

( ) Use CDROM or DVD
No media detected (/dev/sr0)

(●) Use ISO image:
/Downloads/Fedora-Live-Workstation

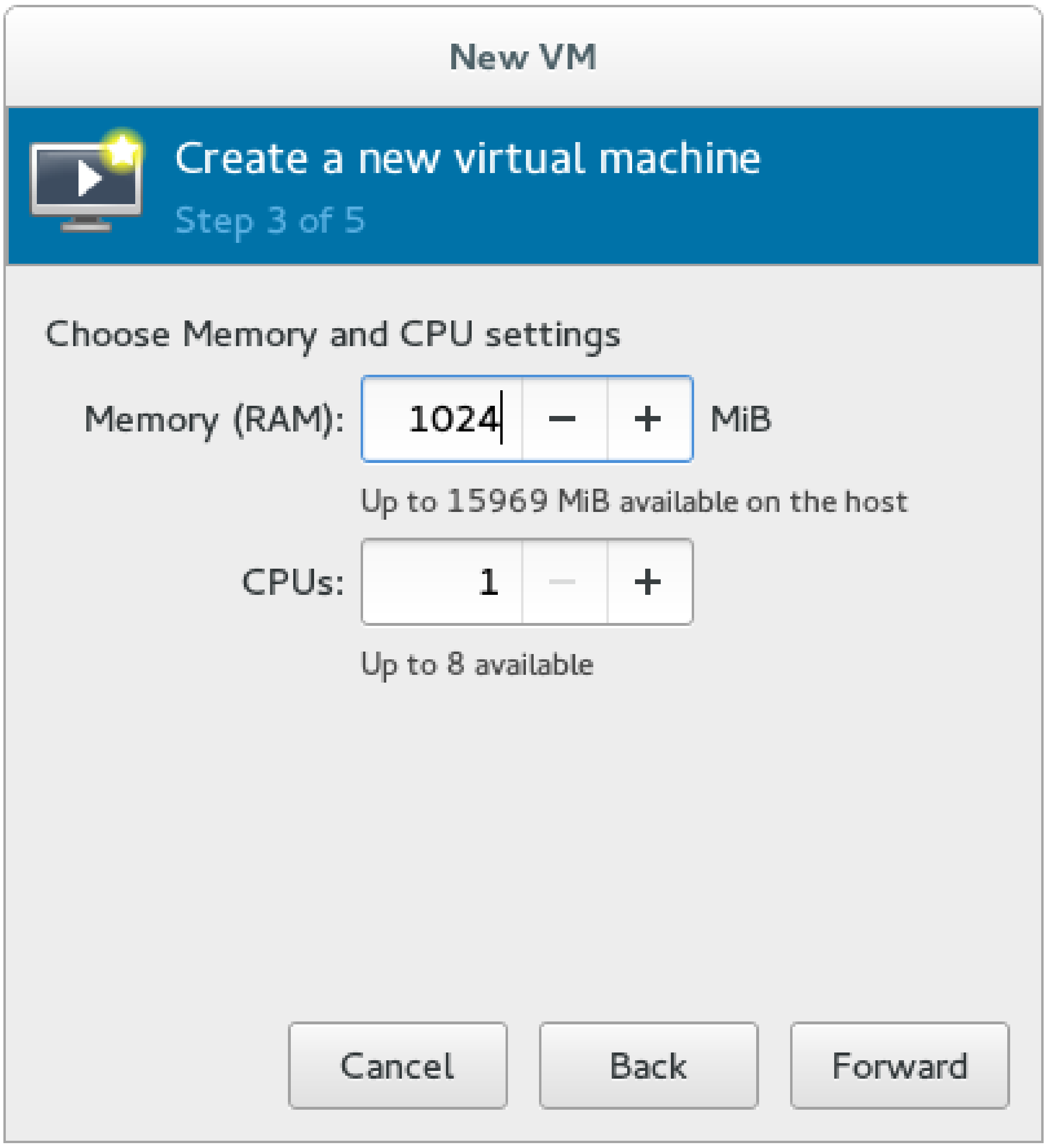
[✓] Automatically detect operating system based on install media

OS type: Linux
Version: Fedora

Figure 6.3. Local ISO image installation

[3] http://docs.fedoraproject.org/en-US/Fedora_Draft_Documentation/0.1/html/Virtualization_Deployment_and_Administration_Guide/index.html

6. **Configure CPU and memory**
   The next step involves configuring the number of CPUs and amount of memory to allocate to the virtual machine. The wizard shows the number of CPUs and amount of memory you can allocate; configure these settings and click **Forward**.


New VM

Create a new virtual machine
Step 3 of 5

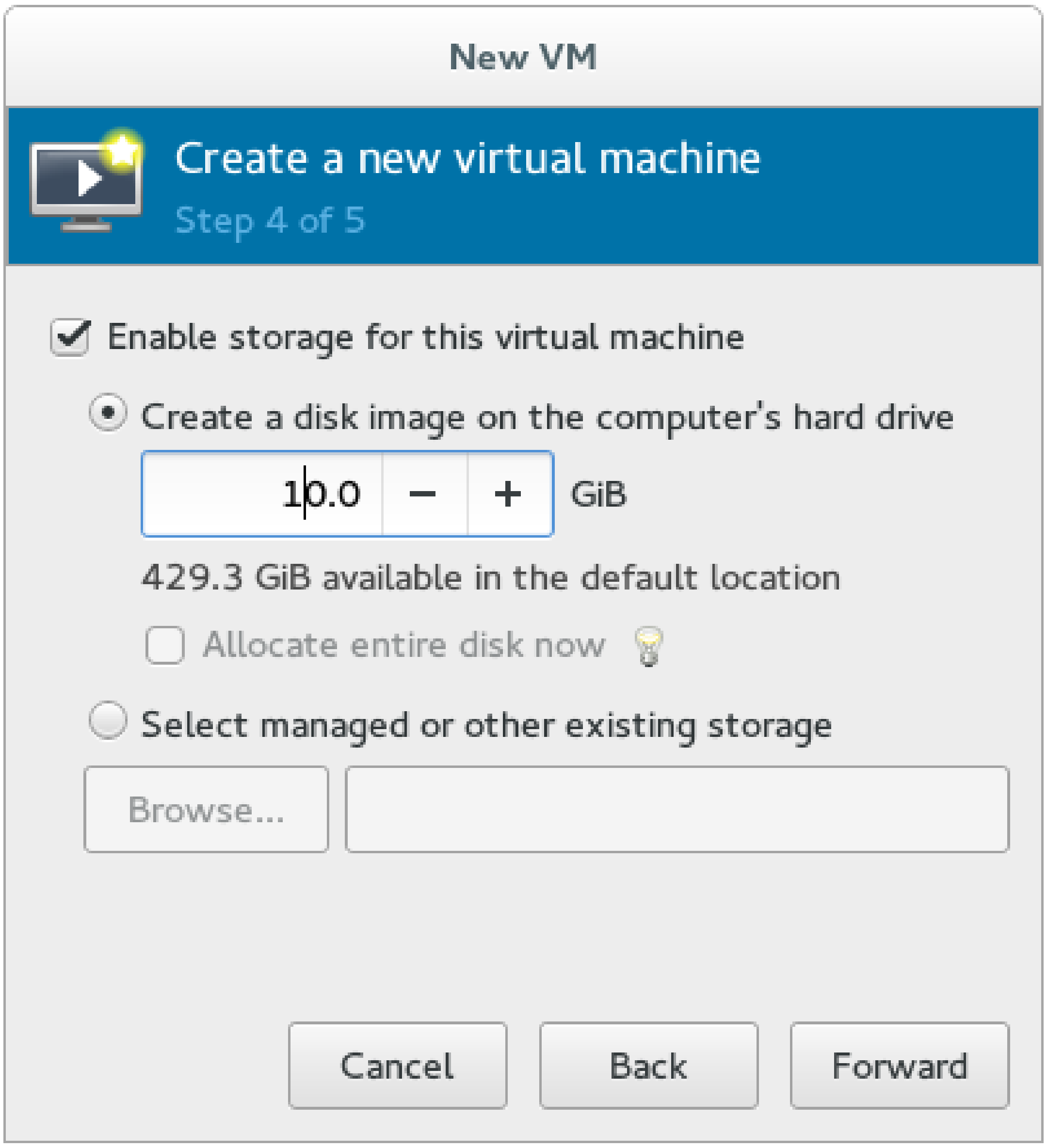
Choose Memory and CPU settings

Memory (RAM): 1024 MiB
Up to 15969 MiB available on the host

CPUs: 1
Up to 8 available

Figure 6.4. Configuring CPU and Memory

7. **Configure storage**
   Assign storage to the guest virtual machine.



☑ Enable storage for this virtual machine

● Create a disk image on the computer's hard drive

10.0 GiB

429.3 GiB available in the default location

☐ Allocate entire disk now

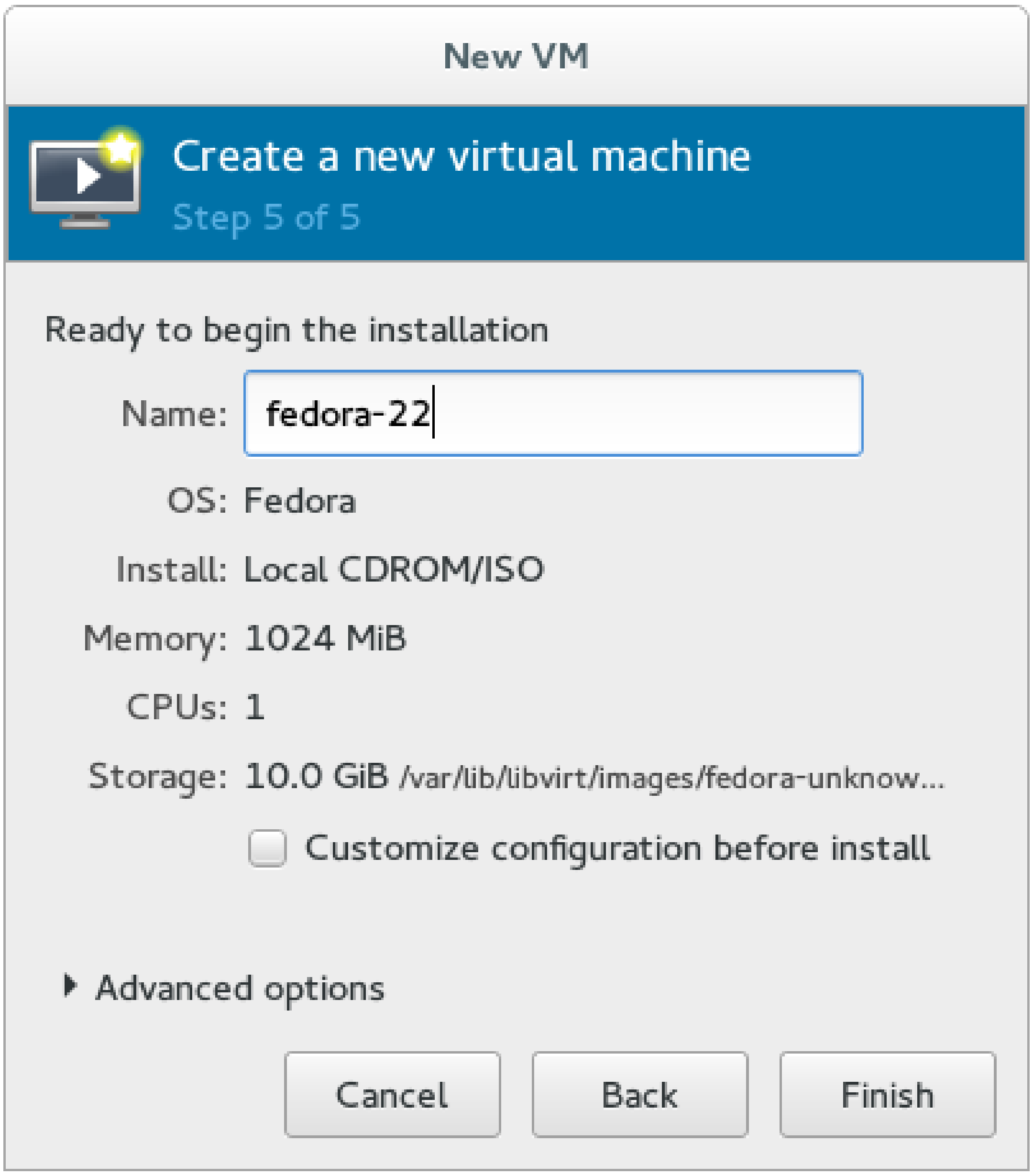
○ Select managed or other existing storage

Figure 6.5. Configuring virtual storage

If you chose to import an existing disk image during the first step, **`virt-manager`** will skip this step.

Assign sufficient space for your virtual machine and any applications it requires, then click **Forward** to continue.

8. **Final configuration**
   Verify the settings of the virtual machine and click **Finish** when you are satisfied; doing so will create the virtual machine with default networking settings, virtualization type, and architecture.


New VM

Create a new virtual machine

Step 5 of 5

Ready to begin the installation

Name: fedora-22

OS: Fedora

Install: Local CDROM/ISO

Memory: 1024 MiB

CPUs: 1

Storage: 10.0 GiB /var/lib/libvirt/images/fedora-unknow...

☐ Customize configuration before install

Figure 6.6. Verifying the configuration

If you prefer to further configure the virtual machine's hardware first, check the **Customize configuration before install** box first before clicking **Finish**. Doing so will open another wizard that will allow you to add, remove, and configure the virtual machine's hardware settings.

After configuring the virtual machine's hardware, click **Apply**. `virt-manager` will then create the virtual machine with your specified hardware settings.

# Appendix A. Advanced Virtualization Concepts

## A.1. Virtualized hardware devices

Virtualization on Fedora presents three distinct types of system devices to virtual machines. The three types include:

- Virtualized and emulated devices
- Para-virtualized devices
- Physically shared devices

These hardware devices all appear as being physically attached to the virtual machine but the device drivers work in different ways.

### A.1.1. Virtualized and emulated devices

KVM implements many core devices for virtual machines in software. These emulated hardware devices are crucial for virtualizing operating systems.

Emulated devices are virtual devices which exist entirely in software.

Emulated drivers may use either a physical device or a virtual software device. Emulated drivers are a translation layer between the virtual machine and the Linux kernel (which manages the source device). The device level instructions are completely translated by the KVM hypervisor. Any device, of the same type (storage, network, keyboard, and mouse) and recognized by the Linux kernel, may be used as the backing source device for the emulated drivers.

Virtual CPUs (vCPUs)

A host system can have up to 160 virtual CPUs (vCPUs) that can be presented to guests for their use, regardless of the number of host CPUs.

Emulated graphics devices

Two emulated graphics devices are provided. These devices can be connected to with the SPICE (Simple Protocol for Independent Computing Environments) protocol or with VNC:

- A Cirrus CLGD 5446 PCI VGA card (using the *cirrus* device)
- A standard VGA graphics card with Bochs VESA extensions (hardware level, including all non-standard modes)

Emulated system components

The following core system components are emulated to provide basic system functions:

- Intel i440FX host PCI bridge
- PIIX3 PCI to ISA bridge
- PS/2 mouse and keyboard
- EvTouch USB graphics tablet
- PCI UHCI USB controller and a virtualized USB hub

- Emulated serial ports
- EHCI controller, virtualized USB storage and a USB mouse
- USB 3.0 xHCI host controller

Emulated sound devices

Fedora provides an emulated (Intel) HDA sound device, `intel-hda`.

The following two emulated sound devices are also available, but are not recommended due to compatibility issues with certain guest operating systems:

- `ac97`, an emulated Intel 82801AA AC97 Audio compatible sound card
- `es1370`, an emulated ENSONIQ AudioPCI ES1370 sound card

Emulated watchdog devices

Fedora provides two emulated watchdog devices. A watchdog can be used to automatically reboot a virtual machine when it becomes overloaded or unresponsive.

The *watchdog* package must be installed on the guest.

The two devices available are:

- `i6300esb`, an emulated Intel 6300 ESB PCI watchdog device.
- `ib700`, an emulated iBase 700 ISA watchdog device.

Emulated network devices

There are two emulated network devices available:

- The `e1000` device emulates an Intel E1000 network adapter (Intel 82540EM, 82573L, 82544GC).
- The `rtl8139` device emulates a Realtek 8139 network adapter.

Emulated storage drivers

Storage devices and storage pools can use these emulated devices to attach storage devices to virtual machines. The guest uses an emulated storage driver to access the storage pool.

Note that like all virtual devices, the storage drivers are not storage devices. The drivers are used to attach a backing storage device, file or storage pool volume to a virtual machine. The backing storage device can be any supported type of storage device, file, or storage pool volume.

The emulated IDE driver

KVM provides two emulated PCI IDE interfaces. An emulated IDE driver can be used to attach any combination of up to four virtualized IDE hard disks or virtualized IDE CD-ROM drives to each virtual machine. The emulated IDE driver is also used for virtualized CD-ROM and DVD-ROM drives.

The emulated floppy disk drive driver

The emulated floppy disk drive driver is used for creating virtualized floppy drives.

## A.1.2. Para-virtualized devices

Para-virtualized devices are drivers for virtual devices that increase the I/O performance of virtual machines.

Para-virtualized devices decrease I/O latency and increase I/O throughput to near bare-metal levels. It is recommended to use the para-virtualized device drivers for virtual machines running I/O intensive applications.

The para-virtualized devices must be installed on the guest operating system. The para-virtualized device drivers must be manually installed on Windows guests.

The para-virtualized network driver (virtio-net)
: The para-virtualized network driver can be used as the driver for existing network devices or new network devices for virtual machines.

The para-virtualized block driver (virtio-blk)
: The para-virtualized block driver is a driver for all storage devices, is supported by the hypervisor, and is attached to the virtual machine (except for floppy disk drives, which must be emulated).

The para-virtualized controller device (virtio-scsi)
: The para-virtualized SCSI controller device provides a more flexible and scalable alternative to virtio-blk. A virtio-scsi guest is capable of inheriting the feature set of the target device, and can handle hundreds of devices compared to virtio-blk, which can only handle 28 devices.

The para-virtualized clock
: Guests using the Time Stamp Counter (TSC) as a clock source may suffer timing issues. KVM works around hosts that do not have a constant Time Stamp Counter by providing guests with a para-virtualized clock.

The para-virtualized serial driver (virtio-serial)
: The para-virtualized serial driver is a bytestream-oriented, character stream driver, and provides a simple communication interface between the host's user space and the guest's user space.

The balloon driver (virtio-balloon)
: The balloon driver can designate part of a virtual machine's RAM as not being used (a process known as balloon *inflation*), so that the memory can be freed for the host (or for other virtual machines on that host) to use. When the virtual machine needs the memory again, the balloon can be *deflated* and the host can distribute the RAM back to the virtual machine.

The para-virtualized random number generator (virtio-rng)
: The para-virtualized random number generator enables virtual machines to collect entropy, or randomness, directly from the host to use for encrypted data and security. Virtual machines can often be starved of entropy because typical inputs (such as hardware usage) are unavailable. Sourcing entropy can be time-consuming; virtio-rng makes this process faster by injecting entropy directly into guest virtual machines from the host.

### A.1.3. Physical host devices

Certain hardware platforms allow virtual machines to directly access various hardware devices and components. This process in virtualization is known as *device assignment*. Device assignment is also known as *passthrough*.

PCI device assignment
: The KVM hypervisor supports attaching PCI devices on the host system to virtual machines. PCI device assignment allows guests to have exclusive access to PCI devices for a range of tasks. It allows PCI devices to appear and behave as if they were physically attached to the guest virtual machine.

Device assignment is supported on PCI Express devices, with the exception of graphics cards. Parallel PCI devices may be supported as assigned devices, but they have severe limitations due to security and system configuration conflicts.

USB passthrough

The KVM hypervisor supports attaching USB devices on the host system to virtual machines. USB device assignment allows guests to have exclusive access to USB devices for a range of tasks. It allows USB devices to appear and behave as if they were physically attached to the virtual machine.

SR-IOV

SR-IOV (Single Root I/O Virtualization) is a PCI Express standard that extends a single physical PCI function to share its PCI resources as separate, virtual functions (VFs). Each function is capable of being used by a different virtual machine via PCI device assignment.

An SR-IOV capable PCI-e device, provides a Single Root Function (for example, a single Ethernet port) and presents multiple, separate virtual devices as unique PCI device functions. Each virtual device may have its own unique PCI configuration space, memory-mapped registers, and individual MSI-based interrupts.

NPIV

N_Port ID Virtualization (NPIV) is a functionality available with some Fibre Channel devices. NPIV shares a single physical N_Port as multiple N_Port IDs. NPIV provides similar functionality for Fibre Channel Host Bus Adapters (HBAs) that SR-IOV provides for PCIe interfaces. With NPIV, virtual machines can be provided with a virtual Fibre Channel initiator to Storage Area Networks (SANs).

NPIV can provide high density virtualized environments with enterprise-level storage solutions.

### A.1.4. CPU models

CPU models define which host CPU features are available to the guest operating system. **qemu-kvm** and **libvirt** contain definitions for a number of current processor models, allowing users to enable features that are available only in newer CPU models. The CPU feature set available to guests depends on support in the host CPU kernel. The **qemu-kvm** code must also allow the feature to be enabled.

To safely migrate virtual machines between hosts with different CPU feature sets, **qemu-kvm** does not expose all CPU features from the host CPU to guest operating systems by default. Instead, CPU features are exposed to virtual machines based on the chosen CPU model.

It is also possible to enable or disable specific CPU features in a virtual machine's XML configuration. However, it is safer to use predefined CPU models, as incorrect configuration can cause compatibility issues with the guest operating system.

## A.2. guestfish

*guestfish* is a command line tool for examining and modifying the file systems of the host. This tool uses *libguestfs* and exposes all functionality provided by the `guestfs` API. This tool ships in its own package entitled *guestfish*.

**Warning**

Using **`guestfish`** on running virtual machines can cause disk-image corruption. Use the **`guestfish`** command with the **`--ro`** (read-only) option if the disk image is being used by a running virtual machine.

## A.3. Other useful tools

The following tools are used to access a virtual machine's disk via the host. The guest's disk is usually accessed directly via the **`disk-image`** file located on the host. However it is sometimes possible to gain access via the **libvirt** domain. The commands that follow are part of the **libvirt** domain and are used to gain access to the guest's disk image.

**`guestmount`**

A command line tool used to mount virtual machine file systems and disk images on the host machine. This tool is installed as part of the *libguestfs-mount* package.

**Warning**

Using **`guestmount`** in **`--r/w`** (read/write) mode to access a disk that is currently being used by a guest can cause the disk to become corrupted. Do not use **`guestmount`** in **`--r/w`** (read/write) mode on live virtual machines. Use the **`guestmount`** command with the **`--ro`** (read-only) option if the disk image is being used.

**`virt-cat`**

A command line tool that can be used to quickly view the contents of one or more files in a specified virtual machine's disk or disk image. This tool is installed as part of the *libguestfs-tools* package.

**`virt-df`**

A command line tool used to show the actual physical disk usage of virtual machines. Similar to the command line tool **`df`**. Note that this tool does not work across remote connections. It is installed as part of the *libguestfs-tools* package.

**`virt-edit`**

A command line tool used to edit files that exist on a specified virtual machine. This tool is installed as part of the *libguestfs-tools* package.

**Warning**

Using **`virt-edit`** on live virtual machines can cause disk corruption in the virtual machine. Although the **`virt-edit`** command will try to prevent users from editing files on live virtual machines, it is not guaranteed to catch all instances. Do not use **`virt-edit`** on a live virtual machine.

**`virt-filesystems`**

A command line tool used to discover file systems, partitions, logical volumes and their sizes in a disk image or virtual machine. One common use is in shell scripts, to iterate over all file systems in a disk image. This tool is installed as part of the *libguestfs-tools* package.

This tool replaces **`virt-list-filesystems`** and **`virt-list-partitions`**.

**`virt-inspector`**

A command line tool that can examine a virtual machine or disk image to determine the version of its operating system and other information. It can also produce XML output, which can be piped into other programs. Note that **`virt-inspector`** can only inspect one domain at a time. This tool is installed as part of the *libguestfs-tools* package.

**`virt-ls`**

A command line tool that lists files and directories inside a virtual machine. This tool is installed as part of the *libguestfs-tools* package.

**`virt-make-fs`**

A command line tool for creating a file system based on a tar archive or files in a directory. It is similar to tools like **`mkisofs`** and **`mksquashfs`**, but it can create common file system types such as ext2, ext3 and NTFS, and the size of the file system created can be equal to or greater than the size of the files it is based on. This tool is provided as part of the *libguestfs-tools* package.

**`virt-rescue`**

A command line tool that provides a rescue shell and some simple recovery tools for unbootable virtual machines and disk images. It can be run on any virtual machine known to **libvirt**, or directly on disk images. This tool is installed as part of the *libguestfs-tools* package.

**Warning**

Using **`virt-rescue`** on running virtual machines can cause disk corruption in the virtual machine. **`virt-rescue`** attempts to prevent its own use on running virtual machines, but cannot catch all cases.

Using the command with the **`--ro`** (read-only) option will not cause disk corruption, but may give strange or inconsistent results. It is better to avoid using **`virt-rescue`** on a running virtual machine.

**`virt-resize`**
: A command line tool to resize virtual machine disks, and resize or delete any partitions on a virtual machine disk. It works by copying the guest image and leaving the original disk image untouched. This tool is installed as part of the *libguestfs-tools* package.

**Important**

Using **`virt-resize`** on running virtual machines can give inconsistent results. It is best to shut down virtual machines before attempting to resize them.

**`virt-top`**
: A command line utility similar to **`top`**, which shows statistics related to virtualized domains. This tool ships in its own package: *virt-top*.

**`virt-v2v`**
: A graphical tool to convert virtual machines from Xen and VMware hypervisors to run on KVM. This tool ships in its own package: *virt-v2v*.

**`virt-viewer`**
: A minimal tool for displaying the graphical console of a virtual machine via the VNC and SPICE protocols. This tool ships in its own package: *virt-viewer*.

**`virt-what`**
: A shell script that detects whether a program is running in a virtual machine. This tool ships in its own package: *virt-what*.

**`virt-who`**
: The *virt-who* package is a Fedora host agent that queries **libvirt** for guest UUIDs. It then passes that data to the local entitlement server for the purposes of issuing certificates. This tool ships in its own package: *virt-who*.

**`virt-win-reg`**
: A command line tool to export and merge Windows Registry entries from a Windows guest, and perform simple Registry operations. This tool is installed as part of the *libguestfs-tools* package.

**Warning**

Using **`virt-win-reg`** on running virtual machines will cause irreversible disk corruption in the virtual machine. **`virt-win-reg`** attempts to prevent its own use on running virtual machines, but cannot catch all cases.

**Warning**

Modifying the Windows Registry is an inherently risky operation, as the format is deliberately obscure and undocumented. Changes to the registry can leave the system unbootable, so ensure you have a reliable backup before you use the `--merge` option.

**`virt-xml-validate`**
A command line tool to validate **libvirt** XML files for compliance with the published schema. This tool is installed as part of the *libvirt-client* package.

# Appendix B. Revision History

**Revision 2.0-0** **Monday, May 18, 2015** **Sandra McCann** *mccann2@fedoraproject.org*

Merged Products and Tools sections to simplify guide flow for new users, and finalized for Fedora 22 release.

**Revision 1.0-16** **Friday March 29, 2015** **Glen Rundblom** *grundblom@fedoraproject.org*

Built out section on how to move a Boxes VM between hosts, as well as how to delete a Box, tested procedure Fedora and LinuxMint machines in my lab as I wrote the section.

**Revision 1.0-15** **Friday March 27, 2015** **Glen Rundblom** *grundblom@fedoraproject.org*

Broke boxes into it's own chapter, edited it for grammer and made it more novice user oriented

**Revision 1.0-14** **Sunday March 04, 2015** **Glen Rundblom** *grundblom@fedoraproject.org*

Edited boxes grammar and added better tags for readability in Products.xml

**Revision 1.0-13** **Sunday March 01, 2015** **Glen Rundblom** *grundblom@fedoraproject.org*

Created Guide on how to create Virtual machine using boxes interface

**Revision 1.0-12** **Wednesday June 12, 2013** **Dayle Parker** *dayleparker@redhat.com*

Publish draft to Fedora docs site.

**Revision 1.0-11** **Monday June 10, 2013** **Dayle Parker** *dayleparker@redhat.com*

Revised Para-virtualized Devices section based on SME feedback.
Verified references to other Fedora virtualization guides.
Added GNOME Boxes description to Tools.

**Revision 1.0-10** **Thursday May 30, 2013** **Dayle Parker** *dayleparker@redhat.com*

Added virtio-rng description to Para-virtualized Devices section.

**Revision 1.0-09** **Monday May 27, 2013** **Dayle Parker** *dayleparker@redhat.com*

Updated CPU Models section based on SME feedback.

**Revision 1.0-08** **Thursday May 9, 2013** **Dayle Parker** *dayleparker@redhat.com*

Rearranged Migration section and included live storage migration feature description.

**Revision 1.0-07** **Monday May 6, 2013** **Dayle Parker** *dayleparker@redhat.com*

Added xHCI host controller to Emulated system components list.

**Revision 1.0-06** **Friday May 3, 2013** **Dayle Parker** *dayleparker@redhat.com*

Made initial general updates for Fedora 19.

**Revision 1.0-05** **Monday October 22, 2012** **Dayle Parker** *dayleparker@redhat.com*

Branch for Fedora 18 Beta.

**Revision 1.0-04** **Monday October 22, 2012** **Dayle Parker** *dayleparker@redhat.com*

Added virtio-scsi feature description to 4.3.2. Para-virtualized devices.

**Revision 1.0-03** **Thursday September 6, 2012** **Dayle Parker** *dayleparker@redhat.com*

In Chapter 3: Advantages, added Flexibility point for (*BZ#853826*[1]).

**Revision 1.0-02** **Thursday August 23 2012** **Dayle Parker** *dayleparker@redhat.com*

In Tools: deleted virt-inspector2, virt-cat warning, clarified --r/w warning as per feedback.

**Revision 1.0-01** **Tuesday August 14 2012** **Dayle Parker** *dayleparker@redhat.com*

Initial creation of book for Fedora.

[1] https://bugzilla.redhat.com/show_bug.cgi?id=853826