

Fedora 31 Essentials

Fedora 31 Essentials

ISBN-13: 978-1-951442-11-8

© 2020 Neil Smyth / Payload Media, Inc. All Rights Reserved.

This book is provided for personal use only. Unauthorized use, reproduction and/or distribution strictly prohibited. All rights reserved.

The content of this book is provided for informational purposes only. Neither the publisher nor the author offers any warranties or representation, express or implied, with regard to the accuracy of information contained in this book, nor do they accept any liability for any loss or damage arising from any errors or omissions.

This book contains trademarked terms that are used solely for editorial purposes and to the benefit of the respective trademark owner. The terms used within this book are not intended as infringement of any trademarks.

Rev: 2.0

Table of Contents

1. Introduction

- 1.1 Superuser Conventions
- 1.2 Feedback
- 1.3 Errata

2. A Brief History of Linux

- 2.1 What exactly is Linux?
- 2.2 UNIX Origins
- 2.3 Who Created Linux?
- 2.4 The Early Days of Red Hat
- 2.5 Red Hat Support
- 2.6 Open Source
- 2.7 The Fedora Project
- 2.8 CentOS - The Free Alternative
- 2.9 Summary

3. Installing Fedora 31 on a Clean Disk Drive

- 3.1 Fedora Installation Options
- 3.2 Choosing an Installation Option
- 3.3 Obtaining the Fedora 31 Installation Media
- 3.4 Writing the ISO Installation Image to a USB Drive
 - 3.4.1 Linux
 - 3.4.2 macOS
 - 3.4.3 Windows
- 3.5 Installing Fedora 31
- 3.6 Partitioning a Disk for Fedora 31
- 3.7 Adding a User and Root Access
- 3.8 The Physical Installation
- 3.9 Final Configuration Steps
- 3.10 Installing Updates
- 3.11 Displaying Boot Messages
- 3.12 Summary

4. Dual Booting Fedora 31 with Windows

- 4.1 Partition Resizing
- 4.2 Changing the Default Boot Option
- 4.3 Accessing the Windows Partition from Fedora 31
- 4.4 Summary

5. Allocating Windows Disk Partitions to Fedora 31

Table of Contents

- 5.1 Unmounting the Windows Partition
- 5.2 Deleting the Windows Partitions from the Disk
- 5.3 Formatting the Unallocated Disk Partition
- 5.4 Mounting the New Partition
- 5.5 Editing the Boot Menu
- 5.6 Summary

6. A Guided Tour of the GNOME 3 Desktop

- 6.1 Installing the GNOME Desktop
- 6.2 An Overview of the GNOME 3 Desktop
- 6.3 Launching Activities
- 6.4 Managing Windows
- 6.5 Using Workspaces
- 6.6 Calendar and Notifications
- 6.7 Desktop Settings
- 6.8 Summary

7. An Overview of the Fedora 31 Cockpit Web Interface

- 7.1 An Overview of Cockpit
- 7.2 Installing and Enabling Cockpit
- 7.3 Accessing Cockpit
- 7.4 System
- 7.5 Logs
- 7.6 Storage
- 7.7 Networking
- 7.8 Accounts
- 7.9 Services
- 7.10 Applications
- 7.11 Virtual Machines
- 7.12 Podman Containers
- 7.13 Diagnostic Reports
- 7.14 Kernel Dump
- 7.15 SELinux
- 7.16 Software Updates
- 7.17 Terminal
- 7.18 Connecting to Multiple Servers
- 7.19 Enabling Stored Metrics
- 7.20 Summary

8. Using the Bash Shell on Fedora 31

- 8.1 What is a Shell?
- 8.2 Gaining Access to the Shell
- 8.3 Entering Commands at the Prompt

- 8.4 Getting Information about a Command
- 8.5 Bash Command-line Editing
- 8.6 Working with the Shell History
- 8.7 Filename Shorthand
- 8.8 Filename and Path Completion
- 8.9 Input and Output Redirection
- 8.10 Working with Pipes in the Bash Shell
- 8.11 Configuring Aliases
- 8.12 Environment Variables
- 8.13 Writing Shell Scripts
- 8.14 Summary
- 9. Managing Fedora 31 Users and Groups**
 - 9.1 User Management from the Command-line
 - 9.2 User Management with Cockpit
 - 9.3 Summary
- 10. Understanding Fedora 31 Software Installation and Management**
 - 10.1 Repositories
 - 10.2 The fedora Repository
 - 10.3 The fedora-modular Repository
 - 10.4 Summary
- 11. Configuring Fedora 31 systemd Units**
 - 11.1 Understanding Fedora 31 systemd Targets
 - 11.2 Understanding Fedora 31 systemd Services
 - 11.3 Fedora 31 systemd Target Descriptions
 - 11.4 Identifying and Configuring the Default Target
 - 11.5 Understanding systemd Units and Unit Types
 - 11.6 Dynamically Changing the Current Target
 - 11.7 Enabling, Disabling and Masking systemd Units
 - 11.8 Working with systemd Units in Cockpit
 - 11.9 Summary
- 12. Fedora 31 Network Management**
 - 12.1 An Introduction to NetworkManager
 - 12.2 Installing and Enabling NetworkManager
 - 12.3 Basic nmcli Commands
 - 12.4 Working with Connection Profiles
 - 12.5 Interactive Editing
 - 12.6 Configuring NetworkManager Permissions
 - 12.7 Summary
- 13. Basic Fedora 31 Firewall Configuration with firewalld**

Table of Contents

- 13.1 An Introduction to firewalld
 - 13.1.1 Zones
 - 13.1.2 Interfaces
 - 13.1.3 Services
 - 13.1.4 Ports
- 13.2 Checking firewalld Status
- 13.3 Configuring Firewall Rules with firewall-cmd
 - 13.3.1 Identifying and Changing the Default Zone
 - 13.3.2 Displaying Zone Information
 - 13.3.3 Adding and Removing Zone Services
 - 13.3.4 Working with Port-based Rules
 - 13.3.5 Creating a New Zone
 - 13.3.6 Changing Zone/Interface Assignments
 - 13.3.7 Masquerading
 - 13.3.8 Adding ICMP Rules
 - 13.3.9 Implementing Port Forwarding
- 13.4 Managing firewalld from the Cockpit Interface
- 13.5 Managing firewalld using firewall-config
- 13.6 Summary

14. Configuring SSH Key-based Authentication on Fedora 31

- 14.1 An Overview of Secure Shell (SSH)
- 14.2 SSH Key-based Authentication
- 14.3 Setting Up Key-based Authentication
- 14.4 Installing and Starting the SSH Service
- 14.5 SSH Key-based Authentication from Linux and macOS Clients
- 14.6 Managing Multiple Keys
- 14.7 SSH Key-based Authentication from Windows 10 Clients
- 14.8 SSH Key-based Authentication using PuTTY
- 14.9 Generating a Private Key with PuTTYgen
- 14.10 Installing the Public Key for a Google Cloud Instance
- 14.11 Summary

15. Fedora 31 Remote Desktop Access with VNC

- 15.1 Secure and Insecure Remote Desktop Access
- 15.2 Installing the GNOME Desktop Environment
- 15.3 Installing VNC on Fedora 31
- 15.4 Configuring the VNC Server
- 15.5 Connecting to a VNC Server
- 15.6 Establishing a Secure Remote Desktop Session
- 15.7 Establishing a Secure Tunnel on Windows using PuTTY
- 15.8 Shutting Down a Desktop Session
- 15.9 Troubleshooting a VNC Connection

- 15.10 Summary
- 16. Displaying Fedora 31 Applications Remotely (X11 Forwarding)**
 - 16.1 Requirements for Remotely Displaying Fedora 31 Applications
 - 16.2 Remotely Displaying a Fedora 31 Application
 - 16.3 Trusted X11 Forwarding
 - 16.4 Compressed X11 Forwarding
 - 16.5 Displaying Remote Fedora 31 Apps on Windows
 - 16.6 Summary
- 17. Using NFS to Share Fedora 31 Files with Remote Systems**
 - 17.1 Ensuring NFS Services are running on Fedora 31
 - 17.2 Configuring the Fedora 31 Firewall to Allow NFS Traffic
 - 17.3 Specifying the Folders to be Shared
 - 17.4 Accessing Shared Fedora 31 Folders
 - 17.5 Mounting an NFS Filesystem on System Startup
 - 17.6 Unmounting an NFS Mount Point
 - 17.7 Accessing NFS Filesystems in Cockpit
 - 17.8 Summary
- 18. Sharing Files between Fedora 31 and Windows Systems with Samba**
 - 18.1 Samba and Samba Client
 - 18.2 Installing Samba on a Fedora 31 System
 - 18.3 Configuring the Fedora 31 Firewall to Enable Samba
 - 18.4 Configuring the *smb.conf* File
 - 18.4.1 Configuring the [global] Section
 - 18.4.2 Configuring a Shared Resource
 - 18.4.3 Removing Unnecessary Shares
 - 18.5 Configuring SELinux for Samba
 - 18.6 Creating a Samba User
 - 18.7 Testing the *smb.conf* File
 - 18.8 Starting the Samba and NetBIOS Name Services
 - 18.9 Accessing Samba Shares
 - 18.10 Accessing Windows Shares from Fedora 31
 - 18.11 Summary
- 19. An Overview of Virtualization Techniques**
 - 19.1 Guest Operating System Virtualization
 - 19.2 Hypervisor Virtualization
 - 19.2.1 Paravirtualization
 - 19.2.2 Full Virtualization
 - 19.2.3 Hardware Virtualization
 - 19.3 Virtual Machine Networking
 - 19.4 Summary

Table of Contents

20. Installing KVM Virtualization on Fedora 31

- 20.1 An Overview of KVM
- 20.2 KVM Hardware Requirements
- 20.3 Preparing Fedora 31 for KVM Virtualization
- 20.4 Verifying the KVM Installation
- 20.5 Summary

21. Creating KVM Virtual Machines using Cockpit and virt-manager

- 21.1 Installing the Cockpit Virtual Machines Module
- 21.2 Creating a Virtual Machine in Cockpit
- 21.3 Starting the Installation
- 21.4 Working with Storage Volumes and Storage Pools
- 21.5 Creating a Virtual Machine using virt-manager
- 21.6 Starting the Virtual Machine Manager
- 21.7 Configuring the KVM Virtual System
- 21.8 Starting the KVM Virtual Machine
- 21.9 Summary

22. Creating KVM Virtual Machines with virt-install and virsh

- 22.1 Running virt-install to build a KVM Guest System
- 22.2 An Example Fedora 31 virt-install Command
- 22.3 Starting and Stopping a Virtual Machine from the Command-Line
- 22.4 Creating a Virtual Machine from a Configuration File
- 22.5 Summary

23. Creating a Fedora 31 KVM Networked Bridge Interface

- 23.1 Getting the Current Network Settings
- 23.2 Creating a Network Bridge from the Command-Line
- 23.3 Declaring the KVM Bridged Network
- 23.4 Using a Bridge Network in a Virtual Machine
- 23.5 Creating a Bridge Network using nm-connection-editor
- 23.6 Summary

24. Managing KVM using the virsh Command-Line Tool

- 24.1 The virsh Shell and Command-Line
- 24.2 Listing Guest System Status
- 24.3 Starting a Guest System
- 24.4 Shutting Down a Guest System
- 24.5 Suspending and Resuming a Guest System
- 24.6 Saving and Restoring Guest Systems
- 24.7 Rebooting a Guest System
- 24.8 Configuring the Memory Assigned to a Guest OS
- 24.9 Summary

25. An Introduction to Linux Containers

- 25.1 Linux Containers and Kernel Sharing
- 25.2 Container Uses and Advantages
- 25.3 Fedora 31 Container Tools
- 25.4 The Docker Registry
- 25.5 Container Networking
- 25.6 Summary

26. Working with Containers on Fedora 31

- 26.1 Installing the Container Tools
- 26.2 Pulling a Container Image
- 26.3 Running the Image in a Container
- 26.4 Managing a Container
- 26.5 Saving a Container to an Image
- 26.6 Removing an Image from Local Storage
- 26.7 Removing Containers
- 26.8 Building a Container with Buildah
- 26.9 Building a Container from Scratch
- 26.10 Container Bridge Networking
- 26.11 Managing Containers in Cockpit
- 26.12 Summary

27. Setting Up a Fedora 31 Web Server

- 27.1 Requirements for Configuring a Fedora 31 Web Server
- 27.2 Installing the Apache Web Server Packages
- 27.3 Configuring the Firewall
- 27.4 Port Forwarding
- 27.5 Starting the Apache Web Server
- 27.6 Testing the Web Server
- 27.7 Configuring the Apache Web Server for Your Domain
- 27.8 The Basics of a Secure Web Site
- 27.9 Configuring Apache for HTTPS
- 27.10 Obtaining an SSL Certificate
- 27.11 Summary

28. Configuring a Fedora 31 Postfix Email Server

- 28.1 The structure of the Email System
 - 28.1.1 Mail User Agent
 - 28.1.2 Mail Transfer Agent
 - 28.1.3 Mail Delivery Agent
 - 28.1.4 SMTP
 - 28.1.5 SMTP Relay
- 28.2 Configuring a Fedora 31 Email Server

Table of Contents

- 28.3 Postfix Pre-Installation Steps
- 28.4 Firewall/Router Configuration
- 28.5 Installing Postfix on Fedora 31
- 28.6 Configuring Postfix
- 28.7 Configuring DNS MX Records
- 28.8 Starting Postfix on a Fedora 31 System
- 28.9 Testing Postfix
- 28.10 Sending Mail via an SMTP Relay Server
- 28.11 Summary

29. Adding a New Disk Drive to a Fedora 31 System

- 29.1 Mounted File Systems or Logical Volumes
- 29.2 Finding the New Hard Drive
- 29.3 Creating Linux Partitions
- 29.4 Creating a File System on a Fedora 31 Disk Partition
- 29.5 An Overview of Journaled File Systems
- 29.6 Mounting a File System
- 29.7 Configuring Fedora 31 to Automatically Mount a File System
- 29.8 Adding a Disk Using Cockpit
- 29.9 Summary

30. Adding a New Disk to a Fedora 31 Volume Group and Logical Volume

- 30.1 An Overview of Logical Volume Management (LVM)
 - 30.1.1 Volume Group (VG)
 - 30.1.2 Physical Volume (PV)
 - 30.1.3 Logical Volume (LV)
 - 30.1.4 Physical Extent (PE)
 - 30.1.5 Logical Extent (LE)
- 30.2 Getting Information about Logical Volumes
- 30.3 Adding Additional Space to a Volume Group from the Command-Line
- 30.4 Adding Additional Space to a Volume Group using Cockpit
- 30.5 Summary

31. Adding and Managing Fedora 31 Swap Space

- 31.1 What is Swap Space?
- 31.2 Recommended Swap Space for Fedora 31
- 31.3 Identifying Current Swap Space Usage
- 31.4 Adding a Swap File to a Fedora 31 System
- 31.5 Adding Swap as a Partition
- 31.6 Adding Space to a Fedora 31 LVM Swap Volume
- 31.7 Adding Swap Space to the Volume Group
- 31.8 Summary

Index

1. Introduction

Backed by Red Hat, Inc. and developed by the Fedora Project, the Fedora Linux distribution is an open source operating system intended to field test the latest Linux developments and technologies before they are adopted by the Red Hat Enterprise Linux distribution (typically shortened to RHEL and pronounced *rell*). This makes Fedora an exciting, cutting-edge operating system that is ideal for gaining experience with the latest Linux developments and features before they appear in enterprise and mission critical environments. Fedora 31 Essentials is designed to provide detailed information on the installation, use and administration of the Fedora 31 distribution. For beginners, the book covers topics such as operating system installation, the basics of the GNOME desktop environment, configuring email and web servers and installing packages and system updates using App Streams. Additional installation topics such as dual booting with Microsoft Windows are also covered, together with all important security topics such as configuring a firewall and user and group administration.

For the experienced user, topics such as remote desktop access, the Cockpit web interface, logical volume management (LVM), disk partitioning, swap management, KVM virtualization, Secure Shell (SSH), Linux Containers and file sharing using both Samba and NFS are covered in detail to provide a thorough overview of this enterprise class operating system.

1.1 Superuser Conventions

Fedora, in common with Linux in general, has two types of user account, one being a standard user account with restricted access to many of the administrative files and features of the operating system, and the other a superuser (*root*) account with elevated privileges. Typically, a user can gain root access either by logging in as the root user, or using the *su* - command and entering the root password. In the following example, a user is gaining root access via the *su* - command:

```
[neil@demo-server ~]$ su -  
Password:  
[root@demo-server ~]#
```

Note that the command prompt for a regular user ends with a \$ sign while the root user has a # character. When working with the command-line, this is a useful indication as to whether or not you are currently issuing commands as the root user.

If the *su* - command fails, the root account on the system has most likely been disabled for security reasons. In this case, the *sudo* command can be used instead as outlined below.

Alternatively, a single command requiring root privileges may be executed by a non-root user via the *sudo* command. Consider the following attempt to update the operating system with the latest patches and packages:

```
[neil@demo-server ~]$ sudo dnf update
```

Introduction

```
Not root, Subscription Management repositories not updated
Error: This command has to be run under the root user.
```

Optionally, user accounts may be configured so that they have access to root level privileges. Instead of using the *su* - command to first gain root access, user accounts with administration privileges are able to run otherwise restricted commands using *sudo*.

```
[neil@demo-server]$ sudo dnf update
```

We trust you have received the usual lecture from the local System Administrator. It usually boils down to these three things:

- #1) Respect the privacy of others.
- #2) Think before you type.
- #3) With great power comes great responsibility.

```
[sudo] password for neil:
Updating Subscription Management repositories.
```

```
.
```

To perform multiple commands without repeatedly using the *sudo* command, a command prompt with persistent super-user privileges may be accessed as follows:

```
[neil@demo-server]$ sudo su -
[neil@demo-server]#
```

The reason for raising this issue so early in the book is that many of the command-line examples outlined in this book will require root privileges. Rather than repetitively preface every command-line example with directions to run the command as root, the command prompt at the start of the line will be used to indicate whether or not the command needs to be performed as root. If the command can be run as a regular user, the command will be prefixed with a \$ command prompt as follows:

```
$ date
```

If, on the other hand, the command requires root privileges, the command will be preceded by a # command prompt:

```
# dnf install openssh
```

1.2 Feedback

We want you to be satisfied with your purchase of this book. If you find any errors in the book, or have any comments, questions or concerns please contact us at feedback@ebookfrenzy.com.

1.3 Errata

While we make every effort to ensure the accuracy of the content of this book, it is inevitable that a book covering a subject area of this size and complexity may include some errors and oversights. Any known issues with the book will be outlined, together with solutions, at the following URL:

<https://www.ebookfrenzy.com/errata/fedora31.html>

In the event that you find an error not listed in the errata, please let us know by emailing our support team at *feedback@ebookfrenzy.com*.

2. A Brief History of Linux

Fedora is one of a number of variants (also referred to as *distributions*) of the Linux operating system. It is community developed by the Fedora Project and sponsored by a U.S. company named Red Hat, Inc., based in Raleigh, North Carolina. The company was founded in the mid-1990s through the merger of two companies owned at the time by Marc Ewing and Bob Young. The origins of Linux, however, go back even further. This chapter will outline the history of both the Linux operating system and Red Hat, Inc. before explaining how Fedora fits into this picture.

2.1 What exactly is Linux?

Linux is an operating system in much the same way that Windows is an operating system (and there any similarities between Linux and Windows end). The term operating system is used to describe the software that acts as a layer between the hardware in a computer and the applications that we all run on a daily basis. When programmers write applications, they interface with the operating system to perform such tasks as writing files to the hard disk drive and displaying information on the screen. Without an operating system, every programmer would have to write code to directly access the hardware of the system. In addition, the programmer would have to be able to support every single piece of hardware ever created to be sure the application would work on every possible hardware configuration. Because the operating system handles all of this hardware complexity, application development becomes a much easier task. Linux is just one of a number of different operating systems available today.

2.2 UNIX Origins

To understand the history of Linux, we first have to go back to AT&T Bell Laboratories in the late 1960s. During this time AT&T had discontinued involvement in the development of a new operating system named Multics. Two AT&T engineers, Ken Thompson and Dennis Ritchie, decided to take what they had learned from the Multics project and create a new operating system named UNIX which quickly gained popularity and wide adoption both with corporations and academic institutions.

A variety of proprietary UNIX implementations eventually came to market including those created by IBM (AIX), Hewlett-Packard (HP-UX) and Sun Microsystems (SunOS and Solaris). In addition, a UNIX-like operating system named MINIX was created by Andrew S. Tanenbaum designed for educational use with source code access provided to universities.

2.3 Who Created Linux?

The origins of Linux can be traced back to the work and philosophies of two people. At the heart of the Linux operating system is something called the kernel. This is the core set of features necessary for the operating system to function. The kernel manages the system's resources and handles communication between the hardware and the applications. The Linux kernel was

A Brief History of Linux

developed by Linus Torvalds who, taking a dislike to MS-DOS, and impatient for the availability of MINIX for the new Intel 80386 microprocessor, decided to write his own UNIX-like kernel. When he had finished the first version of the kernel, he released it under an open source license that enabled anyone to download the source code and freely use and modify it without having to pay Linus any money.

Around the same time, Richard Stallman at the Free Software Foundation, a strong advocate of free and open source software, was working on an open source operating system of his own. Rather than focusing initially on the kernel, however, Stallman decided to begin by developing open source versions of all the UNIX tools, utilities and compilers necessary to use and maintain an operating system. By the time he had finished developing this infrastructure it seemed like the obvious solution was to combine his work with the kernel Linus had written to create a full operating system. This combination became known as GNU/Linux. Purists insist that Linux always be referred to as GNU/Linux (in fact, at one time, Richard Stallman refused to give press interviews to any publication which failed to refer to Linux as GNU/Linux). This is not unreasonable given that the GNU tools developed by the Free Software Foundation make up a significant and vital part of GNU/Linux. Unfortunately, most people and publications simply refer to Linux as Linux and this will probably always continue to be the case.

2.4 The Early Days of Red Hat

In 1993 Bob Young created a company named ACC Corporation which, according to Young, he ran from his “wife’s sewing closet”. The name ACC was intended to represent a catalog business but was also an abbreviation of a small business his wife ran called “Antiques and Collectibles of Connecticut”. Among the items sold through the ACC catalog business were Linux CDs and related open source software.

Around the same time, Marc Ewing had created his own Linux distribution company which he named Red Hat Linux (after his propensity to wear a red baseball cap while at Carnegie Mellon University).

In 1995, ACC acquired Red Hat, adopted the name Red Hat, Inc. and experienced rapid and significant growth. Bob Young stepped down as CEO shortly after the company went public in August of 1999 and has since pursued a number of business and philanthropic efforts including a print-on-demand book publishing company named Lulu and ownership of two Canadian professional sports teams. In 2018, IBM announced plans to acquire Red Hat, Inc. in a deal valued at \$34 billion.

2.5 Red Hat Support

Early releases of Red Hat Linux were shipped to customers on floppy disks and CDs (this, of course, predated the widespread availability of broadband internet connections). When users encountered problems with the software they were only able to contact Red Hat by email. In fact, Bob Young often jokes that this was effective in limiting support requests since, by the time a customer realized they needed help, their computer was usually inoperative and therefore unavailable to be used to send an email message seeking assistance from Red Hat’s support

team. In later years Red Hat provided better levels of support tied to paid subscriptions and now provides a variety of support levels ranging from “self help” (no support) up to premium support.

2.6 Open Source

Red Hat Enterprise Linux 8 is the current commercial offering from Red Hat and is primarily targeted at corporate, mission critical installations. It is also the cornerstone of an expanding ecosystem of products and services offered by Red Hat. RHEL is an open source product in that you can download the source code free of charge and build the software yourself if you wish to do so (a task not to be undertaken lightly). If, however, you wish to download a pre-built, ready to install binary version of the software (either with or without support), you have to pay for it.

2.7 The Fedora Project

Red Hat also sponsors the Fedora Project, the goal of which is to provide access to a free Linux operating system (in both source and binary distributions) in the form of Fedora Linux. Fedora Linux also serves as a proving ground for many of the new features that are eventually adopted into the Red Hat Enterprise Linux operating system family. Red Hat Enterprise Linux 8.0, for example, was based to a large extent on Fedora 29.

2.8 CentOS - The Free Alternative

For users unable to afford a Red Hat Enterprise Linux subscription, another option is provided in the form of the CentOS operating system. The CentOS project, originally a community driven effort but now a collaboration with Red Hat, takes the Red Hat Enterprise Linux source code, removes the Red Hat branding and subscription requirements, compiles it and provides the distribution for download. Of course, while CentOS provides an operating system that is identical to RHEL in many ways, it does not include access to Red Hat technical support.

2.9 Summary

The origins of the Linux operating system can be traced back to the work of Linus Torvalds and Richard Stallman in the form of the Linux kernel combined with the tools and compilers built by the GNU project.

Over the years, the open source nature of Linux has resulted in the release of a wide range of different Linux distributions. One such distribution is Red Hat Enterprise Linux, created by Red Hat, Inc., a company founded by Bob Young and Mark Ewing. Red Hat specializes in providing enterprise level Linux software solutions combined with extensive technical support services. Fedora is developed by the Fedora Project in close collaboration with Red Hat, Inc. and serves as a proving ground for the new features that will eventually be included in future Red Hat Enterprise Linux releases.

3. Installing Fedora 31 on a Clean Disk Drive

There are now two ways in which a Fedora 31 system can be deployed. One method is to either purchase new hardware or re-purpose an existing computer system on which to install and run the operating system. Another option is to create a cloud-based operating system instance using services such as Amazon AWS, Google Cloud or Microsoft Azure (to name but a few). Since cloud-based instances are typically created by selecting a pre-configured, ready to run operating system image that is already optimized for the cloud platform, and using that as the basis for the Fedora system, there is no need to perform a manual operating system installation in this situation.

If, on the other hand, you plan to install Fedora 31 on your own hardware, the first step on the path to learning about Fedora involves installing the operating system.

Fedora can be installed either in a clean disk environment (where an entire disk is cleared of any existing partitions and dedicated entirely to Fedora) or in a dual boot environment where Fedora co-exists with another operating system on the disk (typically a member of the Microsoft Windows family of operating systems).

In this chapter we will be covering the clean disk approach to installation from local or remote installation media. Dual boot installation with a Windows 10 system will be covered in “*Dual Booting Fedora 31 with Windows*”.

3.1 Fedora Installation Options

Fedora can be downloaded free of charge from the following web page:

<https://getfedora.org/>

This page provides a number of download options depending on how the operating system is to be installed and used:

- **Fedora Workstation** - Downloads the installation media for the workstation edition of the operating system. This edition is intended for use on desktop and laptop systems where a graphical desktop environment is needed and is only available for 64-bit x86 systems. The workstation edition can be downloaded in the form of an ISO image which you can then write to a USB drive using the steps outlined later in this chapter. If you are performing the download on a Windows or macOS system, however, the easiest option is to download the Fedora Media Writer tool. This can be used to download the required Fedora edition and write it to a USB drive. Regardless of the download option chosen, the workstation media will allow you to test out Fedora by running a Live Fedora session prior to performing the installation.

Installing Fedora 31 on a Clean Disk Drive

- **Fedora Server** - Downloads the installation media for the server edition of the operating system. This image is intended for performing an installation on servers on which the graphical desktop environment is not required and is available for both x86 and ARM aarch64 64-bit systems. Although the server option only offers links to download the ISO images, the Fedora Media Writer available from the workstation page may also be used to download server images and write them to a USB drive.

When downloading the server edition, the following options are available:

- **Standard ISO Image** - Contains everything to install Fedora Server edition. This allows the installation to be performed without needing a network or internet connection.
- **Netinstall ISO Image** - Contains the minimum needed to begin the installation process during which additional packages are downloaded based on choices made during the configuration phase.

3.2 Choosing an Installation Option

Clearly a decision between the Workstation and the two Server Edition images needs to be made before installation can begin. If you would like to try Fedora 31 before installing it, then the Fedora Workstation option is the best solution since it allows you to boot Fedora from the installation media without first installing it on a disk drive. As shown in Figure 3-1, this option also provides the option to initiate the installation from within the live session:

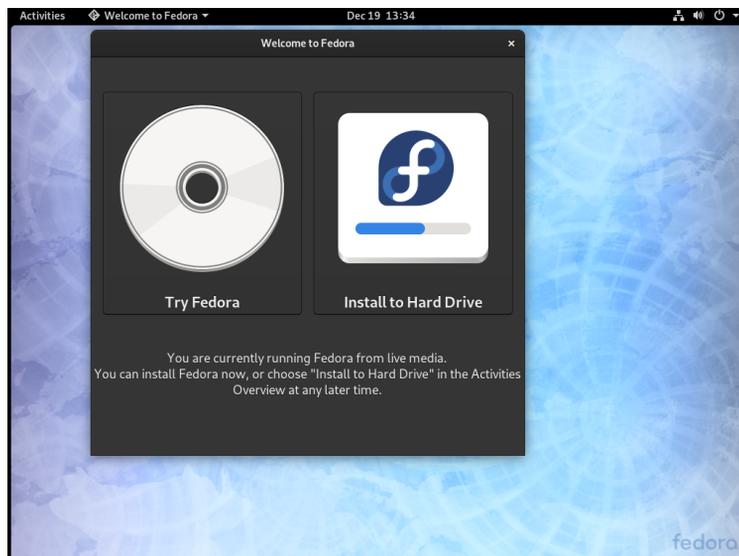


Figure 3-1

If the graphical desktop environment is not required, and the destination system does not have internet access then the Server Edition Standard ISO image is recommended since this allows a fully functional server to be built without the need to download any additional packages.

For installations where the system has internet access and a trial run of Fedora 31 is not required prior to installation, the Server Edition Netinstall ISO image provides the greatest flexibility. This image is much smaller than the other options and allows selections to be made from a wide range of pre-defined configurations (including both Server and Workstation options as well as a Minimal Install configuration).

The examples in this chapter will be based on installing Fedora 31 from the Fedora Server Netinstall image, selecting the option during the installation process to install the full Fedora Workstation edition. If you prefer to install from the Fedora Workstation live session or the Standard Server Edition image, the steps in this chapter are still relevant, though only a subset of the configuration options will be available.

3.3 Obtaining the Fedora 31 Installation Media

For the purposes of this book, download the Fedora 31 Netinstall ISO image from the following URL:

```
https://download.fedoraproject.org/pub/fedora/linux/releases/31/Server/x86\_64/iso/Fedora-Server-netinst-x86\_64-31-1.9.iso
```

The DVD ISO image is self-contained including all of the packages necessary to install a Fedora 31 system and is named using the following convention:

```
Fedora-Server-netinst-<architecture>-<version>.iso
```

For example, the Fedora 31 DVD image for 64-bit Intel systems is named as follows:

```
Fedora-Server-netinst-x86_64-31-1.9.iso
```

Having downloaded the image, either burn it to disk or use the steps in the next section to write the media to a USB drive, configure your virtualization environment to treat it as a DVD drive or use the steps outlined later in this chapter to access the installation image over a network connection.

3.4 Writing the ISO Installation Image to a USB Drive

These days it is more likely that an operating system installation will be performed from a USB drive than from a DVD. Having downloaded the ISO installation image for Fedora 31, the steps to write that image to a USB drive will differ depending on whether the drive is attached to a Linux, macOS or Windows system. The steps outlined in the remainder of this section assume that the USB drive is new, or has been reformatted to remove any existing data or partitions:

3.4.1 Linux

The first step in writing an ISO image to a USB drive on Linux is to identify the device name. Before inserting the USB drive, identify the storage devices already detected on the system by listing the devices in `/dev` as follows:

```
# ls /dev/sd*  
/dev/sda /dev/sda1 /dev/sda2
```

Attach the USB drive to the Linux system and run the `dmesg` command to get a list of recent

Installing Fedora 31 on a Clean Disk Drive

system messages, one of which will be a report that the USB drive was detected and will be similar to the following:

```
[445597.988045] sd 6:0:0:0: [sdb] Attached SCSI removable disk
```

This output tells us that we should expect the device name to include “sdb” which we can confirm by listing device names in */dev* again:

```
# ls /dev/sd*
/dev/sda   /dev/sda1  /dev/sda2  /dev/sdb
```

From this output we can tell that the USB drive has been assigned to */dev/sdb*. The next step before writing the ISO image to the device is to run the *findmnt* command to make sure it has not been auto-mounted:

```
# findmnt /dev/sdb
TARGET                                SOURCE      FSTYPE OPTIONS
/run/media/neil/d6bf9574-7e31-4f54-88b1 /dev/sdb   ext3    rw,nosuid,no
```

If the *findmnt* command indicates that the USB drive has been mounted, unmount it before continuing:

```
# umount /run/media/neil/d6bf9574-7e31-4f54-88b1
```

Once the filesystem has been unmounted, use the *dd* command as follows to write the ISO image to the drive:

```
# dd if=/path/to/iso/<image name>.iso of=/dev/sdb bs=512k
```

The writing process can take some time (as long as 10 - 15 minutes) to complete depending on the image size and speed of the system on which it is running. Once the image has been written, output similar to the following will appear and the USB drive is ready to be used to install Fedora 31:

```
5956+0 records in
5956+0 records out
3122659328 bytes (3.1 GB, 2.9 GiB) copied, 426.234 s, 7.3 MB/s
```

3.4.2 macOS

The first step in writing an ISO image to a USB drive attached to a macOS system is to identify the device using the *diskutil* tool. Before attaching the USB device, open a Terminal window and run the following command:

```
$ diskutil list
/dev/disk0 (internal, physical):
#:                                TYPE NAME                                SIZE      IDENTIFIER
0:      GUID_partition_scheme            *1.0 TB   disk0
1:      EFI EFI                          209.7 MB  disk0s1
2:      Apple_APFS Container disk2        1000.0 GB disk0s2

/dev/disk1 (internal):
#:                                TYPE NAME                                SIZE      IDENTIFIER
```

Installing Fedora 31 on a Clean Disk Drive

```
0:      GUID_partition_scheme                28.0 GB   disk1
1:      EFI EFI                             314.6 MB  disk1s1
2:      Apple_APFS Container disk2          27.7 GB   disk1s2
```

/dev/disk2 (synthesized):

```
#:      TYPE NAME                            SIZE      IDENTIFIER
0:      APFS Container Scheme -              +1.0 TB   disk2
        Physical Stores disk1s2, disk0s2
1:      APFS Volume Macintosh HD            473.6 GB  disk2s1
2:      APFS Volume Preboot                 42.1 MB   disk2s2
3:      APFS Volume Recovery                517.0 MB  disk2s3
4:      APFS Volume VM                      1.1 GB    disk2s4
```

Having established a baseline of detected devices, insert the USB drive into a port on the macOS system and run the command again. The same results should appear with one additional entry for the USB drive resembling the following:

/dev/disk3 (external, physical):

```
#:      TYPE NAME                            SIZE      IDENTIFIER
0:      *16.0 GB                             disk3
```

In the above example, the USB drive has been assigned to /dev/disk3. Before proceeding, unmount the disk as follows:

```
$ diskutil unmountDisk /dev/disk3
Unmount of all volumes on disk3 was successful
```

Finally, use the `dd` command to write the ISO image to the device, taking care to reference the raw disk device (`/dev/rdisk3`) and entering your user password when prompted:

```
$ sudo dd if=/path/to/iso/image.iso of=/dev/rdisk3 bs=1m
```

Once the image has been written, the USB drive is ready.

3.4.3 Windows

A number of free tools are available for Windows that will write an ISO image to a USB drive, but one written specifically for writing Linux ISO images is the Fedora Media Writer tool which can be downloaded from the following URL:

<https://getfedora.org/en/workstation/download/>

Once installed, launch the writer tool and select the *Custom image* option as highlighted in Figure 3-2. Alternatively, if you have chosen to use the Fedora Workstation media to try Fedora before installing, simply click in the Fedora Workstation 31 option to download and write the image to the USB drive.

Installing Fedora 31 on a Clean Disk Drive

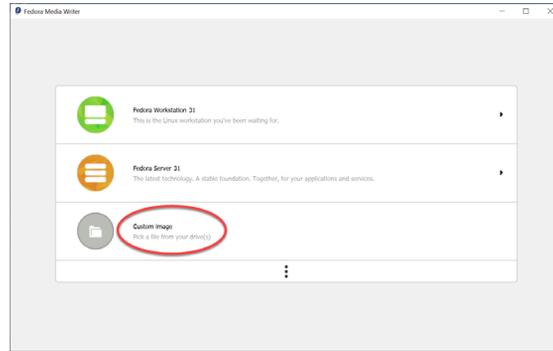


Figure 3-2

In the resulting file selection dialog, navigate to and select the Fedora 31 installation ISO image and click on the *Open* button. After selecting the image, a dialog will appear within which the image can be written to the USB drive. Select the target USB drive from the device menu before clicking on the *Write to Disk* button:

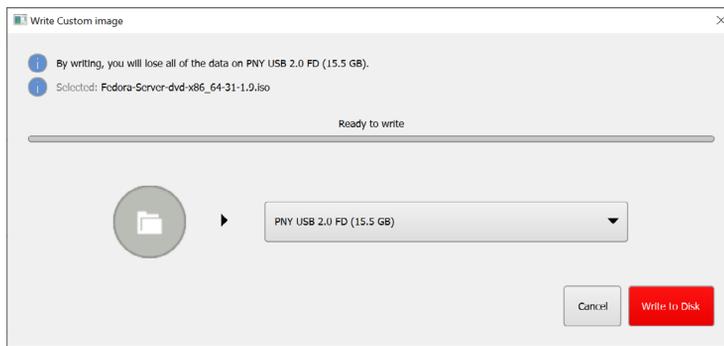


Figure 3-3

Once the image has been written to the device, the device is ready to be used to perform the installation.

3.5 Installing Fedora 31

Insert the Fedora 31 installation media into the appropriate drive and power on the system. If the system tries to boot from the hard disk drive you will need to enter the BIOS set up for your computer and change the boot order so that it boots from the installation media drive first. Once the system has booted you will be presented with the following screen:

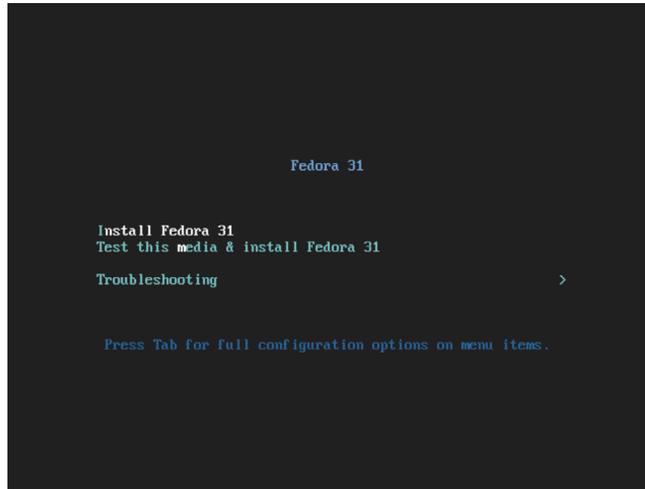


Figure 3-4

Use the arrow keys to navigate between the options and make a selection with the <Enter> key. If the *Troubleshooting* option is selected the screen shown in Figure 3-5 will appear including options to boot from the current operating system on the local drive (if one is installed), test the system memory, or rescue an installed Fedora 31 system. An option is also available to perform the installation in basic graphics mode for systems without a graphical console:

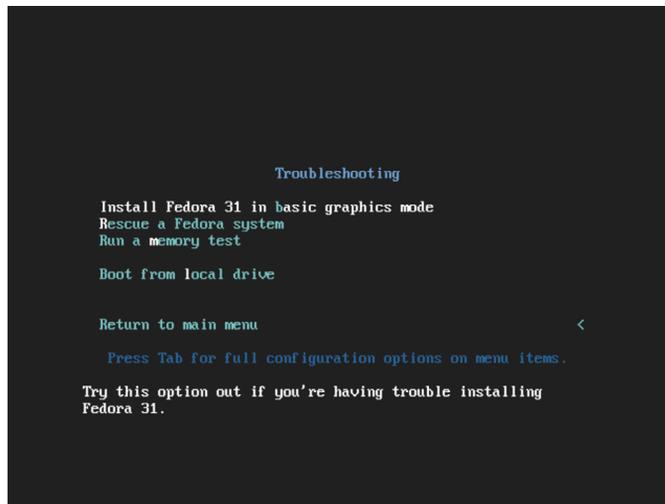


Figure 3-5

Select the option on the main screen to install and, after a short delay, the first screen of the graphical installer will appear:

Installing Fedora 31 on a Clean Disk Drive

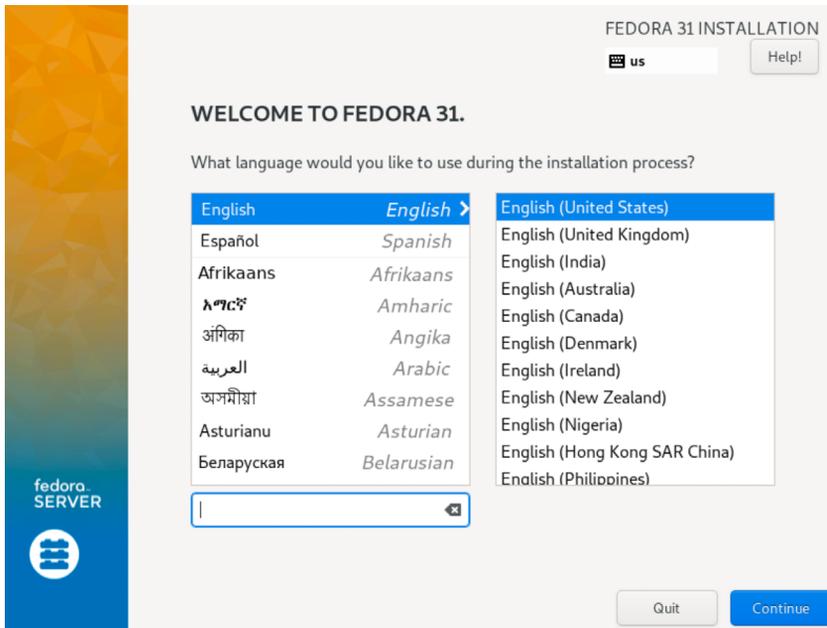


Figure 3-6

Select your preferred language on the first screen before clicking *Continue* to proceed to the main installation screen as shown in Figure 3-7:

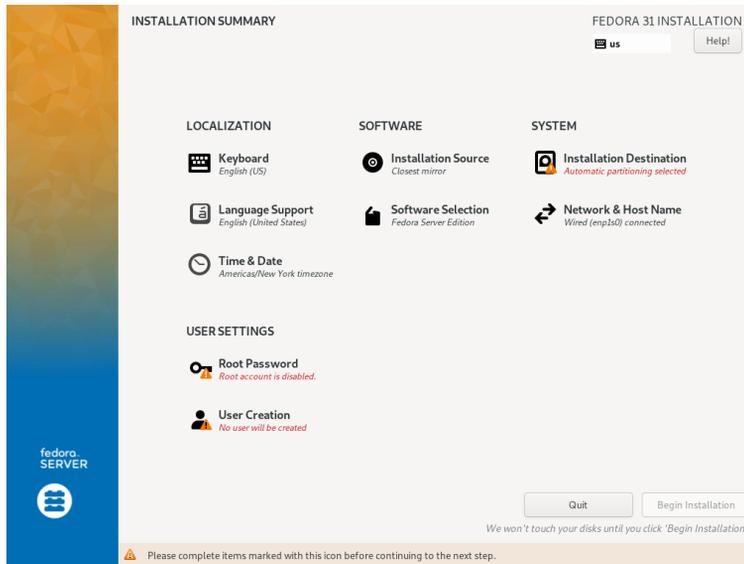


Figure 3-7

Begin by selecting the *Network & Host Name* option, enable a network device on your system and enter a host name before clicking the *Apply* button:

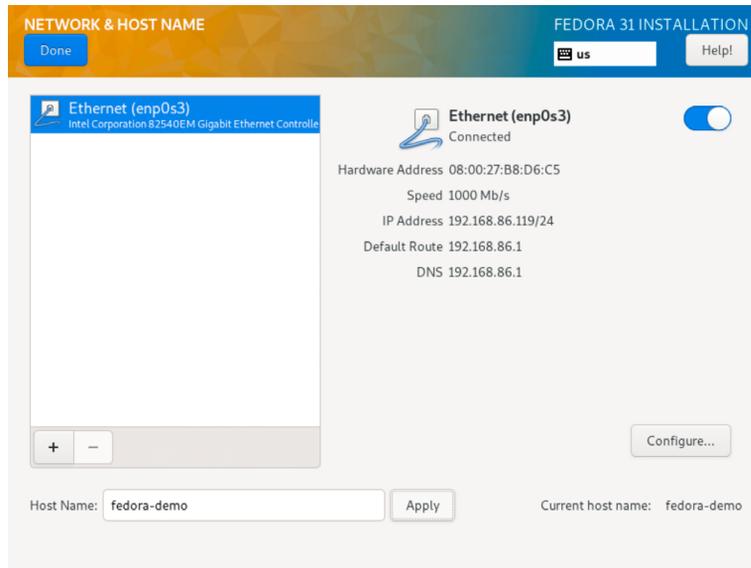


Figure 3-8

If your network connection requires additional settings, click on the *Configure...* button to access the advanced network settings screen illustrated in Figure 3-9:

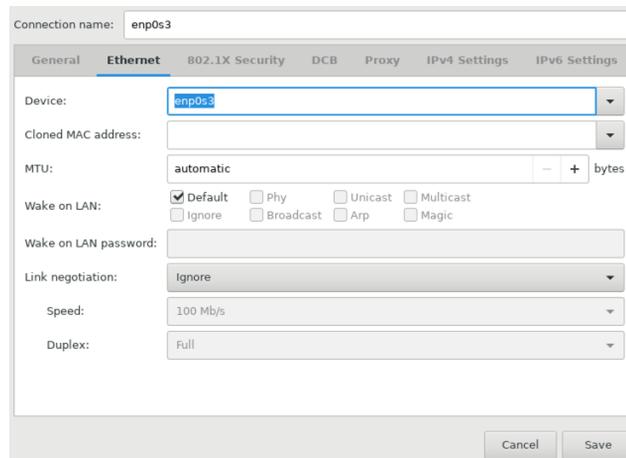


Figure 3-9

Once the host name has been defined and a network connection enabled, click *Done* to return to the main screen.

If you wish to change the keyboard, language or time and date settings, select the corresponding option from the Localization column of the main screen. On the *Time & Date* screen, make a selection corresponding to your geographical location. The option is also provided to use *Network Time* which automatically synchronizes the system with external Network Time Protocol servers.

Installing Fedora 31 on a Clean Disk Drive

Changes to the *Installation Source* settings should not be necessary since the installation is being performed from local media. To perform the installation from media located on a remote server, select *Installation Source*, enable the *On the network* option and then specify the location of the installation media and the type of URL being used. The installation media can, for example, be an ISO image installed either on a remote web server or on a system on the local network using NFS (the topic of NFS file sharing is covered in detail in the chapter entitled “*Using NFS to Share Fedora 31 Files with Remote Systems*”), or the URL of a remote repository (repositories are essentially online collections of the software, packages and applications that make up the operating system from which installations onto the local system can be performed).

By default, a Server Edition installation of Fedora 31 will be performed by the installer. To select a different installation configuration, choose the *Software Selection* option to display the screen shown below:

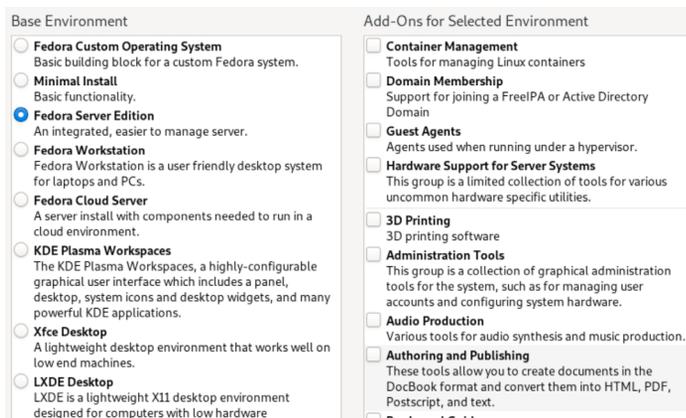


Figure 3-10

Use the left-hand panel to select a basic configuration and the right-hand panel to add any additional packages you know you will need after the system starts up. If the Fedora 31 system is intended to be used with a graphical desktop environment, choose the *Fedora Workstation* option. If the desktop environment is not required and you are unsure which packages to install on the system, use the *Minimal install* option and install additional packages as needed once the system has booted. This avoids installing any packages that may never be needed.

Having configured the basic settings, the next step is to decide how the hard disk drive will be partitioned to accommodate the operating system.

3.6 Partitioning a Disk for Fedora 31

When the *Installation Destination* option is selected, the installer will present a screen similar to the one illustrated in Figure 3-11 below:

Installing Fedora 31 on a Clean Disk Drive

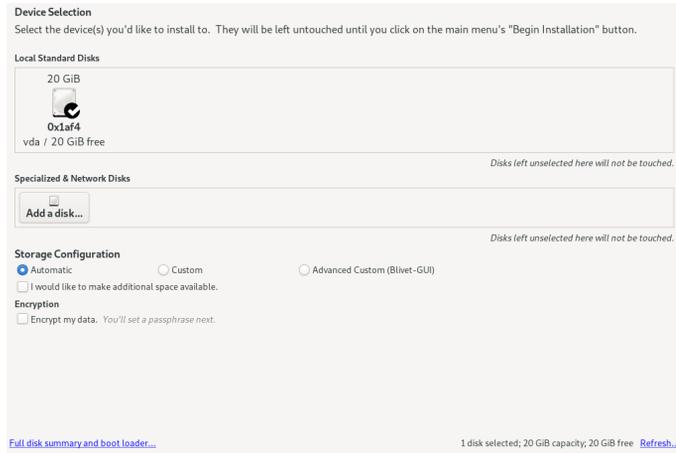


Figure 3-11

By default, the installer is configured to automatically install the operating system using the available space on the currently selected disk drive and to configure standard partition sizes. If the disk previously contained another operating system, these existing partitions will not be deleted, potentially leaving unused space on the drive after Fedora 31 has been installed. To remove any existing partitions so that they can be reclaimed and used by Fedora 31, enable the *I would like to make additional space available* option and click on the *Done* button to display the dialog shown in Figure 3-12.

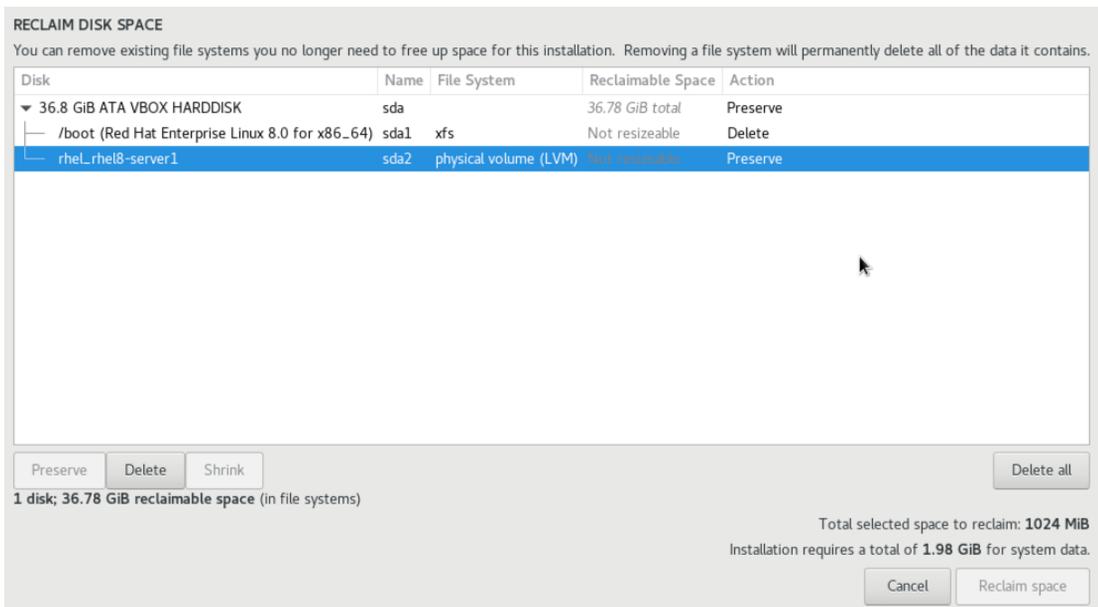


Figure 3-12

To reclaim space, select a partition that is no longer needed and mark it for removal by clicking

Installing Fedora 31 on a Clean Disk Drive

the *Delete* button. Once all the partitions that are to be removed have been selected, click on the *Reclaim Space* button to perform the deletion. The reclaimed space will now be used automatically by the Fedora 31 installer.

To manually configure the layout of the disk in terms of how the available space will be allocated, change the *Storage Allocation* setting from *Automatic* to *Custom* and click *Done* to display the Manual Partitioning screen (Figure 3-13):

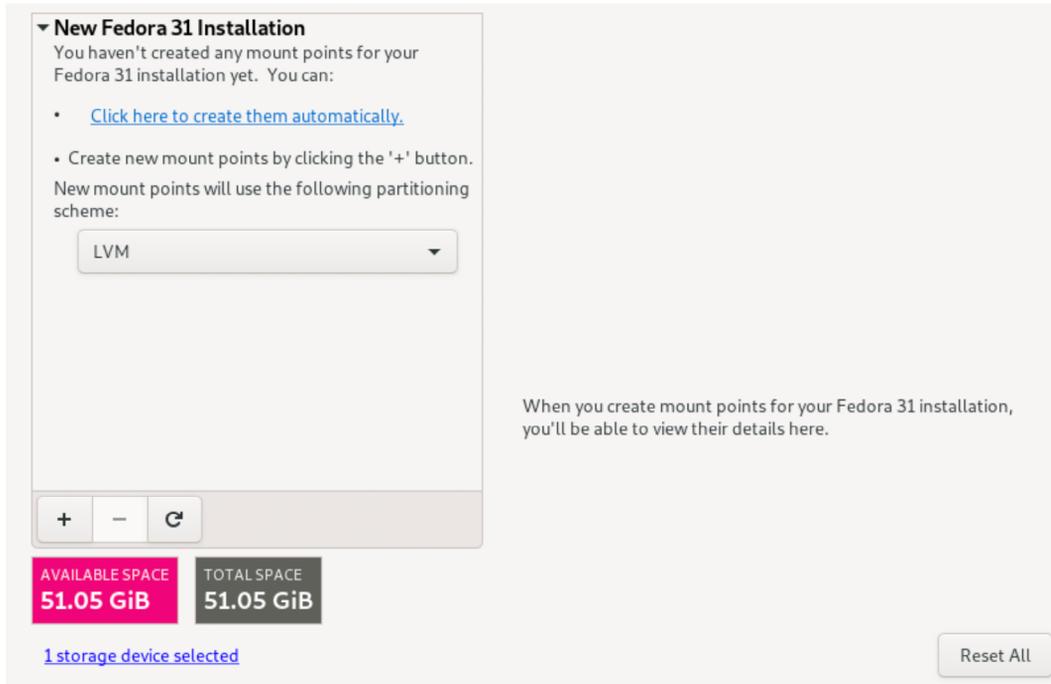


Figure 3-13

The manual partitioning screen provides the option to configure the disk using Logical Volume Management (LVM) or standard partitioning. LVM is the recommended option because it provides flexibility in terms of managing partition sizes once the system is up and running (LVM is covered in detail in the chapter entitled “*Adding a New Disk to a Fedora 31 Volume Group and Logical Volume*”).

Once a partitioning scheme has been selected, the last step is to decide on the sizes of the partitions and the corresponding filesystem mount points. In general, the default configuration provided by the *Click here to create them automatically* option will meet most system needs. To manually create partitions and allocate mount points, click on the + button to declare and assign each partition manually.

Another option is to select automatic partition creation, and then use the resulting screen to manually change the partition configuration as needed. Figure 3-14, for example, shows the partition configuration defined by selecting the automatic creation option and allows the settings

for each partition to be modified before any changes are made to the disk:

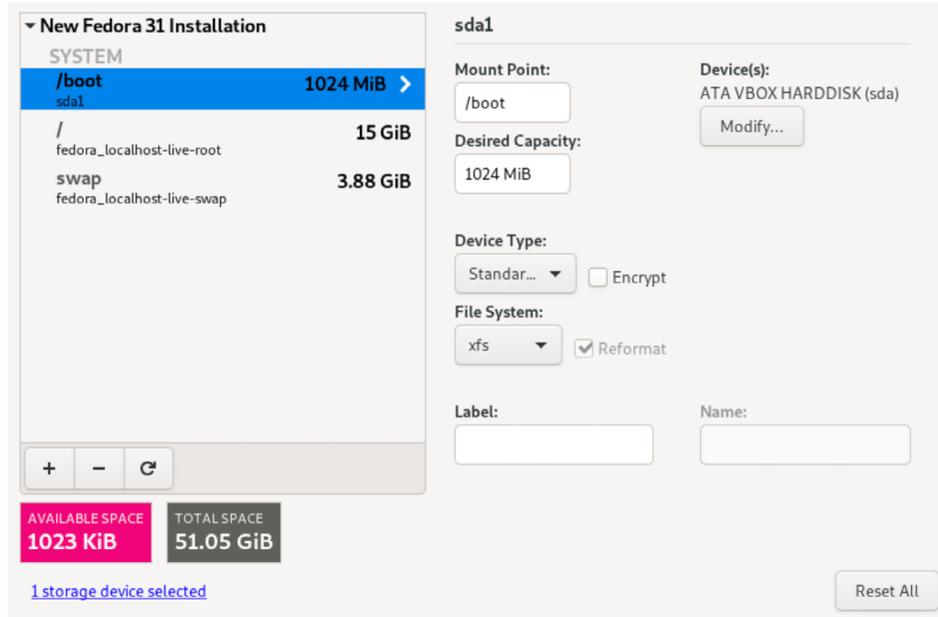


Figure 3-14

Once the disk has been configured, click on *Done*, review the summary of the changes to be made to the disk (Figure 3-15) followed by the *Accept Changes* button:

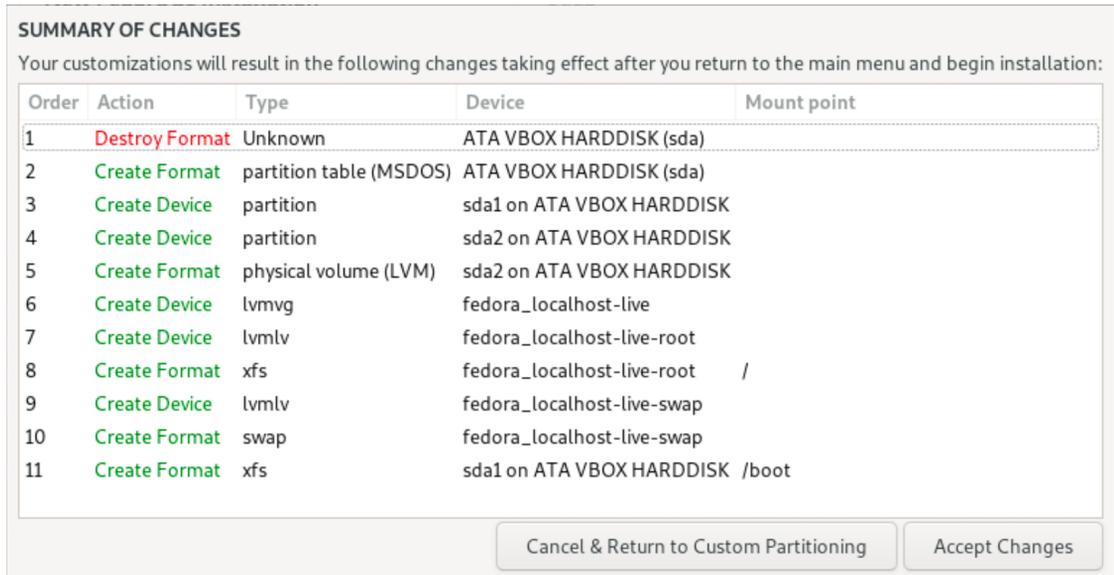


Figure 3-15

Installing Fedora 31 on a Clean Disk Drive

3.7 Adding a User and Root Access

The final step before beginning the installation is to add a user account to the system. To achieve this, click on the User Creation option in the main installation screen and enter user credentials and a password:

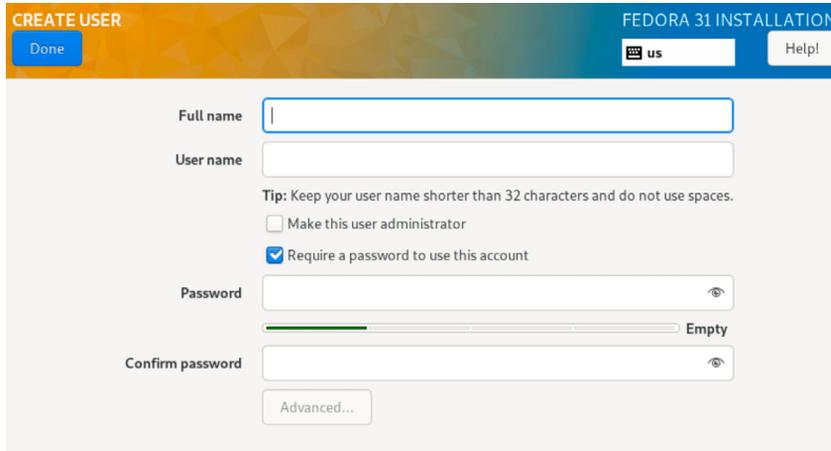


Figure 3-16

If the user is required to perform administrative tasks on the system, enable the *Make this user administrator* option before clicking on the Done button (at least one user should have administrative permissions unless the root account is enabled as outlined below). This will allow the account to perform restricted tasks by making use of the `sudo` command as outlined in the chapter entitled “*Introduction*”.

The root, or super-user account, is a special user that has administrative privileges on the system without the need to use the `sudo` command. To increase security, this account is disabled by default. To enable this account, select the Root Password option, enter a root password and turn off the *Lock root account* option. If you wish to be able to log into the system remotely as the root user using SSH (a topic covered in “*Configuring SSH Key-based Authentication on Fedora 31*”) enable the *Allow root SSH login with password* option. Unless you know that you will need root access, however, it is generally recommended to leave the account disabled.

3.8 The Physical Installation

Having made the appropriate package selections, clicking *Begin Installation* will start the process of partitioning the disks and installing the packages that match the chosen installation settings. During this phase, a screen will appear (Figure 3-17):

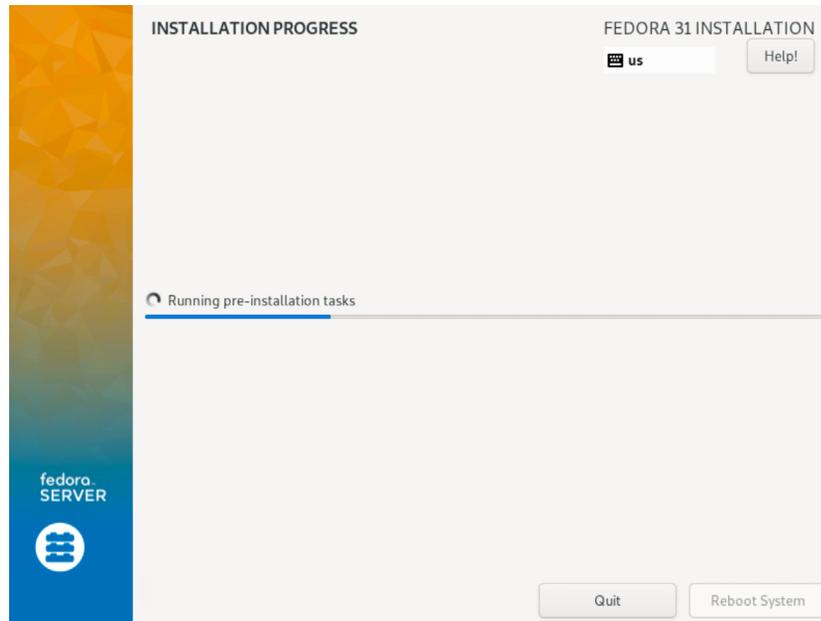


Figure 3-17

Once all the system packages have been installed and configured, remove the installation media and click *Reboot* to restart the system.

3.9 Final Configuration Steps

What appears on the console when the system has started up will depend on whether the *Workstation* option was selected from the Software Selection installation screen. If this was selected during installation, the GNOME Display Manager (GDM) login screen will appear. If the minimal or server configuration options were selected, the text based login prompt will be displayed. Regardless of the configuration, log into the system as the user that was created during the final steps of the installation process.

In the case of a *Workstation* installation, the GNOME initial setup tool will launch providing configuration options such as language, keyboard layout, location services and connections to online accounts including Google and Facebook. After these settings have been configured (or skipped), the GNOME desktop welcome screen will appear containing links to tutorials on how to use the desktop environment.

3.10 Installing Updates

As with most operating systems today, each particular release of a Fedora distribution continues to evolve after it has been released to the public. This generally takes the form of bug fixes and security updates and, occasionally, new features that may be downloaded over the internet and installed on your system.

Best practices dictate that the first step after installing Fedora is to make sure any available updates

Installing Fedora 31 on a Clean Disk Drive

are applied to the system. This can be achieved via the command-line prompt in a Terminal window using the *dnf* package manager tool. To check for the availability of updates, simply run the following command:

```
# dnf check-update
```

Any pending updates may be applied, once again using the *dnf* tool:

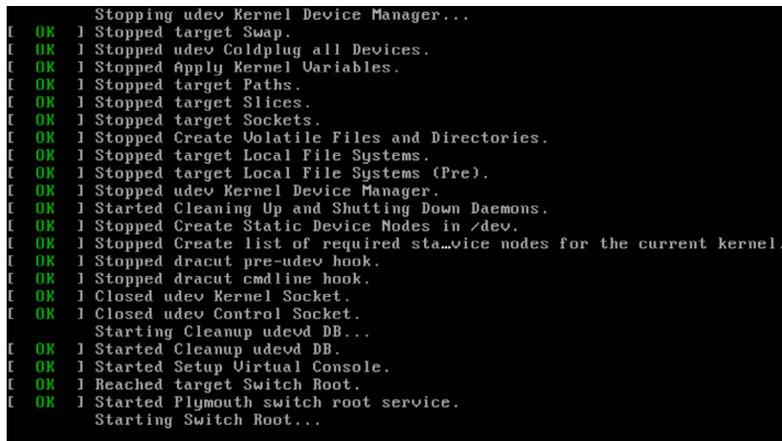
```
# dnf update
```

Upon execution, the *dnf* tool will provide a list of packages that are available for update and prompt for permission to perform the update.

Once the update is complete the installation is essentially finished and Fedora 31 is ready for use.

3.11 Displaying Boot Messages

During the boot process, Fedora will display the Red Hat Graphical Boot (RHGB) screen which hides from view all of the boot messages generated by the system as it loads. To make these messages visible during the boot process (as shown in Figure 3-18), simply press the keyboard Esc key while the system is starting:

A terminal window showing the output of the boot process. The text is as follows:

```
Stopping udev Kernel Device Manager...
[ OK ] Stopped target Swap.
[ OK ] Stopped udev Coldplug all Devices.
[ OK ] Stopped Apply Kernel Variables.
[ OK ] Stopped target Paths.
[ OK ] Stopped target Slices.
[ OK ] Stopped target Sockets.
[ OK ] Stopped Create Volatile Files and Directories.
[ OK ] Stopped target Local File Systems.
[ OK ] Stopped target Local File Systems (Pre).
[ OK ] Stopped udev Kernel Device Manager.
[ OK ] Started Cleaning Up and Shutting Down Daemons.
[ OK ] Stopped Create Static Device Nodes in /dev.
[ OK ] Stopped Create list of required static device nodes for the current kernel.
[ OK ] Stopped dracut pre-udev hook.
[ OK ] Stopped dracut cmdline hook.
[ OK ] Closed udev Kernel Socket.
[ OK ] Closed udev Control Socket.
Starting Cleanup udevd DB...
[ OK ] Started Cleanup udevd DB.
[ OK ] Started Setup Virtual Console.
[ OK ] Reached target Switch Root.
[ OK ] Started Plymouth switch root service.
Starting Switch Root...
```

Figure 3-18

The default behavior can be changed so that messages are always displayed by default by editing the */etc/default/grub* file and changing the `GRUB_CMDLINE_LINUX` setting which, by default, will resemble the following:

```
GRUB_CMDLINE_LINUX="... rhgb quiet"
```

To remove the graphical boot screen so that messages are visible without pressing the Esc key, remove the “rhgb” option from the setting:

```
GRUB_CMDLINE_LINUX="... quiet"
```

This change will cause the system to display only a subset of the boot messages generated by the system. To display all messages generated by the system, also remove the “quiet” option:

```
GRUB_CMDLINE_LINUX="..."
```

Once the changes have been made, run the following command to generate a new boot configuration to take effect next time the system starts:

```
# grub2-mkconfig --output=/boot/grub2/grub.cfg
```

3.12 Summary

The first step in working with Fedora 31 is to install the operating system. In the case of a cloud-based server, this task is typically performed automatically when an operating system image is selected for the system based on a range of options offered by the cloud service provider. Installation on your own hardware, however, involves downloading the installation media in the form of an ISO image, writing that image to suitable storage such as a DVD or USB drive and booting from it. Once running, the installation process allows a range of options to be configured ranging from networking, whether the installation should proceed from the local media or via a remote server or repository, the packages to be installed, and the partitioning of the disk. Once installation is complete, it is important to install any operating system updates that may have been released since the original installation image was created.

4. Dual Booting Fedora 31 with Windows

Fedora, just like most Linux distributions, will happily co-exist on a hard disk drive with just about any version of Windows up to and including Windows 10. This is a concept known as dual-booting. Essentially, when you power up the system, you will be presented with a menu providing the option to boot either your Fedora 31 installation or Windows. Obviously you can only run one operating system at a time, but it is worth noting that the files on the Windows partition of your disk drive will be available to you from Fedora 31 regardless of whether your Windows partition was formatted using NTFS, FAT16 or FAT32.

This installation method involves shrinking the size of the existing Windows partitions and then installing Fedora 31 into the reclaimed space. This chapter will assume that Fedora 31 is being installed on a system currently running Windows 10.

4.1 Partition Resizing

In order to accommodate Fedora on a disk drive that already contains a Windows installation, the first step involves shrinking the Windows partition to make some room. The recommended course of action is to use the Windows Disk Management interface to reduce the size of the partition before attempting to install Fedora 31.

To access Disk Management on Windows 10, right-click on the Start menu and select *Disk Management* from the resulting menu as highlighted in Figure 4-1:

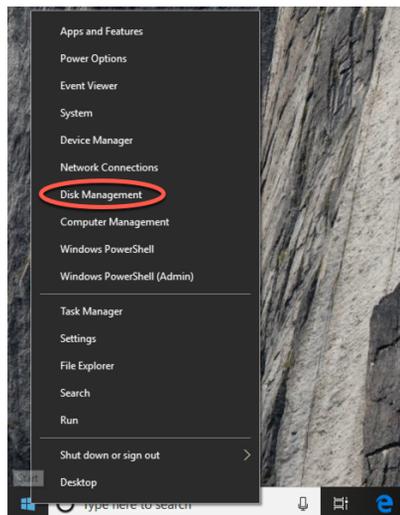


Figure 4-1

Dual Booting Fedora 31 with Windows

Once loaded, the Disk Management tool will display a graphical representation of the disk drives detected on the system:

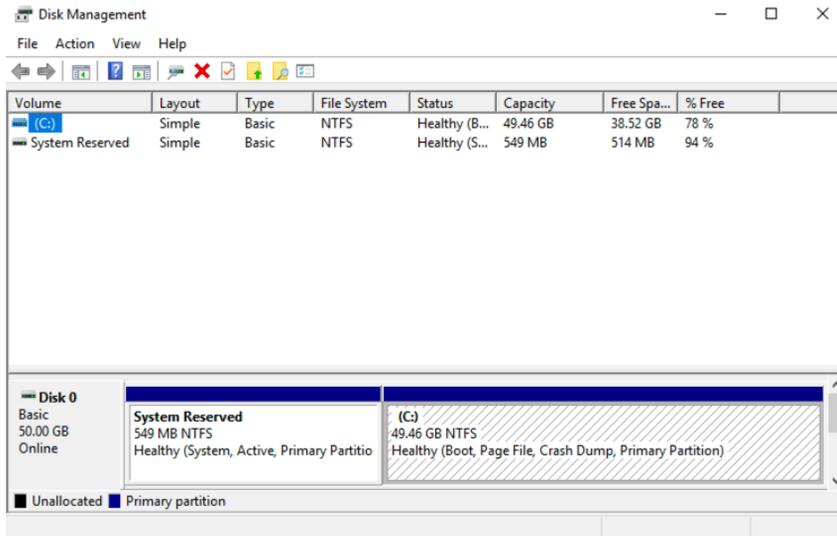


Figure 4-2

Right-click on the partition you wish to reduce in size and select *Shrink Volume...* from the popup menu. The tool will calculate the maximum amount by which the volume size can be reduced without data loss (a process that can take several minutes depending on the overall size of the partition). Once this analysis is complete, a dialog similar to the one in Figure 4-3 below will appear:

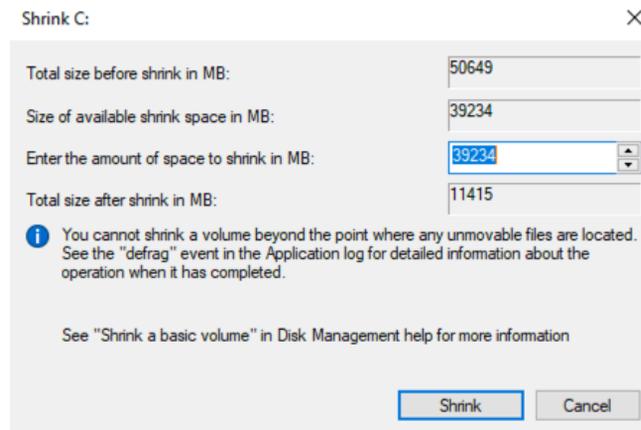


Figure 4-3

Specify a value in the *Enter amount of space to shrink in MB* field and click on the *Shrink* button to proceed. Once the resizing operation is complete, reboot using the Fedora 31 installation media (as outlined in *Installing Fedora 31 on a Clean Disk Drive*) and proceed with the installation

making use of the new free space. During the Fedora installation process, this can be achieved by selecting the *Installation Destination* option on the Installation Summary screen and making sure that the *Automatic* storage configuration option is selected. This will automatically install Fedora into the unallocated space that was created when the Windows partition was reduced in size.

Once the installation is complete, reboot the system and note that Windows is now an option on the boot screen in addition to Fedora:

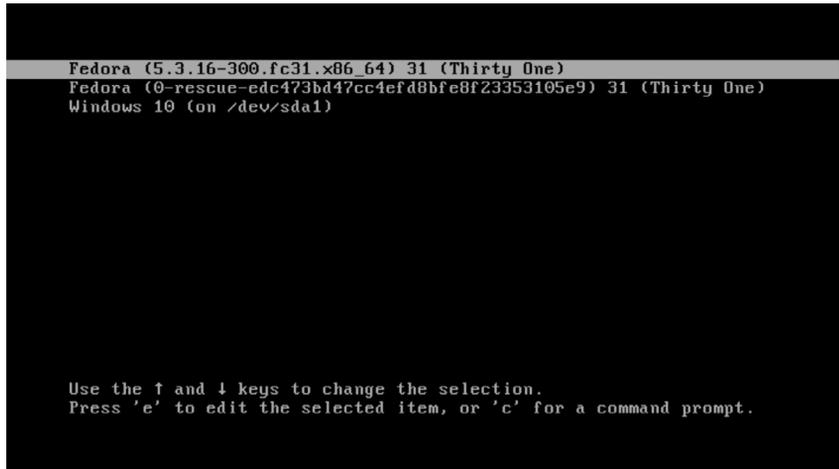


Figure 4-4

4.2 Changing the Default Boot Option

When the system starts, the boot options screen will appear and wait 5 seconds for the user to make an operating system choice. If no selection has been made before the timeout elapses, the default operating system will be started. On a newly configured system, the default operating system will be the standard (non-rescue) Fedora image. This default can, however, be changed from within Fedora.

A range of boot configuration options (including the 5 second timeout and the boot RHGB settings outlined in “*Installing Fedora 31 on a Clean Disk Drive*”) are declared in the `/etc/default/grub` file which reads as follows on a new installation:

```

GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_DISABLE_SUBMENU=true
GRUB_TERMINAL_OUTPUT="console"
GRUB_CMDLINE_LINUX="resume=/dev/mapper/fedora-swap rd.lvm.lv=fedora/root rd.lvm.lv=fedora/swap rhgb quiet"
GRUB_DISABLE_RECOVERY="true"
GRUB_ENABLE_BLSCFG=true

```

The first step in changing the default boot system is to declare the `GRUB_SAVEDDEFAULT` setting

Dual Booting Fedora 31 with Windows

within this file:

```
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR="$(sed 's, release .*$,,g' /etc/system-release)"
GRUB_DEFAULT=saved
GRUB_SAVEDEFAULT=true
.
.
```

This setting allows a new default value to be saved within the boot configuration. Next, run the *grub2-set-default* command to change the default setting using a numbering system that counts the first option as 0. For example, if the Windows 10 option is position 3 in the menu, the command to make Windows 10 the default boot option would read as follows:

```
# grub2-set-default 2
```

Check that the new setting has taken effect by running the following command:

```
# grub2-editenv list
saved_entry=2
kernelopts=root=/dev/mapper/fedora-root ro resume=/dev/mapper/fedora-swap rd.lvm.
lv=fedora/root rd.lvm.lv=fedora/swap rhgb quiet
boot_success=0
```

Note that the *saved_entry* value is now set to 2. After changing the default, regenerate the boot configuration file as follows:

```
# grub2-mkconfig --output=/boot/grub2/grub.cfg
```

Reboot the system and verify that the boot menu defaults to the Windows 10 option and that Windows loads after the timeout expires.

4.3 Accessing the Windows Partition from Fedora 31

When running Fedora in a dual boot configuration it is still possible to access files located on the Windows partition. This can be achieved by manually mounting the partition from the command-line. Before doing so, however, some additional packages need to be installed on the system. First the *fuse* kernel module needs to be downloaded and installed:

```
# dnf install fuse
# modprobe fuse
```

Once the requisite packages are installed, the next step is to create a directory to use as the mount point for our Windows partition. In this example we will create a directory named */mnt/windows*:

```
# mkdir /mnt/windows
```

In order to identify the device name that has been assigned to the Windows partition, use the *fdisk* command as follows:

```
# fdisk -l
.
.
Device Boot          Start      End      Blocks      Id      System
```

```

/dev/sda1 *          1          13          102400      7  HPFS/NTFS
/dev/sda2          13          1400       11141120   7  HPFS/NTFS
/dev/sda3          1400         1464        512000    83  Linux
/dev/sda4          1464         2611        9214976   5  Extended
/dev/sda5          1464         2611        9213952   8e  Linux LVM

```

In the above output, the main Windows partition containing the files we need access to is represented by `/dev/sda2`. Next, we need to run the `mount` command (assuming the Windows partition is `/dev/sda2` and NTFS format) as follows:

```
# mount /dev/sda2 /mnt/windows
```

Check that the mount was successful by listing the contents of the top level directory of the mount point:

```
# ls /mnt/windows
'$Recycle.Bin'          ProgramData             swapfile.sys
'Documents and Settings'  'Program Files'        'System Volume Information'
pagefile.sys           'Program Files (x86)\'  Users
PerfLogs               Recovery                Windows
```

To automate the mount each time the system is booted, simply add the appropriate mount line to the `/etc/fstab` file:

```
/dev/sda2 /mnt/windows ntfs defaults 0 0
```

To unmount the Windows file system at any time:

```
# umount /mnt/windows
```

4.4 Summary

Fedora 31 can safely co-exist on the same disk drive as a Windows operating system by creating a dual boot environment. This involves shrinking the amount of space occupied by the Windows system to make room for Fedora 31 before performing the installation. To access the Windows filesystem from within Fedora, the Fuse NTFS driver needs to be installed and used to mount the Windows partitions.

5. Allocating Windows Disk Partitions to Fedora 31

In the previous chapter we looked at how to install on the same disk as Windows. This so called “dual boot” configuration allows the user to have both operating systems installed on a single disk drive with the option to boot one or the other when the system is powered on.

This chapter is intended for users who have decided they like Fedora 31 enough to delete Windows entirely from the disk, and use the resulting space for Linux. In the following sections we will work through this process step by step.

5.1 Unmounting the Windows Partition

If the steps in the “*Dual Booting Fedora 31 with Windows*” chapter were followed to mount the Windows partition from within Fedora 31, steps should be taken to unmount the partition before continuing with this chapter. Assuming that the Windows partition was mounted as `/mnt/windows`, it can be unmounted as follows:

```
# umount /mnt/windows
```

The `/etc/fstab` file should also be edited to remove the `/mnt/windows` auto-mount if it was previously added.

5.2 Deleting the Windows Partitions from the Disk

The first step in freeing up the Windows partition for use by Fedora is to delete that partition. Before doing so, however, it is imperative that any data you need to keep is backed up from both the Windows and Fedora partitions. Having done that, it is safe to proceed with this chapter.

In order to remove the Windows partitions we first need to identify the disk on which they reside using the `fdisk` tool:

```
# fdisk -l
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe3673009
```

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1	*	2048	1126399	1124352	549M	7	HPFS/NTFS/exFAT
/dev/sda2		1126400	53655551	52529152	25G	7	HPFS/NTFS/exFAT
/dev/sda3		53655552	55752703	2097152	1G	83	Linux

Allocating Windows Disk Partitions to Fedora 31

```
/dev/sda4      55752704 104857599 49104896 23.4G  5 Extended
/dev/sda5      55754752 104857599 49102848 23.4G  8e Linux LVM
```

In the above example output the system contains one physical disk drive referenced by device name `/dev/sda`. On that disk drive are five partitions accessed via the device names `/dev/sda1` through `/dev/sda5` respectively. Based on the values in the System column, there are two NTFS partitions. The first is the Windows system partition while the second, much larger, NTFS partition is the Windows boot partition containing the Windows operating system and user data.

To remove the partitions, start the `fdisk` tool using the device name of the disk containing the partition (`/dev/sda` in this instance) and follow the instructions to once again display the partition and sector information:

```
# fdisk /dev/sda
```

```
Welcome to fdisk (util-linux 2.32.1).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.
```

```
Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe3673009
```

Device	Boot	Start	End	Sectors	Size	Id	Type
<code>/dev/sda1</code>	*	2048	1126399	1124352	549M	7	HPFS/NTFS/exFAT
<code>/dev/sda2</code>		1126400	53655551	52529152	25G	7	HPFS/NTFS/exFAT
<code>/dev/sda3</code>		53655552	55752703	2097152	1G	83	Linux
<code>/dev/sda4</code>		55752704	104857599	49104896	23.4G	5	Extended
<code>/dev/sda5</code>		55754752	104857599	49102848	23.4G	8e	Linux LVM

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

Currently, the Windows system partition is listed as being the bootable partition. Since we will be deleting this partition, the Linux boot partition needs to be marked as bootable. In the above configuration, this is represented by `/dev/sda3`. Remaining within the `fdisk` tool, make this the bootable partition as follows:

```
Command (m for help): a
Partition number (1,3-5, default 5): 3
The bootable flag on partition 3 is enabled now.
```

Before proceeding, make a note of the start and end addresses of the partitions we will be deleting (in other words the start of `/dev/sda1` and the end of `/dev/sda2`).

At the command prompt, delete the Windows partitions (these being partitions 1 and 2 on our example system):

```
Command (m for help): d
Partition number (1-5, default 5): 1
```

Partition 1 has been deleted.

```
Command (m for help): d
Partition number (2-5, default 5): 2
```

Partition 2 has been deleted.

Now that we have deleted the Windows partitions we need to create the new partition in the vacated disk space. The partition number must match the number of the partition removed (in this case 1) and is going to be a primary partition. It will also be necessary to enter the Start and End sectors of the partition exactly as reported for the old partition (*fdisk* will typically offer the correct values by default, though it is wise to double check). If you are prompted to remove the NTFS signature, enter Y:

```
Command (m for help): n
Partition type
  p   primary (1 primary, 1 extended, 2 free)
  l   logical (numbered from 5)
Select (default p): p
Partition number (1,2, default 1): 1
First sector (2048-104857599, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-53655551, default 53655551):
```

Created a new partition 1 of type 'Linux' and of size 25.6 GiB.
Partition #1 contains a ntfs signature.

Do you want to remove the signature? [Y]es/[N]o: y

The signature will be removed by a write command.

Having made these changes the next step is to check that the settings are correct (taking this opportunity to double check that the Linux boot partition is bootable):

```
Command (m for help): p
Disk /dev/sda: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe3673009
```

Allocating Windows Disk Partitions to Fedora 31

Device	Boot	Start	End	Sectors	Size	Id	Type
/dev/sda1		2048	53655551	53653504	25.6G	83	Linux
/dev/sda3	*	53655552	55752703	2097152	1G	83	Linux
/dev/sda4		55752704	104857599	49104896	23.4G	5	Extended
/dev/sda5		55754752	104857599	49102848	23.4G	8e	Linux LVM

Filesystem/RAID signature on partition 1 will be wiped.

To commit the changes we now need to write the new partition information to disk and quit from the *fdisk* tool:

```
Command (m for help): w
The partition table has been altered.
Syncing disks.
```

5.3 Formatting the Unallocated Disk Partition

In order to make the new partition suitable for use by Fedora 31, it needs to have a file system created on it. The default file system type for the current release of Fedora is XFS and will be covered in greater detail in the chapter entitled “*Adding a New Disk Drive to a Fedora 31 System*”. Creation of the file system is performed using the *mkfs.xfs* command as follows:

```
# mkfs.xfs -f /dev/sda1
meta-data=/dev/sda1          isize=512    agcount=4, agsize=1676672 blks
=                               sectsz=512   attr=2, projid32bit=1
=                               crc=1       finobt=1, sparse=1, rmapbt=0
=                               reflink=1
data            =             bsize=4096  blocks=6706688, imaxpct=25
=                               sunit=0     swidth=0 blks
naming         =version 2     bsize=4096  ascii-ci=0, ftype=1
log            =internal log  bsize=4096  blocks=3274, version=2
=                               sectsz=512   sunit=0 blks, lazy-count=1
realtime      =none          extsz=4096  blocks=0, rtextents=0
```

5.4 Mounting the New Partition

Next, we need to mount the new partition. In this example we will mount it in a directory named */data*. You are free, however, to mount the new partition using any valid mount point you desire or to use it as part of a logical volume (details of which are covered in the chapter entitled “*Adding a New Disk to a Fedora 31 Volume Group and Logical Volume*”). First we need to create the directory to act as the mount point:

```
# mkdir /data
```

Secondly, we need to edit the mount table in */etc/fstab* so that the partition is automatically mounted each time the system starts. At the bottom of the */etc/fstab* file, add the following line to mount the new partition (modifying the */dev/sda1* device to match your environment):

```
/dev/sda1 /data xfs defaults 0 0
```

Finally, we can manually mount the new partition (note that on subsequent reboots this will not

be necessary as the partition will automount as a result of the setting we added to the */etc/fstab* file above).

```
# mount /data
```

To check the partition, run the following command to display the available space:

```
# df -h /data
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       26G   215M   26G   1% /data
```

5.5 Editing the Boot Menu

The next step is to modify the Fedora boot menu. Since this was originally a dual boot system, the menu is configured to provide the option of booting either Windows or Fedora. Now that the Windows partition is gone, we need to remove this boot option. Start by editing the */etc/grub.d/40_custom* file and removing the Windows menu entry:

```
#!/bin/sh
exec tail -n +3 $0
# This file provides an easy way to add custom menu entries.  Simply type the
# menu entries you want to add after this comment.  Be careful not to change
# the 'exec tail' line above.
menuentry "Windows 10" {
    set root=(hd0,1)
    chainloader +1
}
```

Save the file and use the *grub2-mkconfig* tool to generate the */boot/grub2/grub.cfg* file as follows:

```
# grub2-mkconfig --output=/boot/grub2/grub.cfg
Generating grub configuration file ...
done
```

Next time the system restarts, the Windows 10 option will no longer be provided by the boot menu.

5.6 Summary

The Windows partitions in a dual boot configuration can be removed at any time to free up space for a Fedora system by identifying which partitions belong to Windows and then deleting them using the *fdisk* tool. Once deleted, the unallocated space can be used to create a new filesystem and mounted to make it available to the Fedora system. The final task is to remove the Windows option from the boot menu configuration.

6. A Guided Tour of the GNOME 3 Desktop

Fedora 31 includes the GNOME 3 desktop environment. Although lacking the complexity of Windows and macOS desktops, GNOME 3 provides an uncluttered and intuitive desktop environment that provides all of the essential features of a windowing environment with the added advantage that it can be learned quickly.

In this chapter, the main features of the GNOME desktop will be covered together with an outline of how basic tasks are performed.

6.1 Installing the GNOME Desktop

If either the Workstation or Server with GUI software configuration was selected during the Fedora 31 installation process, the GNOME desktop will already be installed and will automatically launch each time the system starts.

If any other software configuration was selected during the Fedora 31 installation process, the GNOME desktop will not have been included in the packages installed on the system. On server-based systems without a display attached, the idea of installing a graphical desktop environment may seem redundant. It is worth noting, however, that remote access to the GNOME desktop is also possible so, even on so called *headless servers* (i.e. servers lacking a monitor, keyboard and mouse) it may still be beneficial to install the GNOME desktop packages. The topic of establishing remote desktop access will be covered in detail in the “*Fedora 31 Remote Desktop Access with VNC*” chapter of this book.

If the installation configuration did not include the GNOME desktop, it may be installed at any time using the following command:

```
# dnf groupinstall "Workstation"
```

Once the installation is complete, the desktop environment may be launched from the command prompt on a monitor as follows:

```
$ startx
```

6.2 An Overview of the GNOME 3 Desktop

The screen shown in Figure 6-1 below shows the appearance of a typical, newly launched GNOME desktop session before any other programs have been launched or configuration changes made.

A Guided Tour of the GNOME 3 Desktop

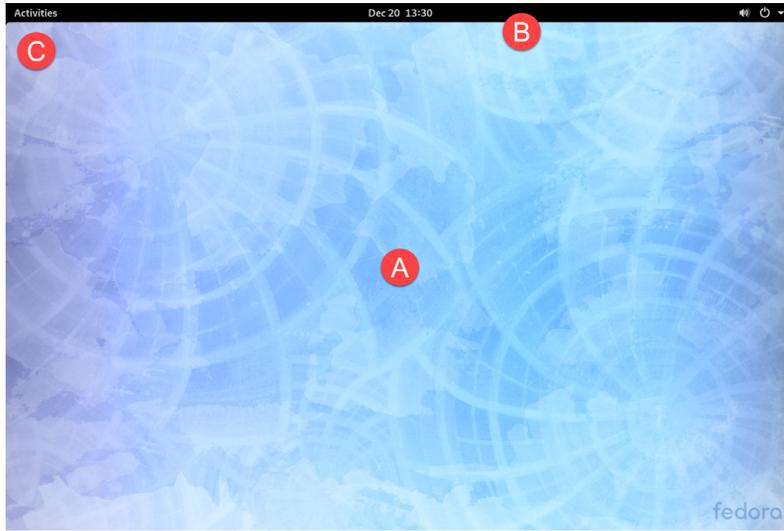


Figure 6-1

The main desktop area (marked A) is where windows will appear when applications and utilities are launched. Unlike other desktop environments, it is not possible to drag and drop files or applications onto the desktop, providing a clean and uncluttered workspace.

The bar at the top of the screen (B) is called the *top bar* and includes the Activities menu (C), the day and time and a collection of buttons and icons including network status, audio volume, battery power and other status and account settings. The application menu for the currently active application running on the desktop will also appear in the top bar. Figure 6-2, for example, shows the application menu for the Terminal program:

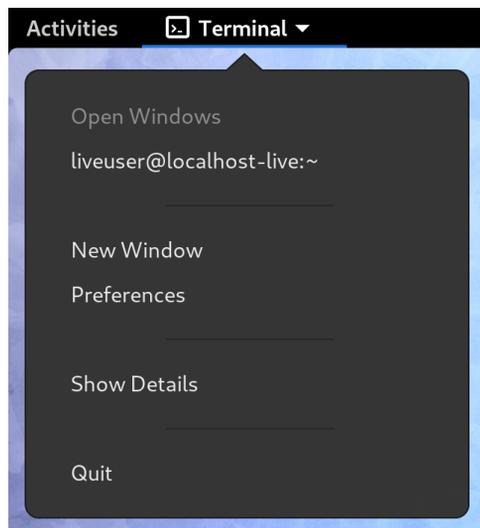


Figure 6-2

6.3 Launching Activities

Applications and utilities are launched using the Activities overview dashboard (referred to as the *dash*) which may be displayed either by clicking on the Activities button in the top bar or pressing the *special key* on the keyboard. On Windows systems this is the Windows key, on macOS the Command key and on Chromebooks the key displaying a magnifying glass.

When displayed, the dash will appear as shown in Figure 6-3 below:

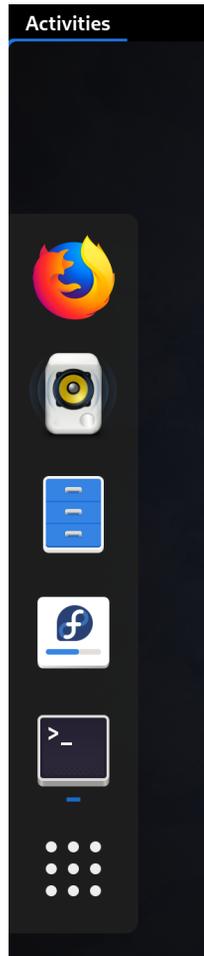


Figure 6-3

By default the dash will display an icon for a predefined set of commonly used applications and will also include an icon for any applications that are currently running. If the application is currently running it will appear with a bar marker beneath the icon.

To launch an application, simply click on the icon in the dash.

To find an application not included on the dash, one option is to select the bottom most icon (the

A Guided Tour of the GNOME 3 Desktop

square comprising nine dots) to display a browsable list of applications as shown in Figure 6-4:

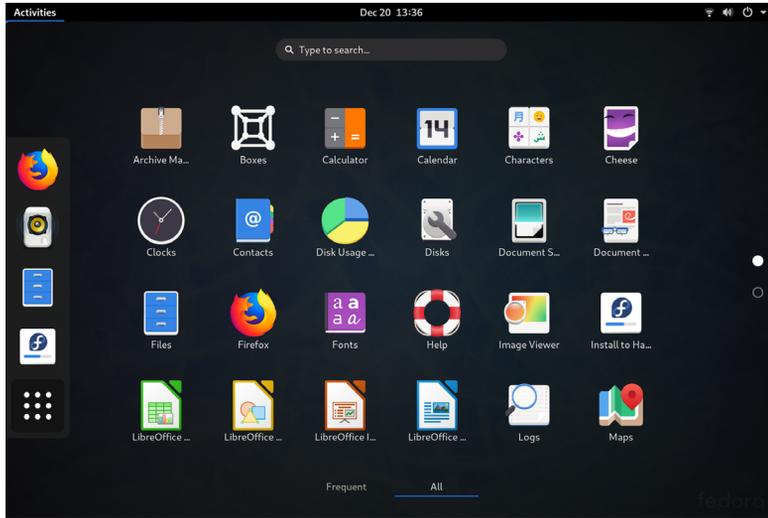


Figure 6-4

Note that the list can be filtered to display all applications or only those used frequently by selecting the buttons at the bottom center of the screen. It is also important to be aware that some entries in the list are actually folders holding additional applications. In the above screenshot, for example, the *Utilities* entry provides access to a collection of other tools such as the system monitor and disk management tools and the Terminal window application.

An alternative to browsing the applications is to perform a search using the search bar which appears when the dash is displayed as shown in Figure 6-5:

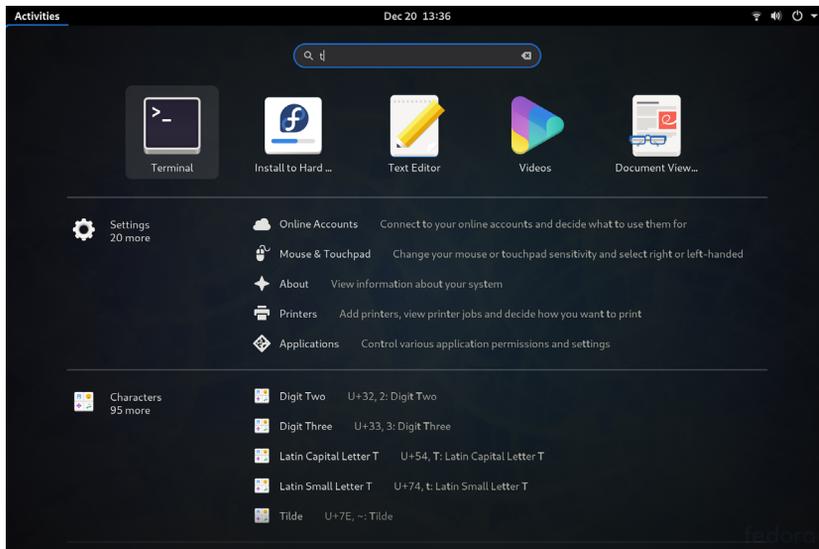


Figure 6-5

As text is typed into the search box, the list of possible matches will be refined.

To add an application to the dash for more convenient access, locate the icon for the application, right-click on it and select the *Add to Favorites* menu option:

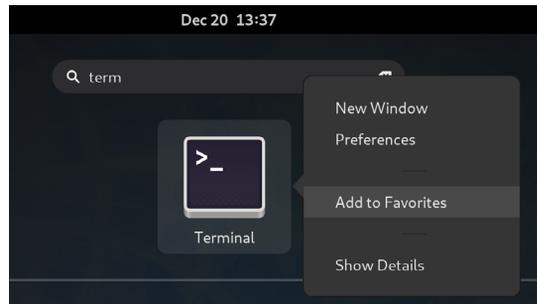


Figure 6-6

To remove an app from the dash, repeat these steps, this time selecting *Remove from Favorites*.

6.4 Managing Windows

As with other desktop environments, applications run on GNOME in windows. When multiple application windows are open, the Super + Tab keyboard shortcut will display the switcher panel (Figure 6-7) allowing a different window to be chosen as the currently active window (the Super key is either the Windows key or, in the case of a Mac keyboard, the Cmd key):



Figure 6-7

If a single application has more than one window open, the switcher will display those windows in a second panel so that a specific window can be selected:

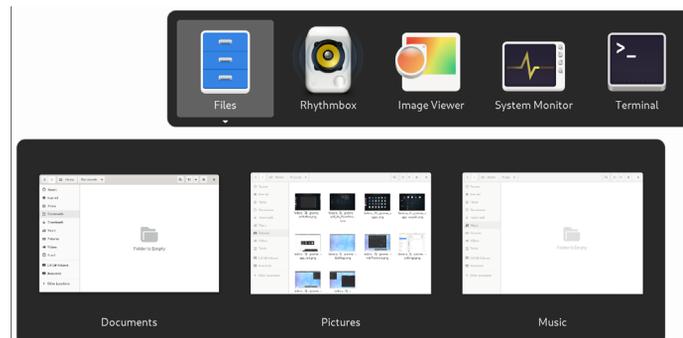


Figure 6-8

A Guided Tour of the GNOME 3 Desktop

To cycle backwards through the icons in the switcher, use the Shift + Tab keyboard shortcut.

To maximize a window so that it fills the entire screen click the title bar and drag the window to the top of the screen. To return the window to its original size, click on the title bar and drag downwards. Alternatively, simply double-click on the title bar to toggle between window sizes. Similarly, dragging a window to the left or right side of the screen will cause the window to fill that half of the screen.

6.5 Using Workspaces

The area of the screen where the application windows appear is referred to as the *workspace* and GNOME 3 allows multiple workspaces to be configured. To create a new workspace, display the Activities overview and move the mouse pointer to the far right of the screen to display the workspaces panel (Figure 6-9):

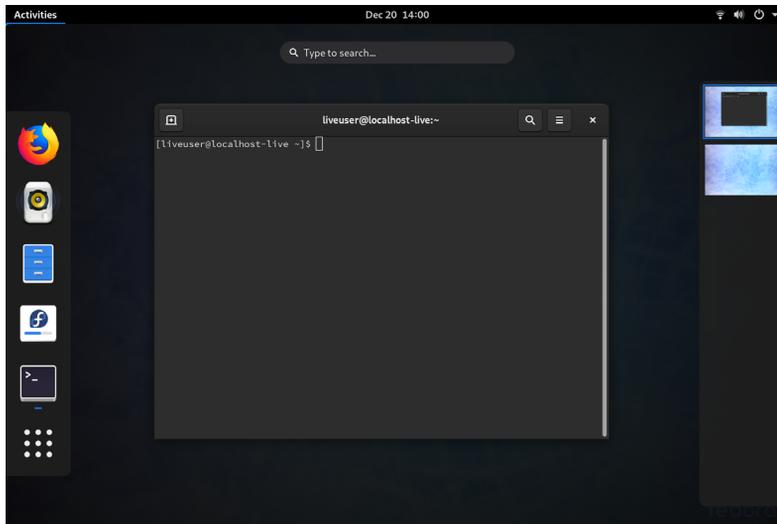


Figure 6-9

To switch to a different panel, simply select it from the list. To move a window from one workspace to another, display the workspaces panel and drag and drop the application window (either the actual window from the current workspace or the thumbnail window in the workspaces panel) onto the destination workspace. When a window is added to a blank workspace, another blank workspace is added to the workspace panel, allowing multiple workspaces to be created.

To remove a workspace either close all the windows on that workspace, or move them to another workspace.

6.6 Calendar and Notifications

When the system needs to notify you of an event (such as the availability of system or application updates), a popup panel will appear at the top of the workspace. Access to the calendar and any previous notifications is available by clicking on the day and time in the top bar as shown in Figure 6-10:

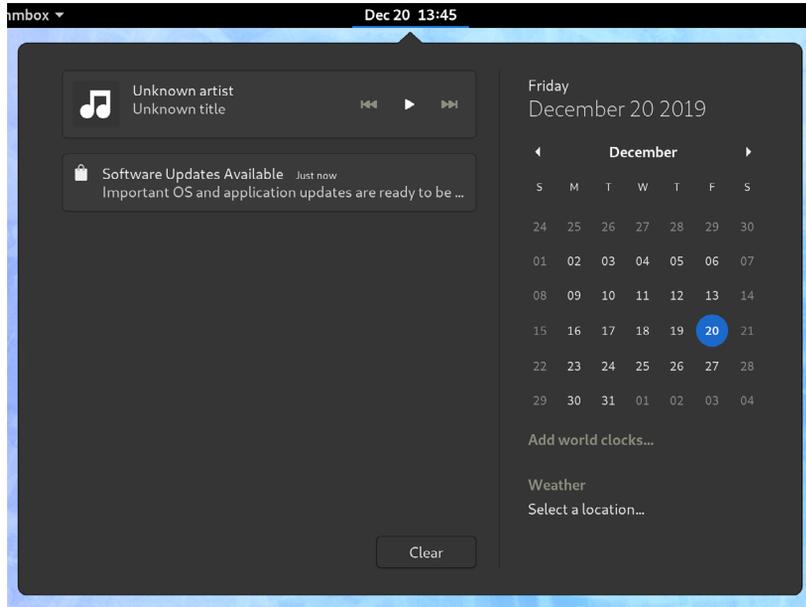


Figure 6-10

6.7 Desktop Settings

To access the Settings application, click on the down arrow on the far right of the top bar and select the button with the tools icon as highlighted in Figure 6-11:

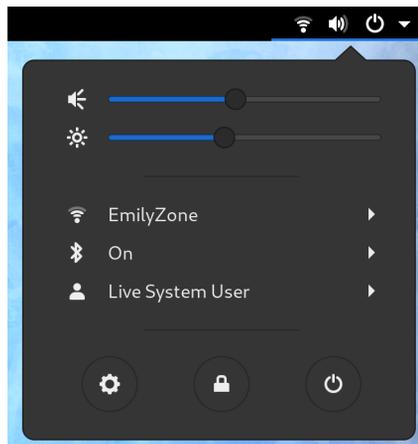


Figure 6-11

The Settings application provides a wide range of options such as Ethernet and Wi-Fi connections, screen background customization options, screen locking and power management controls and language preferences. To explore the settings available in each category, simply select an option from the left-hand panel in the Settings window:

A Guided Tour of the GNOME 3 Desktop

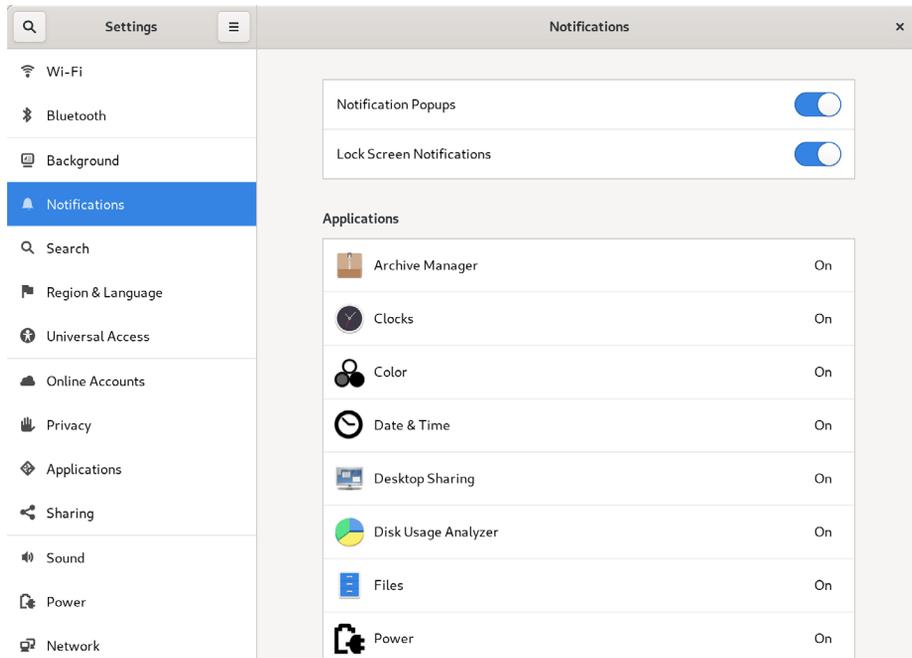


Figure 6-12

The menu shown in Figure 6-11 above also includes options to switch user, adjust audio volume, change to a different Wi-Fi network and to log out, restart or power off the system.

6.8 Summary

Fedora 31 includes the GNOME 3 desktop environment which may either be included during the initial installation or installed later using the *dnf* group package installation feature. Unlike most other desktop environments, GNOME 3 is intended to provide a clean and easy to use windowing user interface. Key areas of the GNOME 3 desktop include the top bar, Activities overview and dash. In addition, GNOME 3 supports multiple workspaces keeping running applications organized and the screen uncluttered. A variety of configuration options is also available within the Settings app including desktop background settings, audio, network configuration and Wi-Fi network selection.

7. An Overview of the Fedora 31 Cockpit Web Interface

Although it comes equipped with the latest in Linux desktop environments, Fedora is very much a server operating system and, as such, the majority of Fedora deployments will either be to remote physical servers or as cloud-based virtual machine instances. Invariably, these systems run without a keyboard, mouse or monitor, with direct access only available via the command-prompt over a network connection. This presents a challenge in terms of administering the system from remote locations. While much can certainly be achieved via remote access to the command-line and desktop environments, this is far from a consistent and cohesive solution to the administrative and monitoring tasks that need to be performed on a daily basis on an enterprise level operating system such as Fedora 31.

This issue has been addressed with the introduction of the Cockpit web-based administration interface. This chapter will explain how to install, configure and access the Cockpit interface while also providing an overview of the key features of Cockpit, many of which will be covered in greater detail in later chapters.

7.1 An Overview of Cockpit

Cockpit is a light-weight, web-based interface that allows general system administrative tasks to be performed remotely. When installed and configured, the system administrator simply opens a local browser window and navigates to the Cockpit port on the remote server. After loading the Cockpit interface into the browser and logging in, a wide range of tasks can be performed visually using administration and monitoring tools.

Behind the scenes, Cockpit uses the same tools to perform tasks as would normally be used when working at the command-line, and updates automatically to reflect changes occurring elsewhere on the system. This allows Cockpit to be used in conjunction with other administration tools and techniques without the risk of one approach overriding another. Cockpit can also be configured to access more than one server, allowing multiple servers to be administered and monitored simultaneously through a single browser session.

Cockpit is installed by default with a wide range of tools already bundled. Cockpit also, however, allows additional extension plugins to be installed as needed. Cockpit is also designed so that you can create your own extensions using a combination of HTML and JavaScript to add missing or custom functionality.

Cockpit's modular design also allows many features to be embedded into other web-based applications.

An Overview of the Fedora 31 Cockpit Web Interface

7.2 Installing and Enabling Cockpit

Cockpit is generally not installed on Fedora 31 by default, but can be set up and enabled in a few simple steps. The first step is to install the Cockpit package as follows:

```
# dnf install cockpit
```

Next, the Cockpit socket service needs to be enabled:

```
# systemctl enable --now cockpit.socket
```

Finally, the necessary ports need to be opened on the firewall to allow remote browser connections to reach Cockpit:

```
# firewall-cmd --add-service=cockpit --permanent
```

```
# firewall-cmd --reload
```

7.3 Accessing Cockpit

If you have access to the desktop environment of the server on which Cockpit has been installed, open a browser window and navigate to `https://localhost:9090` to access the Cockpit sign in screen. If, on the other hand, the server is remote, simply navigate to the server using the domain name or IP address (for example `https://myserver.com:9090`).

When the connection is established, the browser may issue a warning that the connection is not secure. This is because the Cockpit service is using a self-signed certificate. Either select the option to proceed to the web site or, to avoid this message in the future, select the advanced option and add an exception for the server address.

Once connected, the browser will load the log in page shown in Figure 7-1 below:

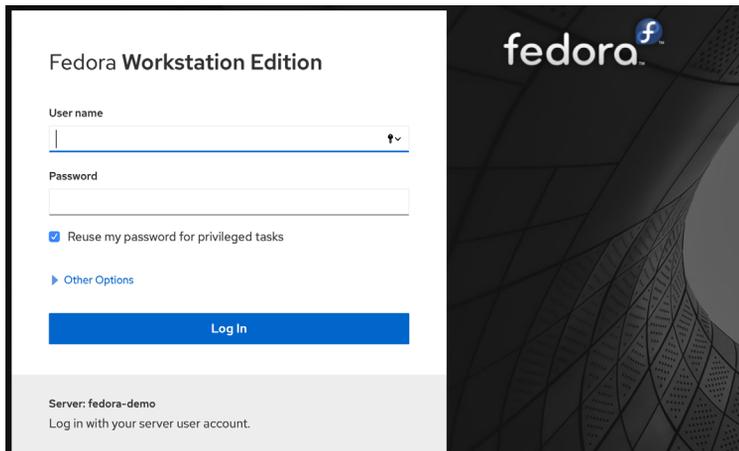


Figure 7-1

Sign in to the Cockpit interface either as root or with your a user account credentials. Note that when signed in as a user some tasks will be restricted within the Cockpit interface due to permission constraints. After signing in, Cockpit will display the System screen.

7.4 System

The System screen provides an overview of the current system including realtime performance metrics for CPU, memory, disk I/O and network usage. This screen also includes information about the system including the underlying hardware, host name, system time and whether the system software is up to date. Options are also provided to restart or shutdown the system.

Figure 7-2, for example, shows the system monitoring page of the Cockpit interface:

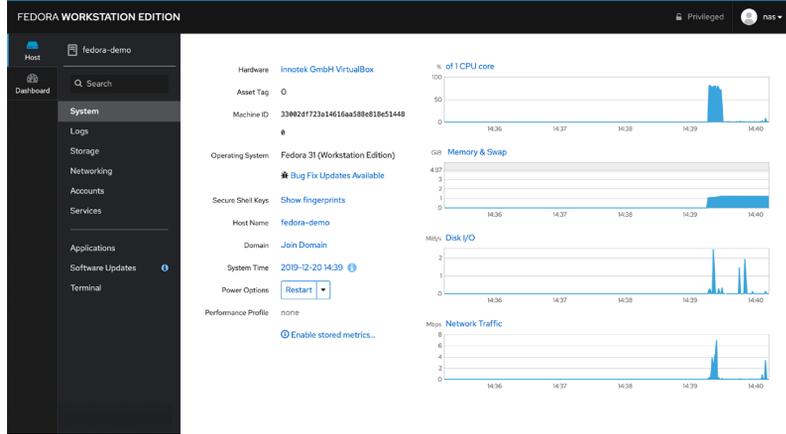


Figure 7-2

7.5 Logs

When the Logs category is selected, Cockpit displays the contents of the *systemd* journal logs. Selecting a log entry will display the entire log message. The log entries are ordered with the most recent at the top and menus are included to filter the logs for different time durations and based on message severity.

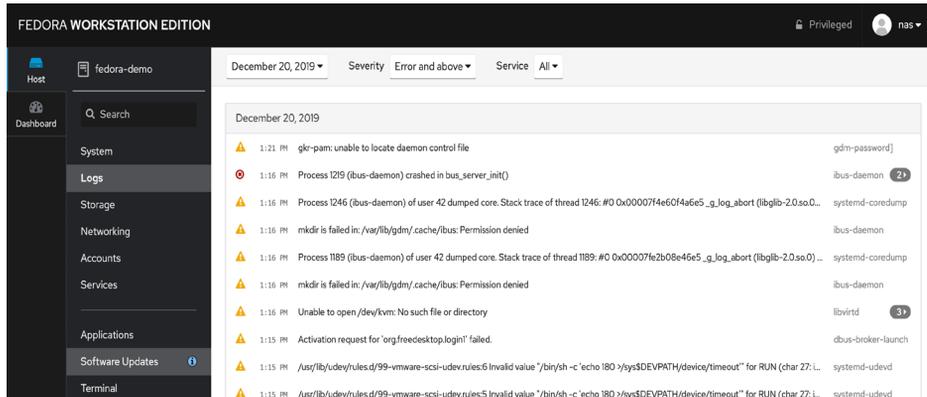


Figure 7-3

An Overview of the Fedora 31 Cockpit Web Interface

7.6 Storage

Select the Storage option to review and manage the storage on the system including disks, partitions and volume groups, Network File System (NFS) mounts and RAID storage. This screen also allows disk I/O activity to be monitored in realtime and lists log output from the system *udisksd* service used to query and manage storage devices.

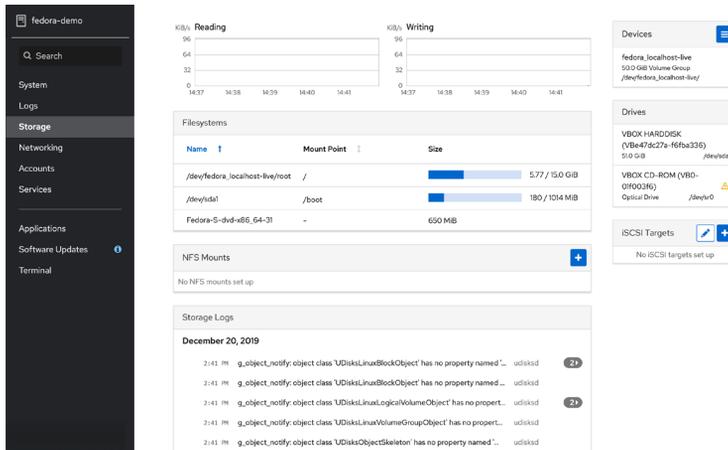


Figure 7-4

7.7 Networking

The Network screen provides information on a wide range of network related configurations and services including network interfaces and firewall settings and allows configuration changes to be made such as creating network bridges or setting up virtual networks.

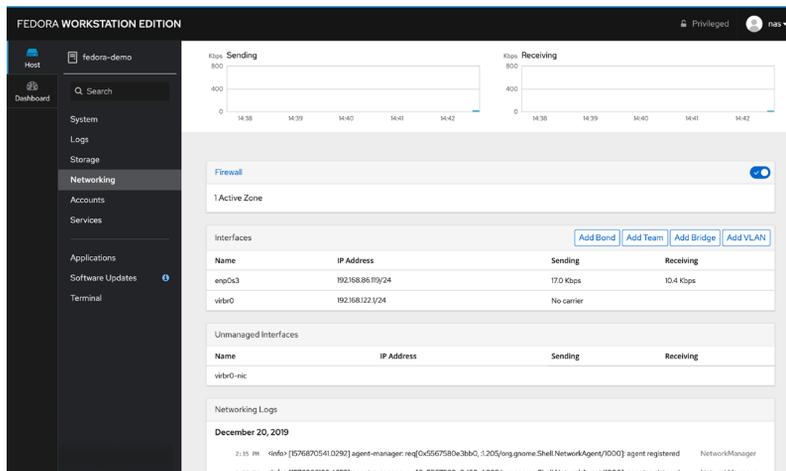


Figure 7-5

7.8 Accounts

Select this option to view the current user accounts configured on the system, and create accounts for additional users. The topic of user management will be covered later in the chapter entitled “*Managing Fedora 31 Users and Groups*”.



Figure 7-6

Click on an existing account to view details and make changes. The user account details page may also be used to review and add Public SSH keys to the user’s account for remote access to the server as outlined in the chapter entitled “*Configuring SSH Key-based Authentication on Fedora 31*”.

7.9 Services

This screen displays a list of the system services running on the server and allows those services to be added, removed, stopped and started.

Name	Description	State	Automatic Startup
abrt-journal-core	Creates ABRT problems from coredumpctl messages	active (running)	Enabled
abrt-oops	ABRT kernel log watcher	active (running)	Enabled
abrt-pstoreoops	Collect UEFI-saved oopses for ABRT	inactive (dead)	Disabled
abrt-vmcore	Harvest vmcores for ABRT	inactive (dead)	Enabled
abrt-xorg	ABRT Xorg log watcher	active (running)	Enabled
abrttd	ABRT Automated Bug Reporting Tool	active (running)	Enabled
accounts-daemon	Accounts Service	active (running)	Enabled
alsa-restore	Save/Restore Sound Card State	inactive (dead)	Static
alsa-state	Manage Sound Card State (restore and store)	active (running)	Static

Figure 7-7

The topic of services will be covered in detail in the chapter entitled “*Configuring Fedora 31 systemd Units*”.

7.10 Applications

As previously mentioned, additional functionality can be added to Cockpit in the form of extensions. These can either be self-developed extensions, or those provided by third parties. The Applications screen lists installed extensions and allows extensions to be added or deleted.

An Overview of the Fedora 31 Cockpit Web Interface

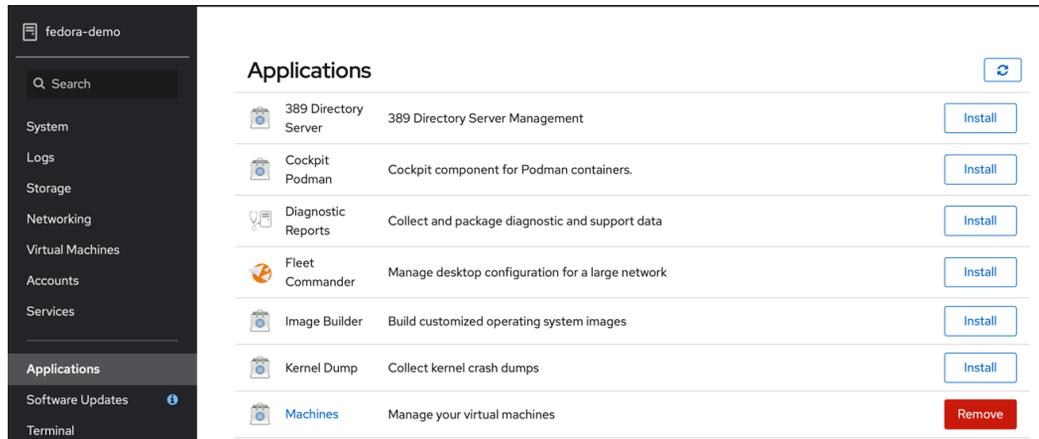


Figure 7-8

7.11 Virtual Machines

Virtualization allows multiple operating system instances to run simultaneously on a single computer system, with each system running inside its own *virtual machine*. The Virtual Machines Cockpit extension provides a way to create and manage the virtual machine guests installed on the server.

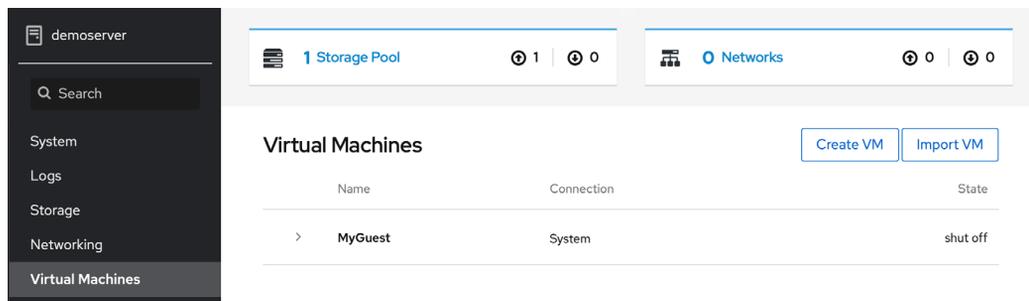


Figure 7-9

The Virtual Machines extension is not installed by default but can be added via the Cockpit Applications screen or by running the following command:

```
# dnf install cockpit-machines
```

The use of virtualization with Fedora 31 is covered starting with the chapter entitled “*An Overview of Virtualization Techniques*”.

7.12 Podman Containers

Linux containers provide a way to run multiple instances of an operating system on a single computer, all sharing the kernel of the host system, providing a lightweight and highly efficient alternative to virtualization. Podman is one of a set of tools provided with Fedora 31 to create and

manage Linux containers and the Podman Containers Cockpit screen provides a user friendly environment in which to download and create container images.

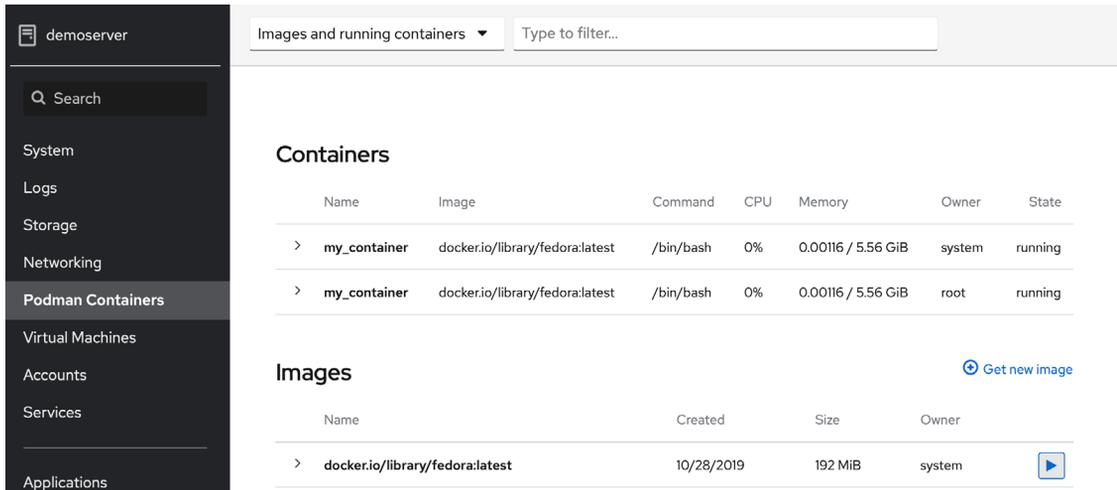


Figure 7-10

The Podman extension is not installed by default but can be added via the Cockpit Applications screen or by running the following command:

```
# dnf install cockpit-podman
```

Linux containers are covered later in the book starting with “*An Introduction to Linux Containers*”.

7.13 Diagnostic Reports

When selected, this option allows a diagnostic report to be generated and downloaded to the local system for analysis. The generated report is provided in the form of a compressed archive file containing a collection of all the log files on the server system.

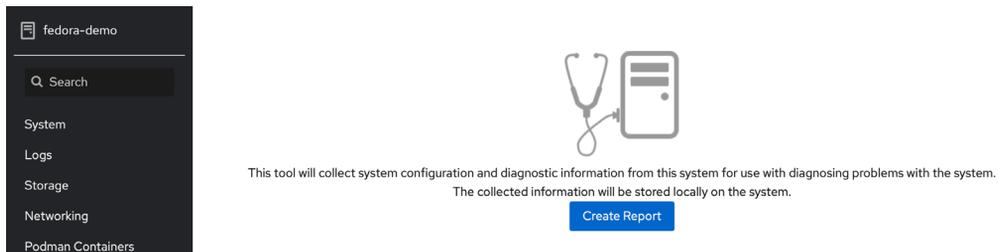


Figure 7-11

If the Diagnostic Reports extension is not installed it can be added via the Cockpit Applications screen or by running the following command:

```
# dnf install sosreport
```

An Overview of the Fedora 31 Cockpit Web Interface

7.14 Kernel Dump

This screen simply displays the current KDump service status and indicates where the kernel dump file will be located in the event of a kernel crash on the system. This dump file can then be used to identify the cause of the crash. A button is provided to test the Kernel dump by triggering a kernel crash, though use of this option is not recommended on a production server system:

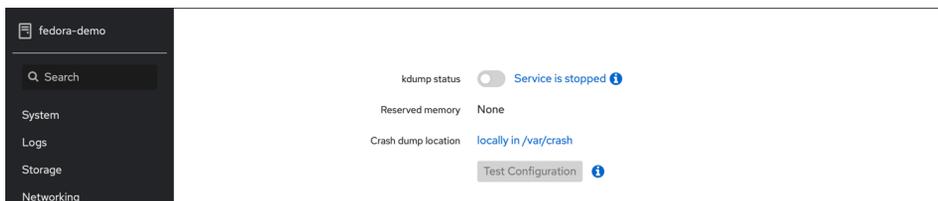


Figure 7-12

The Kernel Dump extension may not be installed by default but can be added via the Cockpit Applications screen or by running the following command:

```
# dnf install kdump
```

7.15 SELinux

When selected, the SELinux category indicates whether SELinux policy is currently being enforced and displays any alerts generated by this security system.

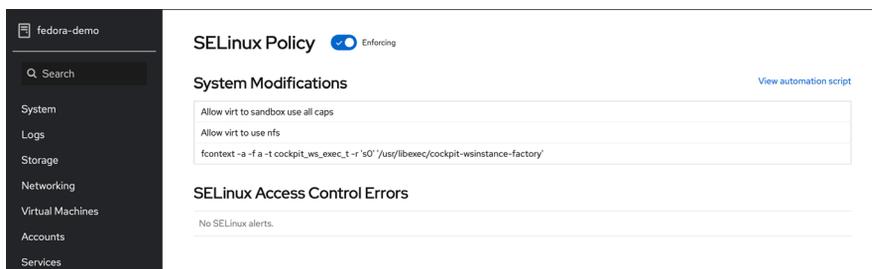


Figure 7-13

The SELinux extension is not installed by default but can be added via the Cockpit Applications screen or by running the following command:

```
# dnf install cockpit-selinux
```

7.16 Software Updates

If any software updates are available for the system they will be listed on this screen. If updates are available, they can be installed from this screen.

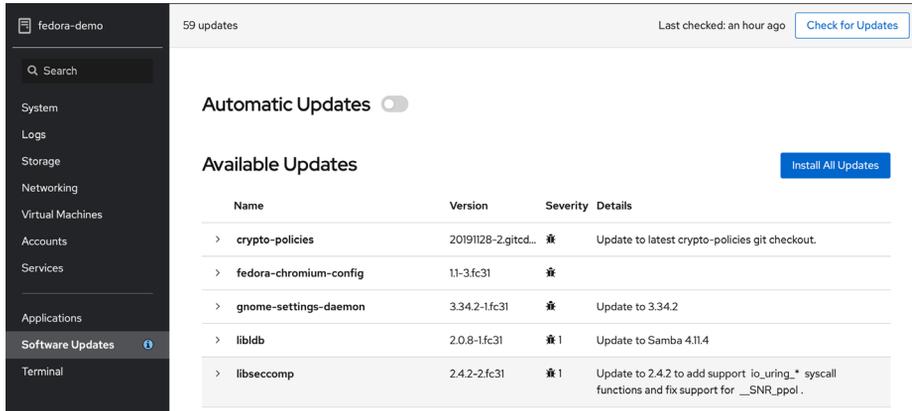


Figure 7-14

7.17 Terminal

As the name suggests, the Terminal screen provides access to the command-line prompt.

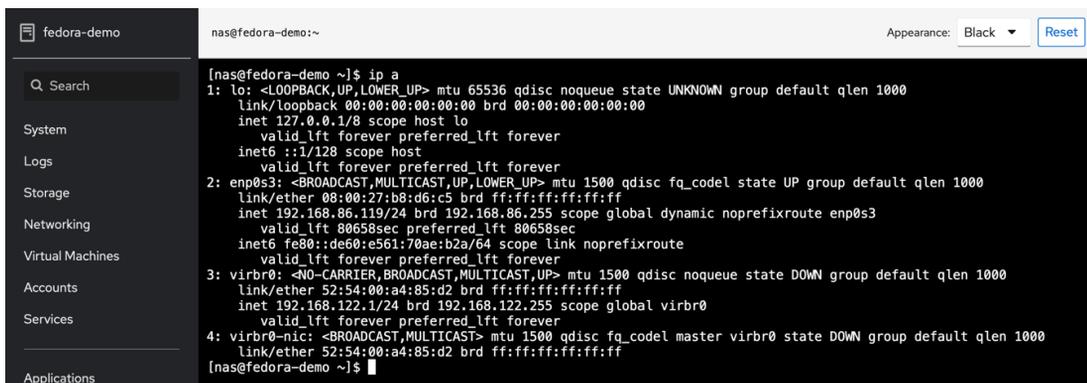


Figure 7-15

7.18 Connecting to Multiple Servers

Cockpit can be configured to administer multiple servers from within a single session. This requires that the Cockpit dashboard be installed on the primary system (in other words the system to which the initial Cockpit session will be established). To install the Cockpit dashboard package, run the following command:

```
# dnf install cockpit-dashboard
```

Once the dashboard has been installed, sign out of Cockpit and then sign in again. The dashboard will now appear in the Cockpit interface as highlighted in Figure 7-16:

An Overview of the Fedora 31 Cockpit Web Interface

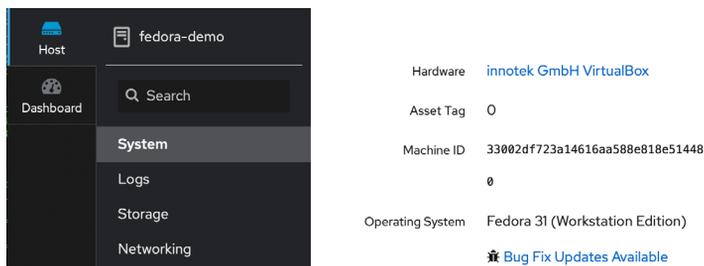


Figure 7-16

When selected, the dashboard page will display performance graphs for the current system and provide a list of currently connected systems:

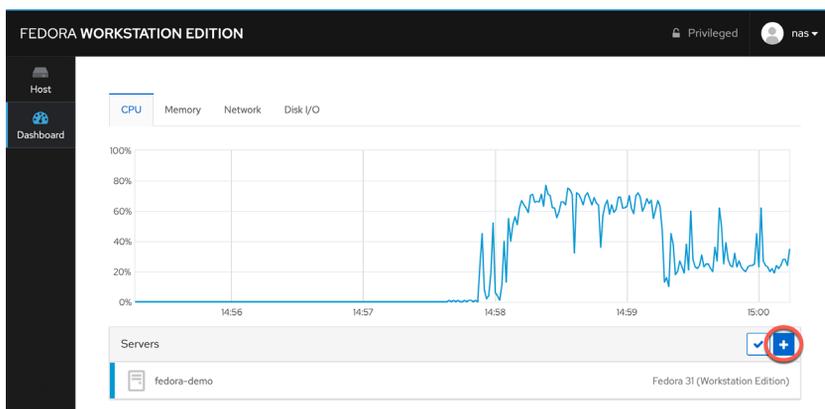
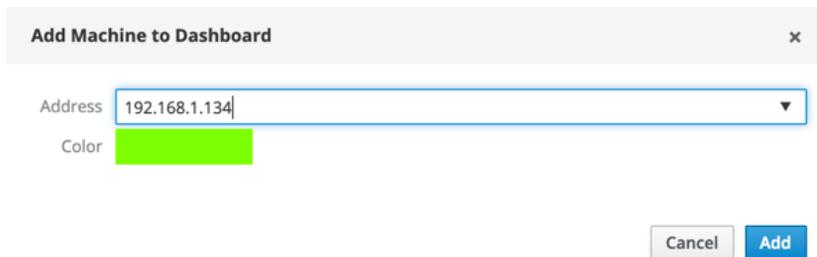


Figure 7-17

To add another system, click on the + button highlighted in Figure 7-17 above, enter the IP address or host name of the other system and select a color by which to distinguish this server from any others added to Cockpit before clicking on the Add button:



The screenshot shows the 'Add Machine to Dashboard' dialog box. It has a title bar with a close button (x). Below the title bar, there is an 'Address' field containing '192.168.1.134' and a 'Color' field with a green color swatch. At the bottom right, there are 'Cancel' and 'Add' buttons.

Figure 7-18

Enter the user name and password to be used when connecting to the other system, then click on the log in button. The newly added server will now be listed in the Cockpit dashboard and will appear in graphs represented by the previously selected color:

An Overview of the Fedora 31 Cockpit Web Interface

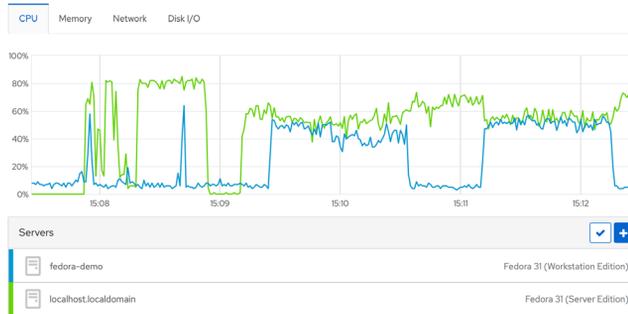


Figure 7-19

To switch between systems when using Cockpit, simply use the drop down menu shown in Figure 7-20 below:

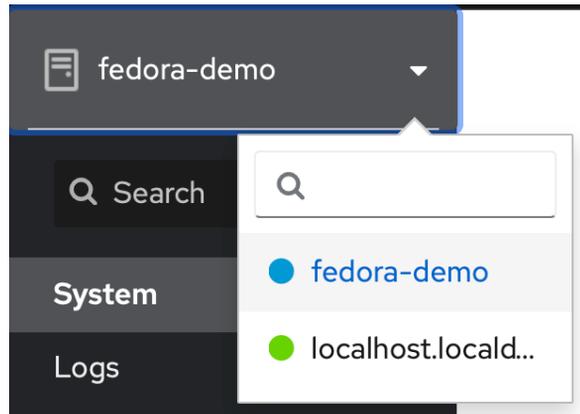


Figure 7-20

7.19 Enabling Stored Metrics

In a standard installation Cockpit does not retain any of the performance metric data beyond what is displayed in the short time window covered by the graphs. To retain the data collected by Cockpit, the stored metrics feature needs to be installed. Begin by installing the *cockpit-pcp* package as follows:

```
# dnf install cockpit-pcp
```

To enable metric persistence, display the System screen and click on the *Enable stored metrics...* link highlighted in Figure 7-21:

An Overview of the Fedora 31 Cockpit Web Interface

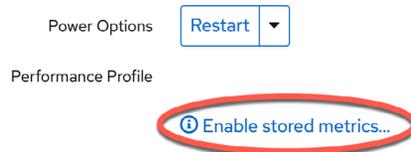


Figure 7-21

Once this option has been selected, Cockpit will request permission to install the *cockpit-pcp*, *pcp-lib*s and *pcp-selinux* packages. Once these packages have been installed, the *Enable persistent metrics* link will have been replaced by a control labeled *Store Metrics*. When this option is enabled, the performance graphs will include additional controls allowing you to move back and forth in time to view historical data and to change the data time frame ranging from 5 minutes up to 1 week.

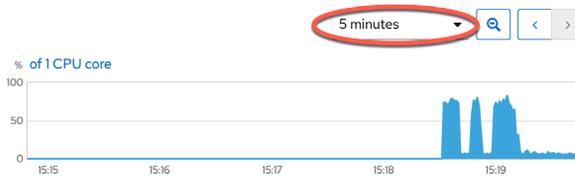


Figure 7-22

7.20 Summary

The Cockpit web interface allows remote system administration tasks to be performed visually from within a web browser without the need to rely on the command-prompt and command-line tools. Once installed and enabled, the system administrator simply opens a web browser, connects to the remote server and signs into the Cockpit interface. Behind the scenes, Cockpit uses the same command-line tools as those available via the command prompt, thereby allowing both options to be used without the risk of configuration conflicts. Cockpit uses a modular framework allowing additional extensions to be added, and for custom extensions to be developed and integrated. A Cockpit session can be used to administer a single server, or configured to access multiple servers simultaneously.

8. Using the Bash Shell on Fedora 31

An important part of learning to work with Fedora 31, and Linux distributions in general, involves gaining proficiency in working in the shell environment. While the graphical desktop environments such as GNOME included with Linux provide a user friendly interface to the operating system, in practice the shell environment provides far greater capabilities, flexibility and automation than can ever be achieved using graphical desktop tools. The shell environment also provides a means for interacting with the operating system when a desktop environment is not available; a common occurrence when working with a server-based operating system such as Fedora 31 or a damaged system that will not fully boot.

The goal of this chapter, therefore, is to provide an overview of the default shell environment on Fedora 31 (specifically the Bash shell).

8.1 What is a Shell?

The shell is an interactive command interpreter environment within which commands may be typed at a prompt or entered into a file in the form of a script and executed. The origins of the shell can be traced back to the early days of the UNIX operating system. In fact, in the early days of Linux before the introduction of graphical desktops the shell was the only way for a user to interact with the operating system.

A variety of shell environments have been developed over the years. The first widely used shell was the Bourne shell, written by Stephen Bourne at Bell Labs.

Yet another early creation was the C shell which shared some syntax similarities with the C Programming Language and introduced usability enhancements such as command-line editing and history.

The Korn shell (developed by David Korn at Bell Labs) is based on features provided by both the Bourne shell and the C shell.

The default shell on Fedora 31 is the Bash shell (shorthand for Bourne Again SHell). This shell, which began life as an open source version of the Bourne shell, was developed for the GNU Project by Brian Fox and is based on features provided by both the Bourne shell and the C shell.

8.2 Gaining Access to the Shell

From within the GNOME desktop environment, the shell prompt may be accessed from a Terminal window by selecting the Activities option in the top bar, entering Terminal into the search bar and clicking on the Terminal icon.

When remotely logging into a Fedora 31 server, for example using SSH, the user is also presented with a shell prompt. Details on accessing a remote server using SSH will be covered in the chapter

Using the Bash Shell on Fedora 31

entitled “*Configuring SSH Key-based Authentication on Fedora 31*”. When booting a server-based system in which a desktop environment has not been installed, the shell is entered immediately after the user completes the login procedure at the physical console terminal or remote login session.

8.3 Entering Commands at the Prompt

Commands are entered at the shell command prompt simply by typing the command and pressing the Enter key. While some commands perform tasks silently, most will display some form of output before returning to the prompt. For example, the `ls` command can be used to display the files and directories in the current working directory:

```
$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
```

The available commands are either built into the shell itself, or reside on the physical file system. The location on the file system of a command may be identified using the `which` command. For example, to find out where the `ls` executable resides on the file system:

```
$ which ls
alias ls='ls --color=auto'
      /usr/bin/ls
```

Clearly the `ls` command resides in the `/usr/bin` directory. Note also that an alias is configured, a topic which will be covered later in this chapter. Using the `which` command to locate the path to commands that are built into the shell will result in a message indicating the executable cannot be found. For example, attempting to find the location of the `history` command (which is actually built into the shell rather than existing as an executable on the file system) will result in output similar to the following:

```
$ which history
/usr/bin/which: no history in (/home/demo/.local/bin:/home/demo/bin:/usr/share/
Modules/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin)
```

8.4 Getting Information about a Command

Many of the commands available to the Linux shell can seem cryptic to begin with. To find out detailed information about what a command does and how to use it, use the `man` command specifying the name of the command as an argument. For example, to learn more about the `pwd` command:

```
$ man pwd
```

When the above command is executed, a detailed description of the `pwd` command will be displayed. Many commands will also provide additional information when run with the `--help` command-line option:

```
$ wc --help
```

8.5 Bash Command-line Editing

Early shell environments did not provide any form of line editing capabilities. This meant that if you spotted an error at the beginning of a long command-line you were typing, you had to delete

all the following characters, correct the error and then re-enter the remainder of the command. Fortunately Bash provides a wide range of command-line editing options as outlined in the following table:

Key Sequence	Action
Ctrl-b or Left Arrow	Move cursor back one position
Ctrl-f or Right Arrow	Move cursor forward one position
Delete	Delete character currently beneath the cursor
Backspace	Delete character to the left of the cursor
Ctrl-_	Undo previous change (can be repeated to undo all previous changes)
Ctrl-a	Move cursor to the start of the line
Ctrl-e	Move cursor to the end of the line
Meta-f or Esc then f	Move cursor forward one word
Meta-b or Esc then b	Move cursor back one word
Ctrl-l	Clear the screen of everything except current command
Ctrl-k	Delete to end of line from current cursor position
Meta-d or Esc then d	Delete to end of current word
Meta-DEL or Esc then DEL	Delete beginning to current word
Ctrl-w	Delete from current cursor position to previous white space

Table 8-1

8.6 Working with the Shell History

In addition to command-line editing features, the Bash shell also provides command-line history support. A list of previously executed commands may be viewed using the *history* command:

```
$ history
 1  ps
 2  ls
 3  ls -l /
 4  ls
 5  man pwd
 6  man apropos
```

In addition, Ctrl-p (or up arrow) and Ctrl-n (or down arrow) may be used to scroll back and forth through previously entered commands. When the desired command from the history is displayed, press the Enter key to execute it.

Another option is to enter the ‘!’ character followed by the first few characters of the command to be repeated followed by the Enter key.

8.7 Filename Shorthand

Many shell commands take one or more filenames as arguments. For example, to display the content of a text file named *list.txt*, the *cat* command would be used as follows:

```
$ cat list.txt
```

Similarly, the content of multiple text files could be displayed by specifying all the file names as arguments:

```
$ cat list.txt list2.txt list3.txt list4.txt
```

Instead of typing in each name, pattern matching can be used to specify all files with names matching certain criteria. For example, the ‘*’ wildcard character can be used to simplify the above example:

```
$ cat *.txt
```

The above command will display the content of all files ending with a *.txt* extension. This could be further restricted to any file names beginning with *list* and ending in *.txt*:

```
$ cat list*.txt
```

Single character matches may be specified using the ‘?’ character:

```
$ cat list?.txt
```

8.8 Filename and Path Completion

Rather than typing in an entire file name or path, or using pattern matching to reduce the amount of typing, the shell provides the *filename completion* feature. In order to use filename completion, simply enter the first few characters of the file or path name and then press the Esc key twice. The shell will then complete the filename for you with the first file or path name in the directory that matches the characters you entered. To obtain a list of possible matches, press Esc = after entering the first few characters.

8.9 Input and Output Redirection

As previously mentioned, many shell commands output information when executed. By default this output goes to a device file called *stdout* which is essentially the terminal window or console in which the shell is running. Conversely, the shell takes input from a device file named *stdin*, which by default is the keyboard.

Output from a command can be redirected from *stdout* to a physical file on the file system using the ‘>’ character. For example, to redirect the output from an *ls* command to a file named *files.txt*, the following command would be required:

```
$ ls *.txt > files.txt
```

Upon completion, *files.txt* will contain the list of files in the current directory. Similarly, the contents of a file may be fed into a command in place of *stdin*. For example, to redirect the contents of a file as input to a command:

```
$ wc -l < files.txt
```

The above command will display the number of lines contained in the *files.txt* file.

It is important to note that the ‘>’ redirection operator creates a new file, or truncates an existing file when used. In order to append to an existing file, use the ‘>>’ operator:

```
$ ls *.dat >> files.txt
```

In addition to standard output, the shell also provides standard error output using *stderr*. While output from a command is directed to *stdout*, any error messages generated by the command are directed to *stderr*. This means that if *stdout* is directed to a file, error messages will still appear in the terminal. This is generally the desired behavior, though *stderr* may also be redirected if desired using the ‘2>’ operator:

```
$ ls dkjfnvkjdnf 2> errormsg
```

On completion of the command, an error reporting the fact that the file named *dkjfnvkjdnf* could not be found will be contained in the *errormsg* file.

Both *stderr* and *stdout* may be redirected to the same file using the *&>* operator:

```
$ ls /etc dkjfnvkjdnf &> alloutput
```

On completion of execution, the *alloutput* file will contain both a listing of the contents of the */etc* directory, and the error message associated with the attempt to list a non-existent file.

8.10 Working with Pipes in the Bash Shell

In addition to I/O redirection, the shell also allows output from one command to be piped directly as input to another command. A pipe operation is achieved by placing the ‘|’ character between two or more commands on a command-line. For example, to count the number of processes running on a system, the output from the *ps* command can be piped through to the *wc* command:

```
$ ps -ef | wc -l
```

There is no limit to the number of pipe operations that can be performed on a command-line. For example, to find the number of lines in a file which contain the name Smith:

```
$ cat namesfile | grep Smith | wc -l
```

8.11 Configuring Aliases

As you gain proficiency with the shell environment it is likely that you will find yourself frequently issuing commands with the same arguments. For example, you may often use the *ls* command with the *l* and *t* options:

```
$ ls -lt
```

To reduce the amount of typing involved in issuing a command, it is possible to create an alias that maps to the command and arguments. For example, to create an alias such that entering the letter *l* will cause the *ls -lt* command to be executed, the following statement would be used:

```
$ alias l="ls -lt"
```

Entering *l* at the command prompt will now execute the original statement.

Using the Bash Shell on Fedora 31

8.12 Environment Variables

Shell environment variables provide temporary storage of data and configuration settings. The shell itself sets up a number of environment variables that may be changed by the user to modify the behavior of the shell. A listing of currently defined variables may be obtained using the *env* command:

```
$ env
SSH_CONNECTION=192.168.0.19 61231 192.168.0.28 22
MODULES_RUN_QUARANTINE=LD_LIBRARY_PATH
LANG=en_US.UTF-8
HISTCONTROL=ignoredups
HOSTNAME=CentOSFedora-pc.ebookfrenzy.com
XDG_SESSION_ID=15
MODULES_CMD=/usr/share/Modules/libexec/modulecmd.tcl
USER=demo
ENV=/usr/share/Modules/init/profile.sh
SELINUX_ROLE_REQUESTED=
PWD=/home/demo
HOME=/home/demo
SSH_CLIENT=192.168.0.19 61231 22
SELINUX_LEVEL_REQUESTED=
.
.
.
```

Perhaps the most useful environment variable is *PATH*. This defines the directories in which the shell will search for commands entered at the command prompt, and the order in which it will do so. The *PATH* environment variable for a user account on a newly installed Fedora 31 system will likely be configured as follows:

```
$ echo $PATH
/home/demo/.local/bin:/home/demo/bin:/usr/share/Modules/bin:/usr/local/bin:/usr/
bin:/usr/local/sbin:/usr/sbin
```

Another useful variable is *HOME* which specifies the home directory of the current user. If, for example, you wanted the shell to also look for commands in the *scripts* directory located in your home directory, you would modify the *PATH* variable as follows:

```
$ export PATH=$PATH:$HOME/scripts
```

The current value of an existing environment variable may be displayed using the *echo* command:

```
$ echo $PATH
```

You can create your own environment variables using the *export* command. For example:

```
$ export DATAPATH=/data/files
```

A useful trick to assign the output from a command to an environment variable involves the use of back quotes (```) around the command. For example, to assign the current date and time to an

environment variable called NOW:

```
$ export NOW=`date`
$ echo $NOW
Tue Apr 2 13:48:40 EDT 2019
```

If there are environment variable or alias settings that you need to be configured each time you enter the shell environment, they may be added to a file in your home directory named *.bashrc*. For example, the following *.bashrc* file is configured to set up the DATAPATH environment variable and an alias:

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
    . /etc/bashrc
fi

# User specific environment
PATH="$HOME/.local/bin:$HOME/bin:$PATH"
export PATH

# Uncomment the following line if you don't like systemctl's auto-paging feature:
# export SYSTEMD_PAGER=

# User specific aliases and functions
export DATAPATH=/data/files
alias l="ls -lt"
```

8.13 Writing Shell Scripts

So far we have focused exclusively on the interactive nature of the Bash shell. By interactive we mean manually entering commands at the prompt one by one and executing them. In fact, this is only a small part of what the shell is capable of. Arguably one of the most powerful aspects of the shell involves the ability to create shell scripts. Shell scripts are essentially text files containing sequences of statements that can be executed within the shell environment to perform tasks. In addition to the ability to execute commands, the shell provides many of the programming constructs such as *for* and *do* loops and *if* statements that you might reasonably expect to find in a scripting language.

Unfortunately a detailed overview of shell scripting is beyond the scope of this chapter. There are, however, many books and web resources dedicated to shell scripting that do the subject much more justice than we could ever hope to achieve here. In this section, therefore, we will only be providing a very small taste of shell scripting.

The first step in creating a shell script is to create a file (for the purposes of this example we name it *simple.sh*) and add the following as the first line:

Using the Bash Shell on Fedora 31

```
#!/bin/sh
```

The `#!` is called the “shebang” and is a special sequence of characters indicating that the path to the interpreter needed to execute the script is the next item on the line (in this case the *sh* executable located in `/bin`). This could equally be, for example, `/bin/csh` or `/bin/ksh` if either were the interpreter you wanted to use.

The next step is to write a simple script:

```
#!/bin/sh
for i in *
do
    echo $i
done
```

All this script does is iterate through all the files in the current directory and display the name of each file. This may be executed by passing the name of the script through as an argument to *sh*:

```
$ sh simple.sh
```

In order to make the file executable (thereby negating the need to pass it through to the *sh* command) the *chmod* command can be used:

```
$ chmod +x simple.sh
```

Once the execute bit has been set on the file’s permissions, it may be executed directly. For example:

```
$ ./simple.sh
```

8.14 Summary

In this chapter of Fedora 31 Essentials we have taken a brief tour of the Bash shell environment. In the world of graphical desktop environments it is easy to forget that the true power and flexibility of an operating system can often only be utilized by dropping down below the user friendly desktop interface and using a shell environment. Moreover, familiarity with the shell is a necessity when required to administer and maintain server-based systems that do not have the desktop installed or when attempting to repair a system that is damaged to the point that the desktop or Cockpit interface will no longer launch.

The capabilities of the shell go far beyond the areas covered in this chapter. If you are new to the shell then we strongly encourage you to seek out additional resources. Once familiar with the concepts you will quickly find that it is quicker to perform many tasks using the shell in a terminal window than it is to wade through menus and dialogs on the desktop.