

Rocky Linux 9 Essentials

Rocky Linux 9 Essentials

© 2023 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: 1.0

Table of Contents

1. Introduction

- 1.1 Superuser Conventions
- 1.2 Opening a Terminal Window
- 1.3 Editing Files
- 1.4 Feedback
- 1.5 Errata

2. A Brief History of Rocky 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 Stream - The Free Alternative
- 2.9 Rocky Linux
- 2.10 Summary

3. Installing Rocky Linux 9 on a Clean Disk Drive

- 3.1 Obtaining the Rocky Linux Installation Media
- 3.2 Writing the ISO Installation Image to a USB Drive
 - 3.2.1 Linux
 - 3.2.2 macOS
 - 3.2.3 Windows/macOS
- 3.3 Installing Rocky Linux 9
- 3.4 Partitioning a Disk for Rocky Linux 9
- 3.5 Disk Encryption
- 3.6 User Settings
- 3.7 The Physical Installation
- 3.8 Final Configuration Steps
- 3.9 Installing Updates
- 3.10 Displaying Boot Messages
- 3.11 Summary

4. Dual Booting Rocky Linux 9 with Windows

- 4.1 Partition Resizing
- 4.2 Changing the Default Boot Option
- 4.3 Accessing the Windows Partition from Rocky 9

Table of Contents

4.4 Summary

5. Allocating Windows Disk Partitions to Rocky Linux 9

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 Summary

6. A Guided Tour of the GNOME 40 Desktop

6.1 Installing the GNOME Desktop

6.2 An Overview of the GNOME 40 Desktop

6.3 Activity Overview

6.4 Managing Windows

6.5 Using Workspaces

6.6 Calendar and Notifications

6.7 GNOME Desktop Settings

6.8 Beyond Basic Customization

6.9 Installing GNOME Desktop Apps

6.10 Summary

7. An Overview of the Cockpit Web Interface

7.1 An Overview of Cockpit

7.2 Installing and Enabling Cockpit

7.3 Accessing Cockpit

7.4 Overview

7.5 Logs

7.6 Storage

7.7 Networking

7.8 Accounts

7.9 Services

7.10 Applications

7.11 Virtual Machines

7.12 Software Updates

7.13 Terminal

7.14 Connecting to Multiple Servers

7.15 Enabling Stored Metrics

7.16 Summary

8. Using the Bash Shell on Rocky Linux 9

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 Rocky Linux 9 Users and Groups
9.1	User Management from the Command-line
9.2	User Management with Cockpit
9.3	User Management using the Settings App
9.4	Summary
10.	Rocky Linux 9 Software Installation and AppStreams
10.1	Repositories
10.2	The Repository
10.3	The AppStream Repository
10.4	Summary
11.	Managing Rocky Linux 9 systemd Units
11.1	Understanding Rocky 9 systemd Targets
11.2	Understanding Rocky 9 systemd Services
11.3	Rocky Linux 9 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.	Rocky Linux 9 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.	Rocky Linux 9 Firewall Basics

Table of Contents

- 13.1 Understanding Ports and Services
- 13.2 Securing Ports and Services
- 13.3 Rocky Linux 9 Services and iptables Rules
- 13.4 Well-Known Ports and Services
- 13.5 Summary

14. Rocky Linux 9 Firewall Configuration with firewalld

- 14.1 An Introduction to firewalld
 - 14.1.1 Zones
 - 14.1.2 Interfaces
 - 14.1.3 Services
 - 14.1.4 Ports
- 14.2 Checking firewalld Status
- 14.3 Configuring Firewall Rules with firewall-cmd
 - 14.3.1 Identifying and Changing the Default Zone
 - 14.3.2 Displaying Zone Information
 - 14.3.3 Adding and Removing Zone Services
 - 14.3.4 Working with Port-based Rules
 - 14.3.5 Creating a New Zone
 - 14.3.6 Changing Zone/Interface Assignments
 - 14.3.7 Masquerading
 - 14.3.8 Adding ICMP Rules
 - 14.3.9 Implementing Port Forwarding
- 14.4 Managing firewalld from the Cockpit Interface
- 14.5 Managing firewalld using firewall-config
- 14.6 Summary

15. Configuring SSH Key-based Authentication on Rocky Linux 9

- 15.1 An Overview of Secure Shell (SSH)
- 15.2 SSH Key-based Authentication
- 15.3 Setting Up Key-based Authentication
- 15.4 Installing and Starting the SSH Service
- 15.5 SSH Key-based Authentication from Linux and macOS Clients
- 15.6 Managing Multiple Keys
- 15.7 SSH Key-based Authentication from Windows Clients
- 15.8 SSH Key-based Authentication using PuTTY
- 15.9 Generating a Private Key with PuTTYgen
- 15.10 Summary

16. Rocky Linux 9 Remote Desktop Access with VNC

- 16.1 Secure and Insecure Remote Desktop Access
- 16.2 Installing the GNOME Desktop Environment
- 16.3 Installing VNC on Rocky Linux 9

- 16.4 Configuring the VNC Server
- 16.5 Connecting to a VNC Server
- 16.6 Establishing a Secure Remote Desktop Session
- 16.7 Establishing a Secure Tunnel on Windows using PuTTY
- 16.8 Shutting Down a Desktop Session
- 16.9 Troubleshooting a VNC Connection
- 16.10 Summary
- 17. Displaying Rocky Linux 9 Applications Remotely (X11 Forwarding)**
 - 17.1 Requirements for Remotely Displaying Rocky Linux 9 Applications
 - 17.2 Displaying a Rocky Linux 9 Application Remotely
 - 17.3 Trusted X11 Forwarding
 - 17.4 Compressed X11 Forwarding
 - 17.5 Displaying Remote Rocky Linux 9 Apps on Windows
 - 17.6 Summary
- 18. Using NFS on Rocky Linux 9 to Share Files with Remote Systems**
 - 18.1 Ensuring NFS Services are running on Rocky Linux 9
 - 18.2 Configuring the Rocky Linux 9 Firewall to Allow NFS Traffic
 - 18.3 Specifying the Folders to be Shared
 - 18.4 Accessing Shared Folders
 - 18.5 Mounting an NFS Filesystem on System Startup
 - 18.6 Unmounting an NFS Mount Point
 - 18.7 Accessing NFS Filesystems in Cockpit
 - 18.8 Summary
- 19. Sharing Files between Rocky Linux 9 and Windows Systems with Samba**
 - 19.1 Accessing Windows Resources from the GNOME Desktop
 - 19.2 Samba and Samba Client
 - 19.3 Installing Samba on Rocky 9
 - 19.4 Configuring the Rocky 9 Firewall to Enable Samba
 - 19.5 Configuring the smb.conf File
 - 19.5.1 Configuring the [global] Section
 - 19.5.2 Configuring a Shared Resource
 - 19.5.3 Removing Unnecessary Shares
 - 19.6 Configuring SELinux for Samba
 - 19.7 Creating a Samba User
 - 19.8 Testing the smb.conf File
 - 19.9 Starting the Samba and NetBIOS Name Services
 - 19.10 Accessing Samba Shares
 - 19.11 Accessing Windows Shares from Rocky 9
 - 19.12 Summary
- 20. An Overview of Virtualization Techniques**

Table of Contents

- 20.1 Guest Operating System Virtualization
- 20.2 Hypervisor Virtualization
 - 20.2.1 Paravirtualization
 - 20.2.2 Full Virtualization
 - 20.2.3 Hardware Virtualization
- 20.3 Virtual Machine Networking
- 20.4 Summary

21. Installing KVM Virtualization on Rocky Linux 9

- 21.1 An Overview of KVM
- 21.2 KVM Hardware Requirements
- 21.3 Preparing Rocky 9 for KVM Virtualization
- 21.4 Verifying the KVM Installation
- 21.5 Summary

22. Creating KVM Virtual Machines on Rocky Linux 9 using Cockpit

- 22.1 Installing the Cockpit Virtual Machines Module
- 22.2 Creating a Virtual Machine in Cockpit
- 22.3 Starting the Installation
- 22.4 Working with Storage Volumes and Storage Pools
- 22.5 Summary

23. Creating KVM Virtual Machines on Rocky Linux 9 using virt-manager

- 23.1 Starting the Virtual Machine Manager
- 23.2 Configuring the KVM Virtual System
- 23.3 Starting the KVM Virtual Machine
- 23.4 Summary

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

- 24.1 Running virt-install to build a KVM Guest System
- 24.2 An Example Rocky Linux 9 virt-install Command
- 24.3 Starting and Stopping a Virtual Machine from the Command-Line
- 24.4 Creating a Virtual Machine from a Configuration File
- 24.5 Summary

25. Creating a Rocky Linux 9 KVM Networked Bridge Interface

- 25.1 Getting the Current Network Manager Settings
- 25.2 Creating a Network Manager Bridge from the Command-Line
- 25.3 Declaring the KVM Bridged Network
- 25.4 Using a Bridge Network in a Virtual Machine
- 25.5 Creating a Bridge Network using nm-connection-editor
- 25.6 Summary

26. Managing KVM using the virsh Command-Line Tool

- 26.1 The virsh Shell and Command-Line

- 26.2 Listing Guest System Status
- 26.3 Starting a Guest System
- 26.4 Shutting Down a Guest System
- 26.5 Suspending and Resuming a Guest System
- 26.6 Saving and Restoring Guest Systems
- 26.7 Rebooting a Guest System
- 26.8 Configuring the Memory Assigned to a Guest OS
- 26.9 Summary

27. An Introduction to Linux Containers

- 27.1 Linux Containers and Kernel Sharing
- 27.2 Container Uses and Advantages
- 27.3 Rocky Linux 9 Container Tools
- 27.4 The Docker Registry
- 27.5 Container Networking
- 27.6 Summary

28. Working with Containers on Rocky Linux 9

- 28.1 Installing the Container Tools
- 28.2 Pulling a Rocky 9 Container Image
- 28.3 Running the Image in a Container
- 28.4 Managing a Container
- 28.5 Saving a Container to an Image
- 28.6 Removing an Image from Local Storage
- 28.7 Removing Containers
- 28.8 Building a Container with Buildah
- 28.9 Building a Container from Scratch
- 28.10 Container Bridge Networking
- 28.11 Managing Containers in Cockpit
- 28.12 Summary

29. Setting Up a Rocky Linux 9 Web Server

- 29.1 Requirements for Configuring a Rocky 9 Web Server
- 29.2 Installing the Apache Web Server Packages
- 29.3 Configuring the Firewall
- 29.4 Port Forwarding
- 29.5 Starting the Apache Web Server
- 29.6 Testing the Web Server
- 29.7 Configuring the Apache Web Server for Your Domain
- 29.8 The Basics of a Secure Website
- 29.9 Configuring Apache for HTTPS
- 29.10 Obtaining an SSL Certificate
- 29.11 Summary

Table of Contents

30. Configuring a Rocky Linux 9 Postfix Email Server

- 30.1 The Structure of the Email System
 - 30.1.1 Mail User Agent
 - 30.1.2 Mail Transfer Agent
 - 30.1.3 Mail Delivery Agent
 - 30.1.4 SMTP
 - 30.1.5 SMTP Relay
- 30.2 Configuring a Rocky Linux 9 Email Server
- 30.3 Postfix Pre-Installation Steps
- 30.4 Firewall/Router Configuration
- 30.5 Installing Postfix on Rocky Linux 9
- 30.6 Configuring Postfix
- 30.7 Configuring DNS MX Records
- 30.8 Starting Postfix on a Rocky Linux 9 System
- 30.9 Testing Postfix
- 30.10 Sending Mail via an SMTP Relay Server
- 30.11 Summary

31. Adding a New Disk Drive to a Rocky Linux 9 System

- 31.1 Mounted File Systems or Logical Volumes
- 31.2 Finding the New Hard Drive
- 31.3 Creating Linux Partitions
- 31.4 Creating a File System on a Rocky Linux 9 Disk Partition
- 31.5 An Overview of Journaled File Systems
- 31.6 Mounting a File System
- 31.7 Configuring Rocky Linux 9 to Mount a File System Automatically
- 31.8 Adding a Disk Using Cockpit
- 31.9 Summary

32. Adding a New Disk to a Rocky Linux 9 Volume Group and Logical Volume

- 32.1 An Overview of Logical Volume Management (LVM)
 - 32.1.1 Volume Group (VG)
 - 32.1.2 Physical Volume (PV)
 - 32.1.3 Logical Volume (LV)
 - 32.1.4 Physical Extent (PE)
 - 32.1.5 Logical Extent (LE)
- 32.2 Getting Information about Logical Volumes
- 32.3 Adding Additional Space to a Volume Group from the Command-Line
- 32.4 Adding Additional Space to a Volume Group Using Cockpit
- 32.5 Summary

33. Adding and Managing Rocky Linux 9 Swap Space

- 33.1 What is Swap Space?

- 33.2 Recommended Swap Space for Rocky Linux 9
- 33.3 Identifying Current Swap Space Usage
- 33.4 Adding a Swap File to a Rocky Linux 9 System
- 33.5 Adding Swap as a Partition
- 33.6 Adding Space to a Rocky Linux 9 LVM Swap Volume
- 33.7 Adding Swap Space to the Volume Group
- 33.8 Summary

34. Rocky Linux 9 System and Process Monitoring

- 34.1 Managing Processes
- 34.2 Real-time System Monitoring with top
- 34.3 Command-Line Disk and Swap Space Monitoring
- 34.4 Summary

Index

1. Introduction

Rocky Linux 9 Essentials is designed to provide detailed information on the installation, use, and administration of the Rocky Linux 9 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. 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

Rocky Linux 9, 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:

```
[demo@demo-server ~]$ su -  
Password:  
[demo@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.

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

```
[demo@demo-server ~]$ dnf update  
Error: This command has to be run with superuser privileges (under the root user  
on most systems).
```

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*:

Introduction

```
[demo@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 demo:
```

```
.  
.
```

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

```
[demo@demo-server]$ sudo su -  
[demo@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 Opening a Terminal Window

If you are using the GNOME desktop and need to access a command prompt you will need to open a Terminal window. To do this, either press the keyboard Windows key or click on the Activities button in the top left-hand corner of the screen, then select the Terminal from the dash as shown in Figure 1-1:

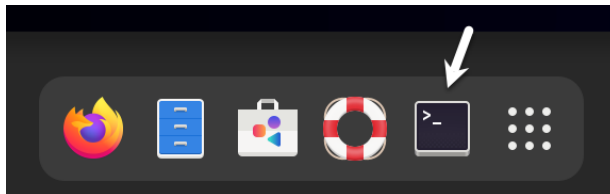


Figure 1-1

1.3 Editing Files

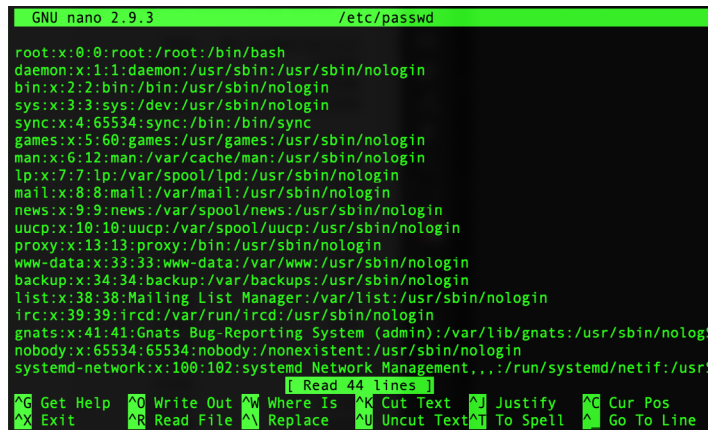
Configuring a Linux system typically involves editing files. For those new to Linux it can be unclear which editor to use. If you are running a terminal session and do not already have a preferred editor we recommend using the *nano* editor. To launch *nano* in a terminal window simply enter the following command:

```
# nano <file>
```

Where <file> is replaced by the path to the file you wish to edit. For example:

```
# nano /etc/passwd
```

Once loaded, *nano* will appear as illustrated in Figure 1-2:



```
GNU nano 2.9.3 /etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nolog$
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,:/run/systemd/netif:/usr$

[ Read 44 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^M Replace ^U Uncut Text ^T To Spell ^_ Go To Line
```

Figure 1-2

To create a new file simply run *nano* as follows:

```
# nano
```

When you have finished editing the file, type Ctrl-S to save the file followed by Ctrl-X to exit. To open an existing file, use the Ctrl-R keyboard shortcut.

If you prefer to use a graphical editor within the GNOME desktop environment *gedit* is a useful starting point for basic editing tasks. To launch *gedit* from the desktop press Alt-F2 to display the Enter a Command window as shown in Figure 1-3:

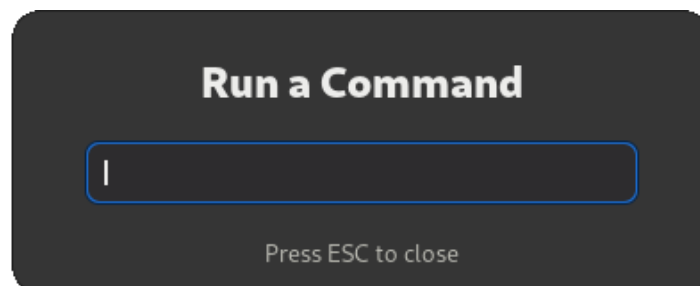
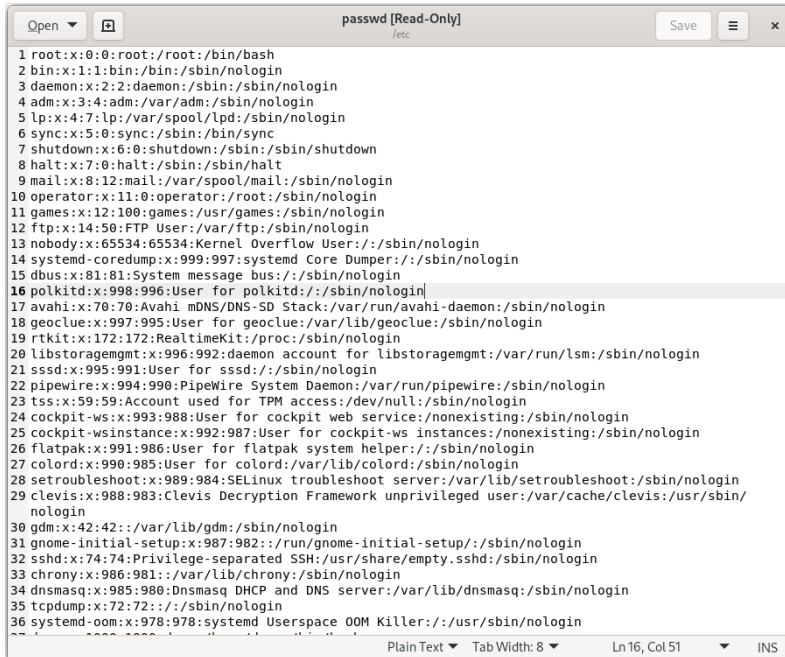


Figure 1-3

Introduction

Enter *gedit* into the text field and press the Enter key. After a short delay, gedit will load ready to open, create and edit files:



```
1 root:x:0:0:root:/root:/bin/bash
2 bin:x:1:1:bin:/bin:/sbin/nologin
3 daemon:x:2:2:daemon:/sbin:/sbin/nologin
4 adm:x:3:4:adm:/var/adm:/sbin/nologin
5 lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
6 sync:x:5:0:sync:/sbin:/bin/sync
7 shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
8 halt:x:7:0:halt:/sbin:/sbin/halt
9 mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
10 operator:x:11:0:operator:/root:/sbin/nologin
11 games:x:12:100:games:/usr/games:/sbin/nologin
12 ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
13 nobody:x:65534:65534:Kernel Overflow User:/sbin/nologin
14 systemd-coredump:x:999:997:systemd Core Dumper:/sbin/nologin
15 dbus:x:81:81:System message bus:/sbin/nologin
16 polkitd:x:998:996:User for polkitd:/sbin/nologin
17 avahi:x:70:70:Avahi mDNS/DNS-SD Stack:/var/run/avahi-daemon:/sbin/nologin
18 geoclue:x:997:995:User for geoclue:/var/lib/geoclue:/sbin/nologin
19 rtkit:x:172:172:RealtimeKit:/proc:/sbin/nologin
20 libstoragemgmt:x:996:992:daemon account for libstoragemgmt:/var/run/lsm:/sbin/nologin
21 sssd:x:995:991:User for sssd:/sbin/nologin
22 pipewire:x:994:990:PipeWire System Daemon:/var/run/pipewire:/sbin/nologin
23 tss:x:59:59:Account used for TPM access:/dev/null:/sbin/nologin
24 cockpit-ws:x:993:988:User for cockpit web service:/nonexisting:/sbin/nologin
25 cockpit-wsinstance:x:992:987:User for cockpit-ws instances:/nonexisting:/sbin/nologin
26 flatpak:x:991:986:User for flatpak system helper:/sbin/nologin
27 colord:x:990:985:User for colord:/var/lib/colord:/sbin/nologin
28 setroubleshoot:x:989:984:SELinux troubleshoot server:/var/lib/setroubleshoot:/sbin/nologin
29 clevis:x:988:983:CLEVIS Decryption Framework unprivileged user:/var/cache/clevis:/usr/sbin/
nologin
30 gdm:x:42:42:/var/lib/gdm:/sbin/nologin
31 gnome-initial-setup:x:987:982:/run/gnome-initial-setup:/sbin/nologin
32 sshd:x:74:74:Privilege-separated SSH:/usr/share/empty.sshd:/sbin/nologin
33 chrony:x:986:981:/var/lib/chrony:/sbin/nologin
34 dnsmasq:x:985:980:Dnsmasq DHCP and DNS server:/var/lib/dnsmasq:/sbin/nologin
35 tcpdump:x:72:72:/sbin/nologin
36 systemd-oom:x:978:978:systemd Userspace OOM Killer:/usr/sbin/nologin
```

Figure 1-4

Alternatively, launch gedit from a terminal window either with or without the path to the file to open:

```
# gedit
# gedit /etc/passwd
```

1.4 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.

1.5 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/rocky9.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 Rocky Linux

Rocky Linux is one of several variants (also referred to as *distributions*) of the Linux operating system. It is based on the source code of the Red Hat Enterprise Linux distribution (RHEL), developed 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 Rocky Linux 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 are many similarities between Linux and Windows). 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 developing a new operating system named Multics. However, 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 and 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

A Brief History of Rocky Linux

communication between the hardware and the applications. The Linux kernel was 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 began 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, the obvious solution was to combine his work with the kernel Linus had written to create a complete 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 refer to Linux as Linux, which 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 acquired 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 could only 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 9 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 Red Hat offers. 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 and the CentOS derivative.

2.8 CentOS Stream - The Free Alternative

For users unable to afford a Red Hat Enterprise Linux subscription, another option is the CentOS Stream operating system. The CentOS project, initially a community-driven effort but now owned by 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. Like Fedora, CentOS Stream field tests new operating system features before they are included in a future RHEL release. As such, it may lack stability but provides access to cutting-edge features.

2.9 Rocky Linux

Rocky Linux was created in 2021 by Greg Kurtzer, a co-founder of the CentOS project. After taking control of the CentOS project, Red Hat changed the philosophy of CentOS Linux from being 100% compatible with Red Hat Enterprise Linux to serving as a platform to test new features before including them in the next RHEL release. The result is an operating system called CentOS Stream that is no longer guaranteed to be as stable and reliable as RHEL. In response to this change, Kurtzer created Rocky Linux based on the original principles of CentOS Linux, namely, to provide an operating system that is 100% compatible with Red Hat Enterprise Linux.

2.10 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. Rocky Linux is essentially built from the same source code base as Red Hat Enterprise Linux, but

A Brief History of Rocky Linux

with the Red Hat branding and subscription requirements removed.

3. Installing Rocky Linux 9 on a Clean Disk Drive

There are now two ways in which a Rocky Linux 9 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 already optimized for the cloud platform and using that as the basis for the Rocky 9 system, there is no need to perform a manual operating system installation in this situation.

If, on the other hand, you plan to install Rocky 9 on your own hardware, the first step on the path to learning about Rocky Linux 9 involves installing the operating system.

Rocky 9 can be installed either in a clean disk environment (where an entire disk is cleared of any existing partitions and dedicated entirely to Rocky 9) or in a dual boot environment where Rocky 9 co-exists with another operating system on the disk (typically a member of the Microsoft Windows family of operating systems).

This chapter will cover the clean disk approach to installation from local or remote installation media. Dual boot installation with a Windows 11 system will be covered in *“Dual Booting Rocky Linux 9 with Windows”*.

3.1 Obtaining the Rocky Linux Installation Media

The Rocky Linux distribution can be downloaded from the project website using the following URL:

<https://rockylinux.org/download>

The installation media is provided in the form of ISO images and is available in the following forms:

- **Boot ISO** - A small image that will boot the system but downloads all of the packages needed for the installation over the internet. Also helpful in rescuing a damaged Rocky installation.
- **Minimal ISO** - This image contains the files necessary to begin the installation and requires access to the full installation media located on a remote server. The boot image can also perform rescue operations on an existing Rocky Linux system.
- **DVD ISO** - A self-contained image, including all of the packages necessary to install a Rocky Linux system

Installing Rocky Linux 9 on a Clean Disk Drive

For the purposes of this book, download the DVD ISO image, which is named using the following convention:

```
Rocky-<version>-<architecture>-dvd.iso
```

For example, the Rocky 9 DVD image for 64-bit Intel systems is named as follows:

```
Rocky-9.1-x86_64-dvd.iso
```

Images are also available for ARM64 (aarch64), ppc64le and s390x architectures.

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.

3.2 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 Rocky 9, 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:

3.2.1 Linux

The first step in writing an ISO image to a USB drive on Linux is identifying 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 system messages, one of which will be a report that the USB drive was detected and will be similar to the following:

```
[ 406.241717] 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
```

This output shows 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) 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 will be ready to be used to install Rocky 9:

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

3.2.2 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
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 rerun the command. The same results should appear with one additional entry for the USB drive resembling the following:

```
/dev/disk3 (external, physical):
```

#:	TYPE	NAME	SIZE	IDENTIFIER
----	------	------	------	------------

Installing Rocky Linux 9 on a Clean Disk Drive

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.

3.2.3 Windows/macOS

Several free tools are available for Windows and macOS 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, insert the destination USB drive, launch the writer tool, and choose the *Select .iso file* option as highlighted in Figure 3-1:

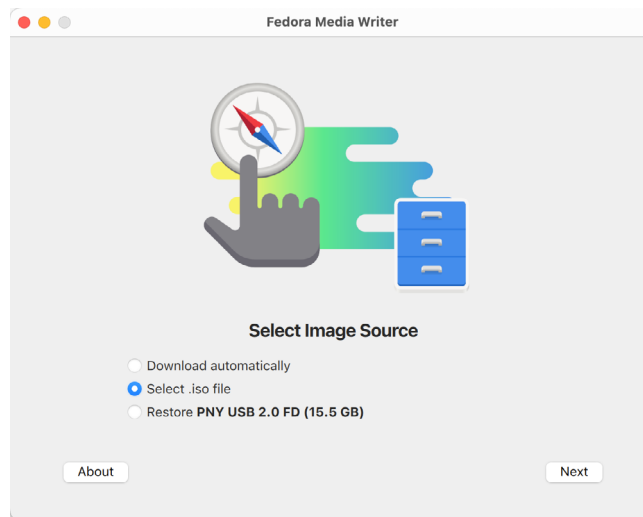


Figure 3-1

Click Next to proceed to the Write Options screen and select the USB Drive before clicking on the *Select...* button:

Installing Rocky Linux 9 on a Clean Disk Drive

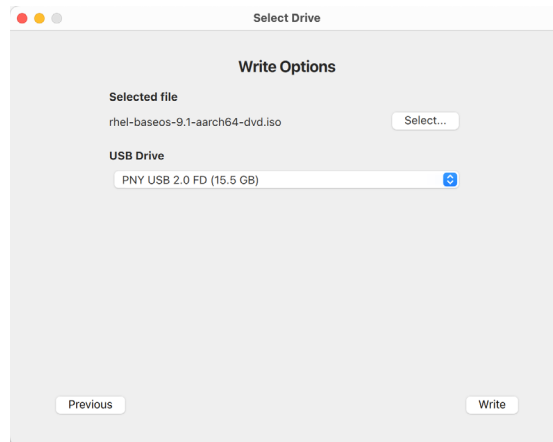


Figure 3-2

In the resulting file selection dialog, navigate to and select the Rocky Linux 9 installation ISO image and click the *Open* button. Finally, click the *Write* button to start writing the image to the USB drive:

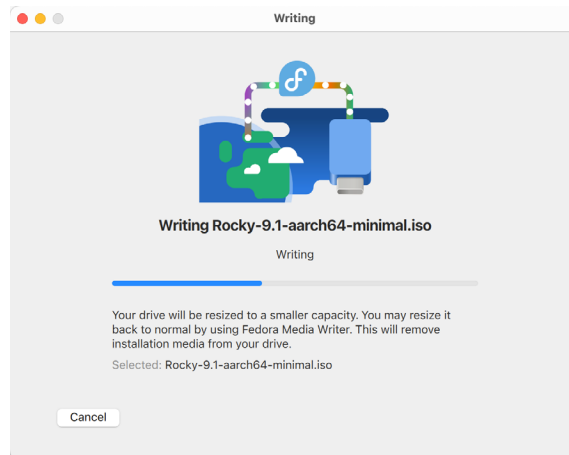


Figure 3-3

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

3.3 Installing Rocky Linux 9

Insert the Rocky 9 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:

Installing Rocky Linux 9 on a Clean Disk Drive

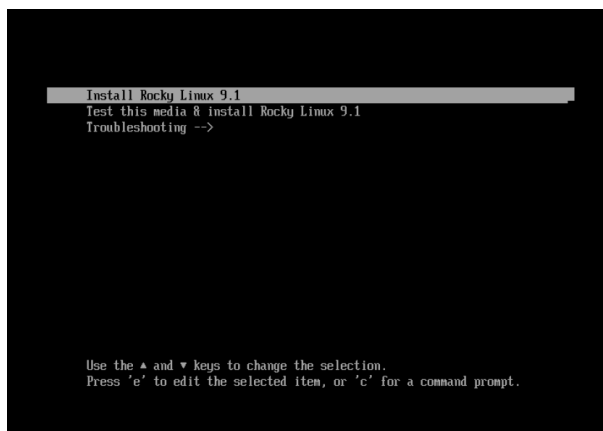


Figure 3-4

Use the arrow keys to navigate between the options and make a selection with the <Enter> key. For example, 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 Rocky 9 system. An option is also available to perform the installation in basic graphics mode for systems without a graphical console:

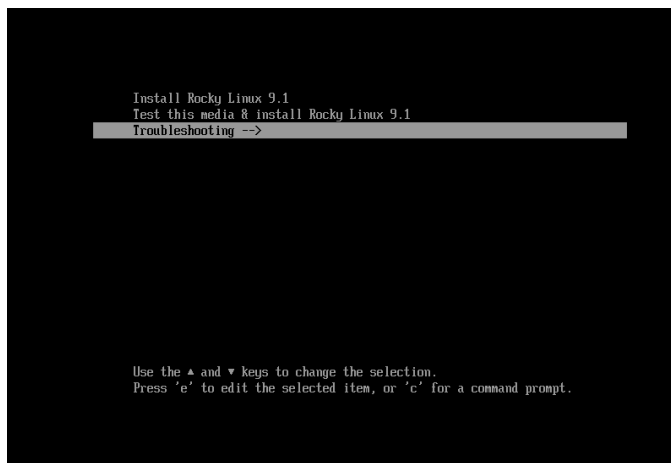


Figure 3-5

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

Installing Rocky Linux 9 on a Clean Disk Drive

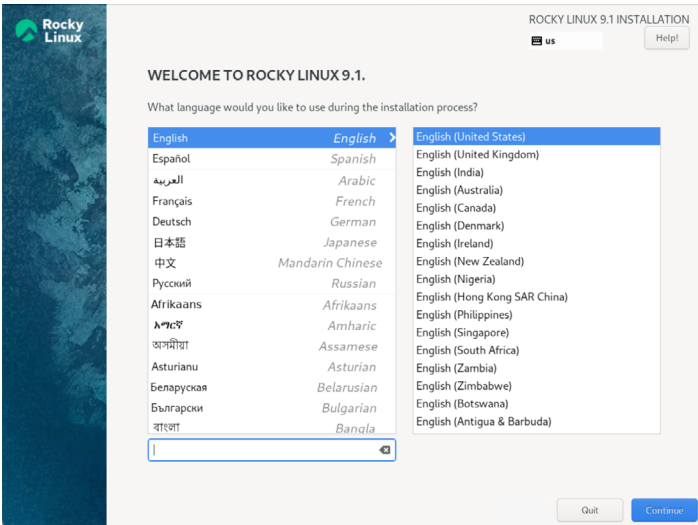


Figure 3-6

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

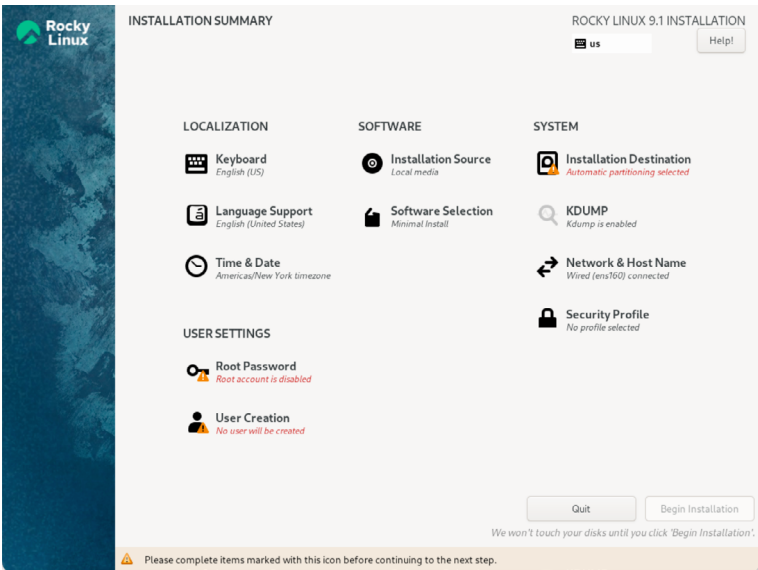


Figure 3-7

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

Installing Rocky Linux 9 on a Clean Disk Drive

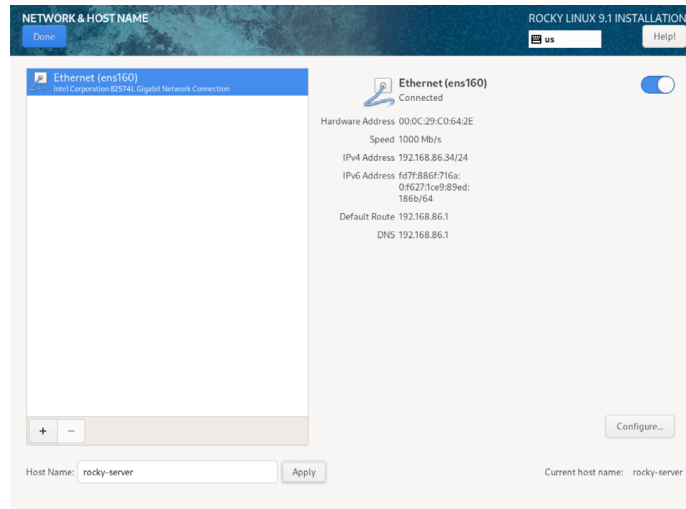


Figure 3-8

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

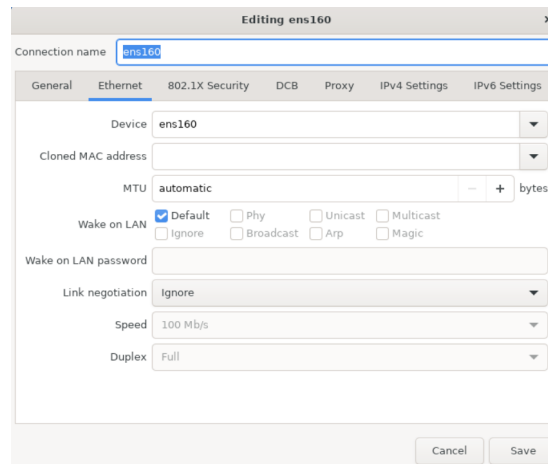


Figure 3-9

Once the hostname 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 choice is also provided to use *Network Time* which automatically synchronizes the system with external Network Time Protocol servers.

Changes to the *Installation Source* settings should not be necessary since the installation is performed from local media. To complete 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 on Rocky Linux 9 to Share 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, the installer will perform a Server with GUI installation of Rocky Linux 9. To select a different installation configuration, choose the *Software Selection* option to display the screen shown below:

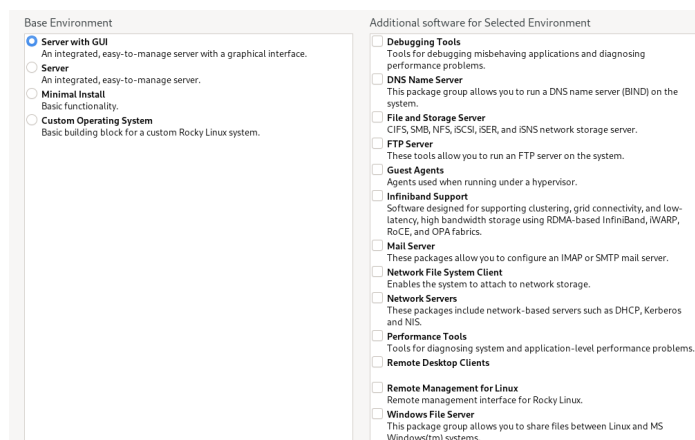


Figure 3-10

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 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 required.

The *Security Policy* option allows additional security packages to be installed on the system that allow security restrictions that comply with the Security Content Automation Protocol (SCAP) standards to be imposed on the system. While selecting a profile from the list installs the necessary packages, the policy restrictions are not enforced until they are enabled on the running system. .

The *Kdump* feature, when enabled, writes out the state of the operating system kernel in the event of a system crash. This dump file can help identify the cause of a crash but takes up system memory when enabled. Unless memory is limited on your system, this can be left in the enabled state.

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

Installing Rocky Linux 9 on a Clean Disk Drive

3.4 Partitioning a Disk for Rocky Linux 9

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

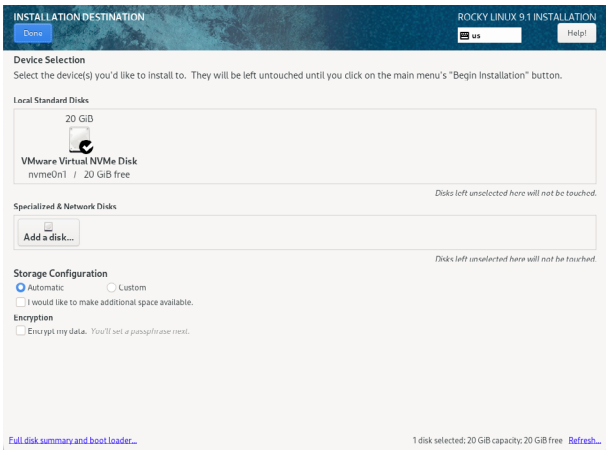


Figure 3-11

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 Rocky 9 has been installed. To remove any existing partitions so that they can be reclaimed and used by Rocky 9, 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:

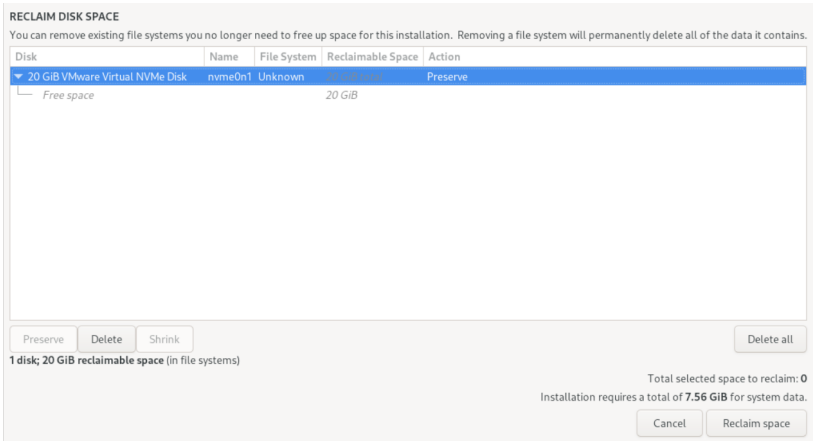


Figure 3-12

To reclaim space, select a partition that is no longer needed and mark it for removal by clicking the *Delete* button. Once all the partitions 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 Rocky 9 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):

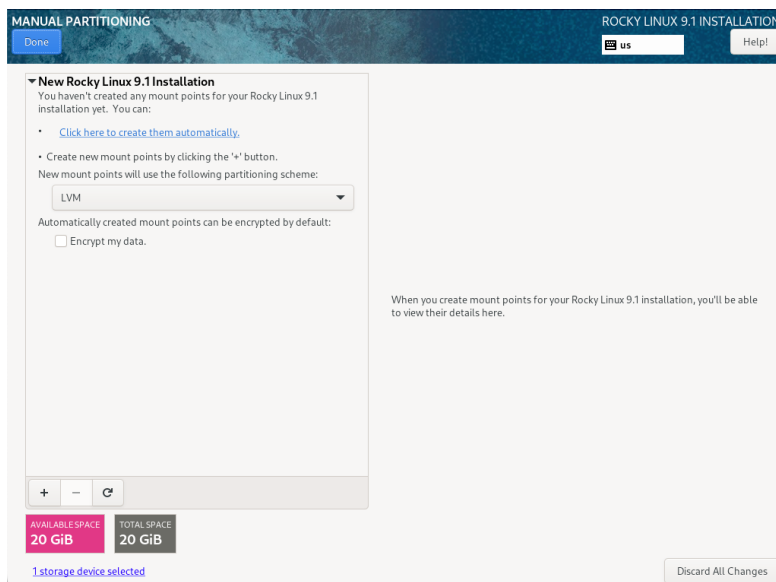


Figure 3-13

The manual partitioning screen allows configuring 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 Rocky Linux 9 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. However, to manually create partitions and allocate mount points, click the + button to declare and assign each partition manually.

Another option is to select automatic partition creation and then use the resulting screen to change the partition configuration as needed manually. 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:

Installing Rocky Linux 9 on a Clean Disk Drive

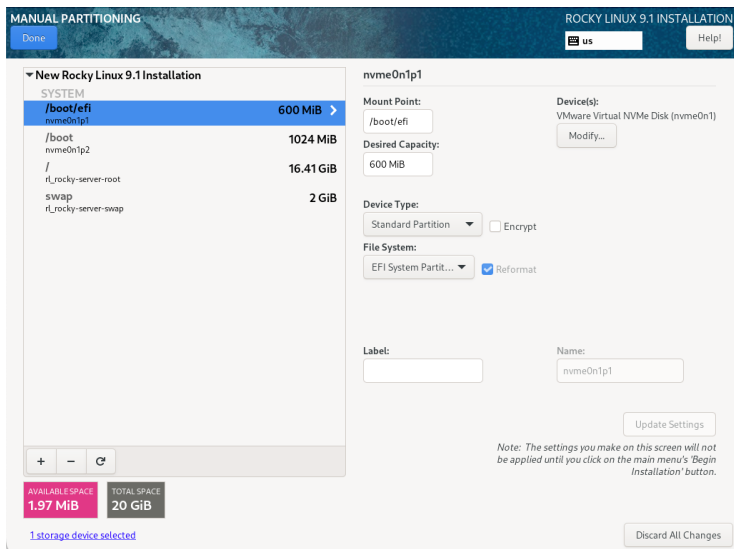


Figure 3-14

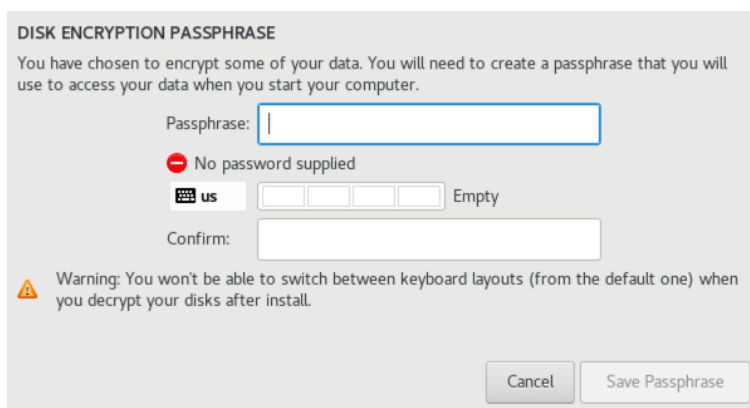
Once the disk has been configured, click on *Done*, review the summary of the changes to be made to the disk (Figure 3-15), followed by the *Accept Changes* button:

SUMMARY OF CHANGES				
Your customizations will result in the following changes taking effect after you return to the main menu and begin installation:				
Order	Action	Type	Device	Mount point
1	destroy format	Unknown	VMware Virtual NVMe Disk (nvme0n1)	
2	create format	partition table (GPT)	VMware Virtual NVMe Disk (nvme0n1)	
3	create device	partition	nvme0n1p1 on VMware Virtual NVMe Disk	
4	create format	EFI System Partition	nvme0n1p1 on VMware Virtual NVMe Disk	/boot/efi
5	create device	partition	nvme0n1p2 on VMware Virtual NVMe Disk	
6	create device	partition	nvme0n1p3 on VMware Virtual NVMe Disk	
7	create format	physical volume (LVM)	nvme0n1p3 on VMware Virtual NVMe Disk	
8	create device	lvmvg	rl_rocky-server	
9	create device	lvm lv	rl_rocky-server-root	
10	create format	xfs	rl_rocky-server-root	/
11	create device	lvm lv	rl_rocky-server-swap	
12	create format	swap	rl_rocky-server-swap	
13	create format	xfs	nvme0n1p2 on VMware Virtual NVMe Disk	/boot

Figure 3-15

3.5 Disk Encryption


Rocky 9 also allows the disk’s contents to be protected by encryption. This will require a passphrase to be entered each time the system boots. To enable encryption, select the *Encrypt my data* option on the main *Installation Destination* screen and, after clicking the *Done* button, enter the passphrase in the dialog shown in Figure 3-16:





DISK ENCRYPTION PASSPHRASE

You have chosen to encrypt some of your data. You will need to create a passphrase that you will use to access your data when you start your computer.

Passphrase:

 No password supplied

  Empty

Confirm:


 Warning: You won't be able to switch between keyboard layouts (from the default one) when you decrypt your disks after install.

Figure 3-16

3.6 User Settings

Before the installation can be completed, you must create a root password and an optional user account. The root, or super-user account, is a special user with administrative privileges on the system. While you will generally use your own account to log into the system, you will need root privileges to configure the system and perform other administrative tasks.

Select the Root Password option and enter a suitably secure password. Options are also available to lock the root account, so it cannot be accessed and to allow access to the root account over remote connections to the system via password authentication. In general, it is recommended to leave the root account unlocked but to disallow remote SSH access using password access. If remote root access is required, it should be implemented using SSH key-based authentication, a topic we will be covering in the “*Configuring SSH Key-based Authentication on Rocky Linux 9*” chapter of this book.

Next, select the User Creation option and enter a user name and password. If you would like this user to be able to perform administrative tasks using the *sudo* command, also enable the *Make this user administrator* checkbox. If this option is not enabled, the user must use the *su* command and enter the root password to gain root privileges (assuming that the option outlined above to lock the root account was not enabled). If the user you add will need to perform administrative tasks, *sudo* is the recommended option, and the checkbox should be enabled.

3.7 The Physical Installation

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

Installing Rocky Linux 9 on a Clean Disk Drive

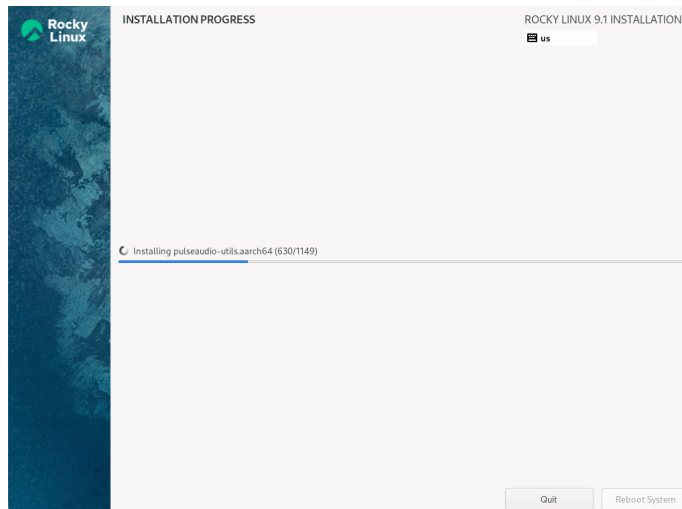


Figure 3-17

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

3.8 Final Configuration Steps

What appears on the console when the system starts will depend on which option was chosen from the Software Selection installation screen. If a GUI option was selected during installation, the GNOME Display Manager (GDM) login screen will appear. On the other hand, 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 created during the installation process's final steps.

In the case of a GUI installation, the GNOME initial setup tool will launch and offer a guided tour of the desktop environment.

3.9 Installing Updates

As with most operating systems today, each release of a Rocky Linux distribution continues to evolve after it has been released to the public. This generally takes the form of bug fixes, 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 Rocky 9 is to make ensure any available updates 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, run the following command:

```
# dnf check-update
```

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

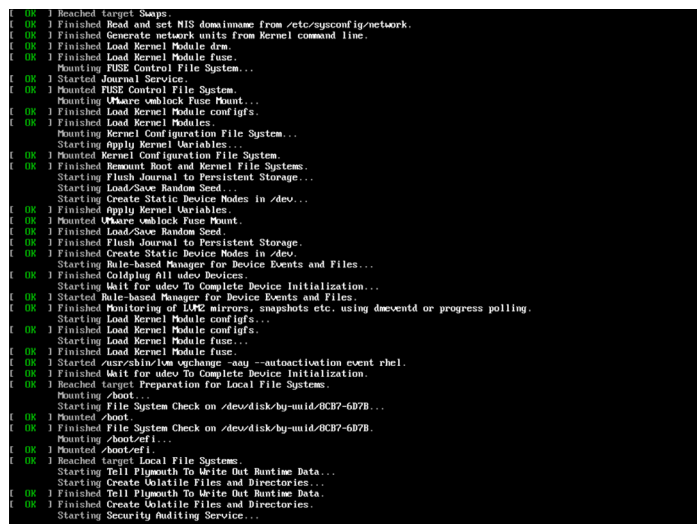
```
# dnf update
```

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

Once the update is complete, the installation is essentially finished, and Rocky 9 is ready for use.

3.10 Displaying Boot Messages

During the boot process, Rocky 9 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), press the keyboard Esc key while the system is starting:



```

[ OK ] Reached target Swags.
[ OK ] Finished Read and set NIS domainname from /etc/sysconfig/network.
[ OK ] Finished Generate network units from Kernel command line.
[ OK ] Finished Load Kernel Module dracut.
[ OK ] Finished Load Kernel Module fuse.
[ OK ] Mounting FUSE Control File System...
[ OK ] Started Journal Service.
[ OK ] Mounted FUSE Control File System.
[ OK ] Mounting Uvare umblock Fuse Mount...
[ OK ] Finished Load Kernel Module configs.
[ OK ] Finished Load Kernel Modules.
[ OK ] Mounting Kernel Configuration File System...
[ OK ] Starting Apply Kernel Variables...
[ OK ] Mounted Kernel Configuration File System.
[ OK ] Finished Remount Root and Kernel File System.
[ OK ] Starting Flush Journal to Persistent Storage...
[ OK ] Starting Load/Save Random Seed...
[ OK ] Starting Create Static Device Nodes in /dev...
[ OK ] Finished Apply Kernel Variables.
[ OK ] Mounted Uvare umblock Fuse Mount.
[ OK ] Finished Load/Save Random Seed.
[ OK ] Finished Flush Journal to Persistent Storage.
[ OK ] Finished Create Static Device Nodes in /dev.
[ OK ] Starting Rule-based Manager for Device Events and Files...
[ OK ] Finished Coldplug All udev Devices.
[ OK ] Starting Wait for udev To Complete Device Initialization...
[ OK ] Started Rule-based Manager for Device Events and Files.
[ OK ] Finished Monitoring of LVM2 mirrors, snapshots etc. using dmeventd or progress polling.
[ OK ] Starting Load Kernel Module configs.
[ OK ] Finished Load Kernel Module configs.
[ OK ] Starting Load Kernel Module fuse.
[ OK ] Finished Load Kernel Module fuse.
[ OK ] Started /usr/sbin/udevadm settle --autoactivation=auto rhel.
[ OK ] Finished Wait for udev To Complete Device Initialization.
[ OK ] Reached target Preparation for Local File Systems.
[ OK ] Mounting /boot.
[ OK ] Starting File System Check on /dev/disk/by-uuid/8C37-687B...
[ OK ] Mounted /boot.
[ OK ] Finished File System Check on /dev/disk/by-uuid/8C37-687B.
[ OK ] Mounting /boot/efi.
[ OK ] Mounted /boot/efi.
[ OK ] Reached target Local File System.
[ OK ] Starting Tell Plymouth To Write Out Runtime Data...
[ OK ] Starting Create Volatile Files and Directories...
[ OK ] Finished Tell Plymouth To Write Out Runtime Data.
[ OK ] Finished Create Volatile Files and Directories.
[ OK ] Starting Security Auditing Service...

```

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="... rhgb quiet"
```

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

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

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

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

3.11 Summary

The first step in working with Rocky Linux 9 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 as 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.

5. Allocating Windows Disk Partitions to Rocky Linux 9

In the previous chapter, we looked at installing Rocky Linux 9 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 Rocky Linux 9 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 Rocky Linux 9 with Windows*” chapter were followed to mount the Windows partition from within Rocky Linux 9, 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 Rocky 9 is to delete that partition. Before doing so, however, any data you need to keep must be backed up from both the Windows and Rocky 9 partitions. Having done that, it is safe to proceed with this chapter.

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

```
# fdisk -l
Disk /dev/nvme0n1: 64 GiB, 68719476736 bytes, 134217728 sectors
Disk model: VMware Virtual NVMe Disk
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: gpt
Disk identifier: 7A38CD86-091E-4781-BFB0-928FD383C935
```

Device	Start	End	Sectors	Size	Type
/dev/nvme0n1p1	2048	206847	204800	100M	EFI System
/dev/nvme0n1p2	206848	239615	32768	16M	Microsoft reserved

Allocating Windows Disk Partitions to Rocky Linux 9

```
/dev/nvme0n1p3      239616   49362943  49123328  23.4G Microsoft basic data
/dev/nvme0n1p4 132933632 134213631  1280000   625M Windows recovery environment
/dev/nvme0n1p5   49362944   51460095  2097152    1G Linux filesystem
/dev/nvme0n1p6   51460096 132933631  81473536  38.8G Linux LVM
```

In the above example output, the system contains one physical disk drive referenced by device name `/dev/nvme0n1`. On that disk drive are six partitions accessed via the device names `/dev/nvme0n1p1` through `/dev/nvme0n1p6`, respectively. Based on the values in the Types column, three Windows-related partitions exist. The first is the Windows system partition, while the second, much larger, partition is the Windows boot partition containing the Windows operating system and user data, followed by the Windows recovery partition.

To remove the partitions, start the *fdisk* tool using the device name of the disk containing the partition (`/dev/nvme0n1` in this instance) and follow the instructions to display the partition and sector information once again:

```
# fdisk /dev/nvme0n1
```

```
Welcome to fdisk (util-linux 2.37.4).
```

```
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/nvme0n1: 64 GiB, 68719476736 bytes, 134217728 sectors
```

```
Disk model: VMware Virtual NVMe Disk
```

```
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: gpt
```

```
Disk identifier: 7A38CD86-091E-4781-BFB0-928FD383C935
```

Device	Start	End	Sectors	Size	Type
<code>/dev/nvme0n1p1</code>	2048	206847	204800	100M	EFI System
<code>/dev/nvme0n1p2</code>	206848	239615	32768	16M	Microsoft reserved
<code>/dev/nvme0n1p3</code>	239616	49362943	49123328	23.4G	Microsoft basic data
<code>/dev/nvme0n1p4</code>	132933632	134213631	1280000	625M	Windows recovery environment
<code>/dev/nvme0n1p5</code>	49362944	51460095	2097152	1G	Linux filesystem
<code>/dev/nvme0n1p6</code>	51460096	132933631	81473536	38.8G	Linux LVM

```
Partition table entries are not in disk order.
```

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

Before proceeding, note the start and end addresses of the partitions we will be deleting (in other words, the start of `/dev/nvme0n1p2` and the sector before the start of `/dev/nvme0n1p5`).

At the command prompt, delete the Windows partitions (these being partitions 2, 3, and 4 on our example system):

```
Command (m for help): d
Partition number (1-6, default 6): 2
```

Partition 2 has been deleted.

```
Command (m for help): d
Partition number (1,3-6, default 6): 3
```

Partition 3 has been deleted.

```
Command (m for help): d
Partition number (1,4-6, default 6): 4
```

Partition 4 has been deleted.

Now that we have deleted the Windows partitions, we need to create the new Rocky 9 partition in the vacated disk space. The partition number must match the number of the first partition removed (in this case, 2). It will also be necessary to enter the Start and End sectors of the partition precisely 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 number (2-4,7-128, default 2): 2
First sector (206848-134217694, default 206848):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (206848-49362943, default
49362943):
```

Created a new partition 2 of type 'Linux filesystem' and of size 23.4 GiB.

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

Having made these changes, the next step is to check that the settings are correct:

```
Command (m for help): p
Disk /dev/nvme0n1: 64 GiB, 68719476736 bytes, 134217728 sectors
Disk model: VMware Virtual NVMe Disk
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: gpt
Disk identifier: 7A38CD86-091E-4781-BFB0-928FD383C935
```

Device	Start	End	Sectors	Size	Type
/dev/nvme0n1p1	2048	206847	204800	100M	EFI System
/dev/nvme0n1p2	206848	49362943	49156096	23.4G	Linux filesystem
/dev/nvme0n1p5	49362944	51460095	2097152	1G	Linux filesystem

Allocating Windows Disk Partitions to Rocky Linux 9

```
/dev/nvme0n1p6 51460096 132933631 81473536 38.8G Linux LVM
```

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

To make the new partition suitable for use by Rocky Linux 9, it needs to have a file system created on it. The recommended file system type for the current release of Rocky Linux is XFS which will be covered in greater detail in the chapter entitled “*Adding a New Disk Drive to a Rocky Linux 9 System*”. Creation of the file system is performed using the *mkfs.xfs* command as follows:

```
# mkfs.xfs -f /dev/nvme0n1p2
meta-data=/dev/nvme0n1p2          isize=512    agcount=4, agsize=1536128 blks
=                               sectsz=512    attr=2, projid32bit=1
=                               crc=1        finobt=1, sparse=1, rmapbt=0
=                               reflink=1    bigtime=1 inobtcount=1
data      =                       bsize=4096    blocks=6144512, imaxpct=25
=                               sunit=0       swidth=0 blks
naming    =version 2             bsize=4096    ascii-ci=0, ftype=1
log       =internal log         bsize=4096    blocks=3000, 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 Rocky Linux 9 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/nvme0n1p2* device to match your environment):

```
/dev/nvme0n1p2    /data    xfs defaults 0 0
```

Finally, we can manually mount the new partition (note that this will not be necessary on subsequent reboots as the partition will automount due to 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
```


Allocating Windows Disk Partitions to Rocky Linux 9

Filesystem	Size	Used	Avail	Use%	Mounted on
/dev/nvme0n1p2	24G	200M	24G	1%	/data

5.5 Summary

The Windows partitions in a dual boot configuration can be removed anytime to free up space for a Rocky Linux 9 system by identifying which partitions belong to Windows and then deleting them. Once deleted, the unallocated space can be used to create a new filesystem and mounted to make it available to the Rocky 9 system.

6. A Guided Tour of the GNOME 40 Desktop

Rocky Linux 9 includes the GNOME 40 desktop environment. Although lacking the complexity of Windows and macOS desktops, GNOME 40 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 the Workstation or Server with GUI software configuration was selected during the Rocky 9 installation process, the GNOME desktop will be installed and automatically launched each time the system starts.

If any other software configuration had been selected during the Rocky 9 installation process, the GNOME desktop would not have been included in the packages installed on the system. Installing a graphical desktop environment may seem redundant on server-based systems without a display attached. It is worth noting, however, that remote access to the GNOME desktop is also possible; even on 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 this book's *“Rocky Linux 9 Remote Desktop Access with VNC”* chapter.

If the installation configuration did not include the GNOME desktop, it can 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 40 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 40 Desktop

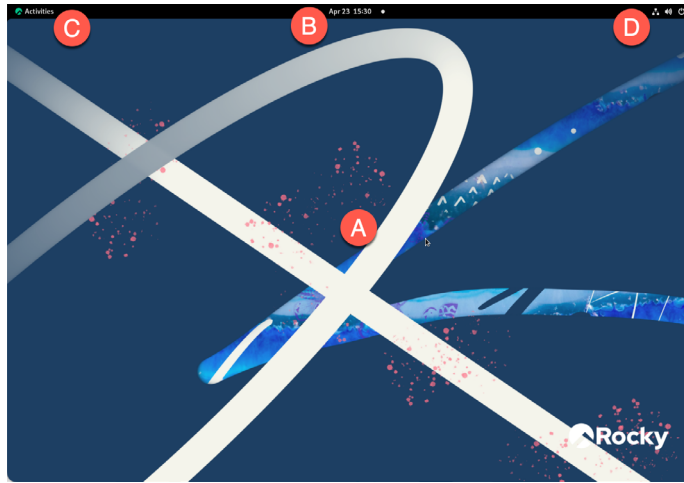


Figure 6-1

The main desktop area (marked A) is where windows will appear when applications and utilities are launched. Unlike other desktop environments, dragging and dropping files or applications onto the desktop is impossible, providing a clean and uncluttered workspace.

The bar at the top of the screen (B) is called the top bar. It includes the Activities menu (C), the day and time (B), and a collection of status icons (D) which, when clicked, displays a menu containing additional options.

In addition, 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:

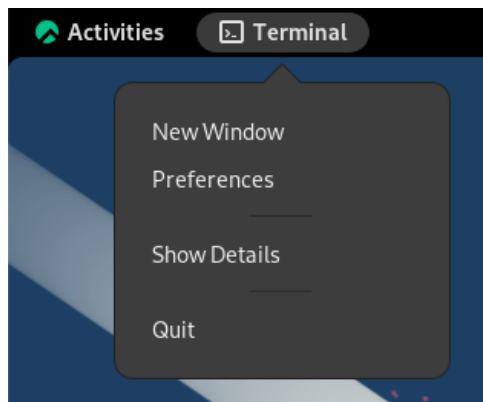


Figure 6-2

6.3 Activity Overview

Applications and utilities are launched using the Activities Overview, which may be displayed by clicking the Activities button in the top bar or pressing the special key on the keyboard. On

Windows keyboards, this is the Windows key; on macOS, the Command key and Chromebooks, the key displaying a magnifying glass.

When displayed, the Activities Overview will resemble Figure 6-3 below:

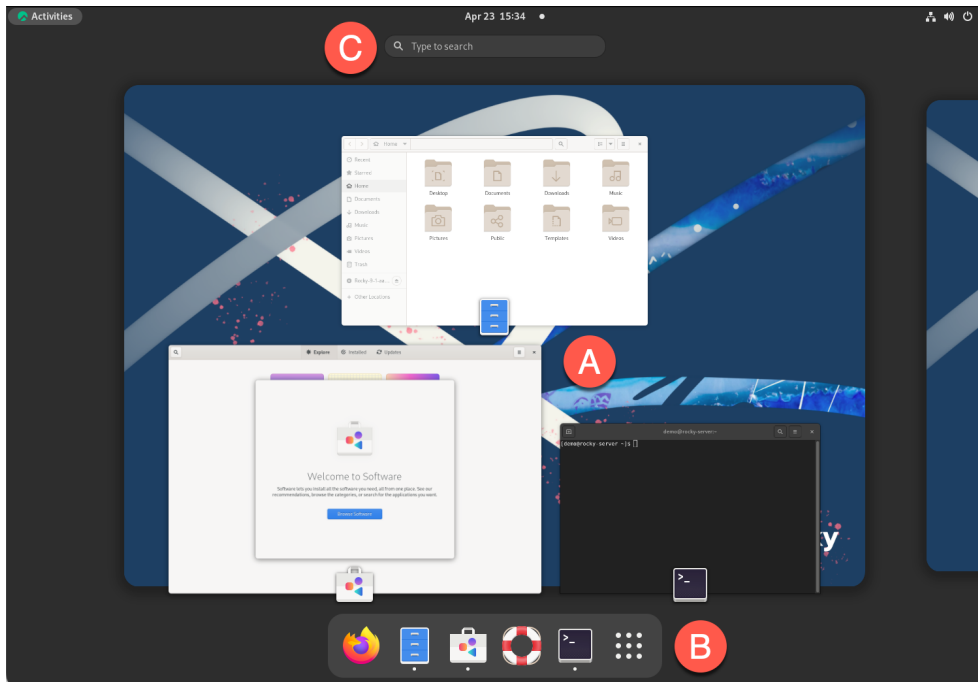


Figure 6-3

The central area of the overview (marked A in Figure 6-3 above) displays all of the currently open app windows. At the bottom of the screen (B) is the dashboard (also referred to as the *dash*) from which apps are launched. By default, the dash will display an icon for a predefined set of commonly used applications and an icon for any currently running applications. If the application is running, it will appear with a dot 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 right-most grid icon (the square comprising nine dots) to display a browsable list of applications as shown in Figure 6-4:

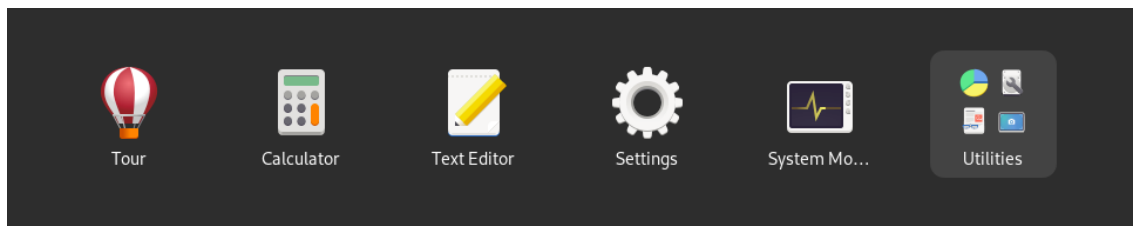


Figure 6-4

A Guided Tour of the GNOME 40 Desktop

It is also important to be aware that some entries in the list are folders holding additional applications. Items in the grid may be repositioned by dragging and dropping the icons. To add an app to a folder, locate it in the grid and drag it into the folder. Moreover, dragging one icon and dropping it onto another will create a new folder containing both apps and to which other apps may be added.

An alternative to browsing the applications is to perform a search using the search bar (marked C in Figure 6-3 above) and shown in action in the figure below:

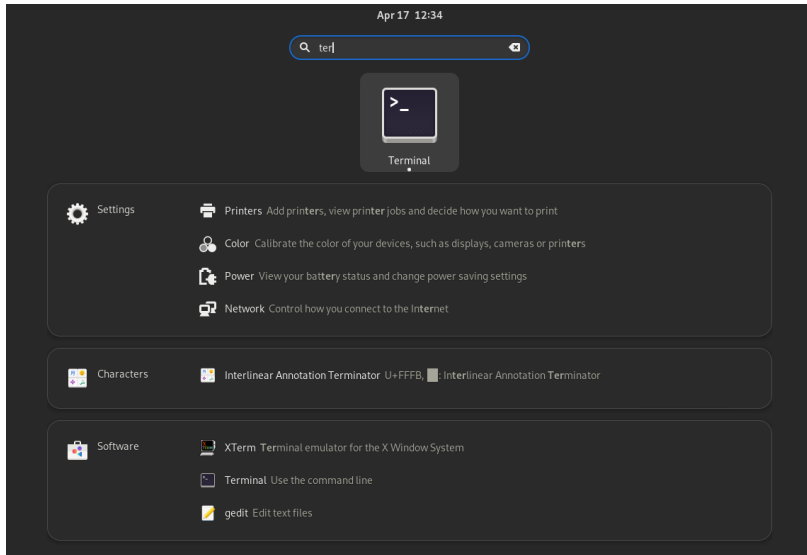


Figure 6-5

The list of possible matches will be refined as the text is typed into the search box.

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:

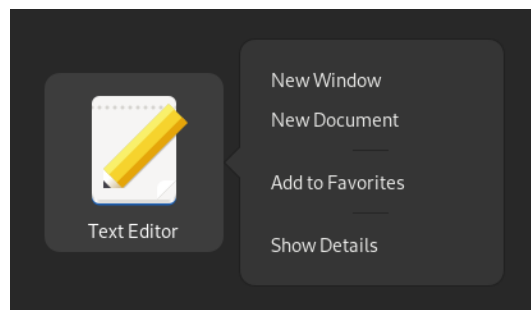


Figure 6-6

To remove an app from the dash, right-click on the icon in the dash and select Remove from Favorites. Alternatively, drag and drop icons to and from the dash to add and remove them.

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):

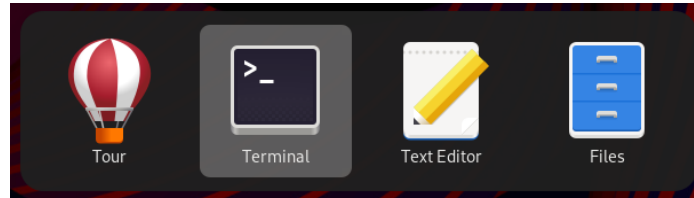


Figure 6-7

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:

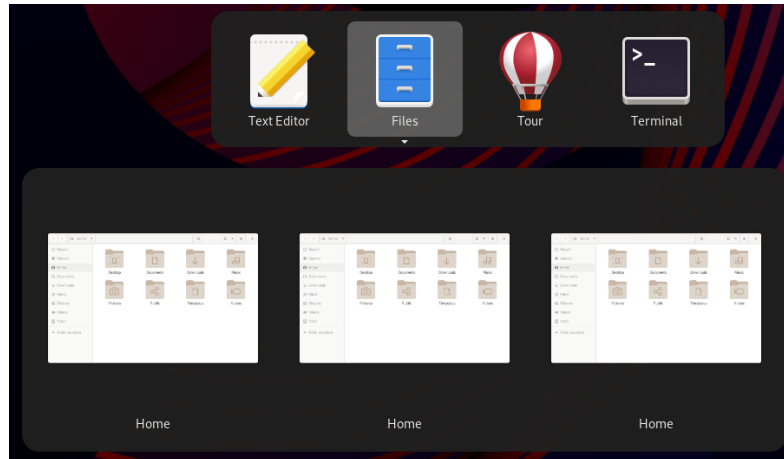


Figure 6-8

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

To maximize a window so 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 it downwards. Alternatively, 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 40 allows multiple workspaces to be configured. The GNOME desktop will launch with two workspaces you can switch between using the Super+Alt+Left and Super+Alt+Right key combinations. When a window is added to a blank workspace, another blank workspace is added to the workspace panel, allowing additional workspaces to be created.

A Guided Tour of the GNOME 40 Desktop

Thumbnails of the current workspaces appear beneath the search field in the Activities Overview as shown in Figure 6-9:

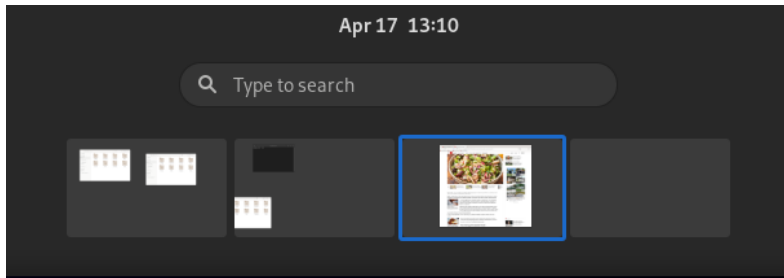


Figure 6-9

To switch to a different workspace, select the corresponding thumbnail. 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. To remove a workspace, close all its windows 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. In addition, 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:

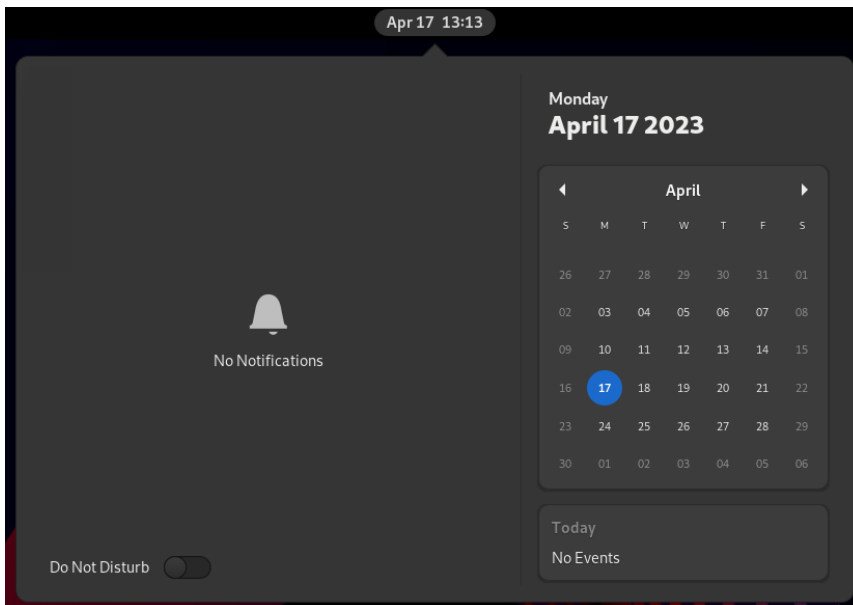


Figure 6-10

6.7 GNOME Desktop Settings

To access the Settings application, click on the row of icons on the right-hand side of the top bar and select Settings as shown in Figure 6-11:

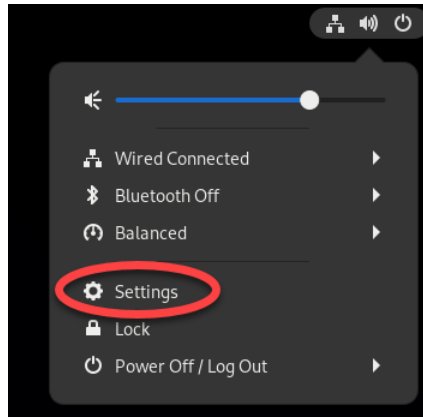


Figure 6-11

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

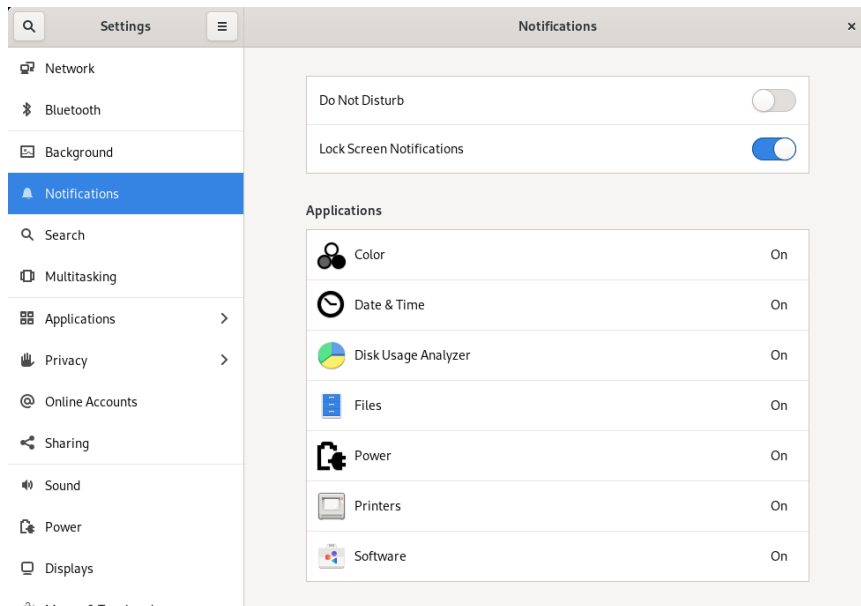


Figure 6-12

A Guided Tour of the GNOME 40 Desktop

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

6.8 Beyond Basic Customization

The GNOME 40 desktop is, by design, a clean and uncluttered environment with minimal customization options. However, it is possible to make additional changes to the desktop. The GNOME Project has developed a tool called GNOME Tweaks for this purpose. Use the following commands to install and run this tool:

```
# dnf install gnome-tweaks
$ gnome-tweaks
```

Once GNOME Tweaks has loaded, the interface shown in Figure 6-13 will appear:

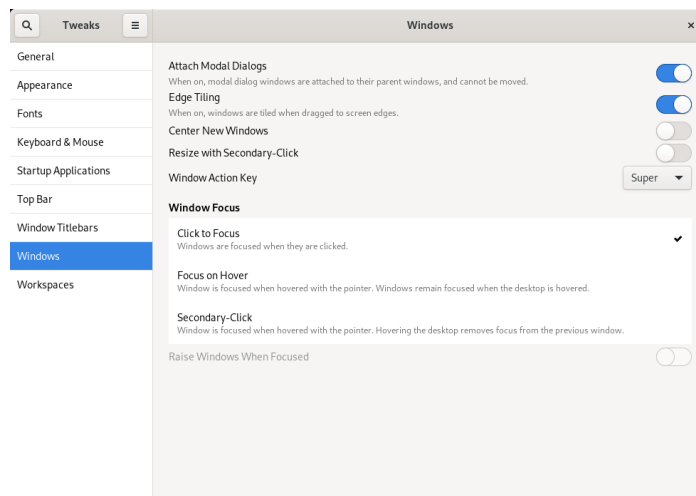


Figure 6-13

A wide range of options for customizing the desktop is now available. Too many to cover in this chapter, so experiment with these settings before proceeding to the next chapter.

6.9 Installing GNOME Desktop Apps

No desktop environment would be complete without an app store, and the GNOME desktop is no exception. The app store for the GNOME desktop is called Software and can be launched from the dash, as highlighted in Figure 6-14:

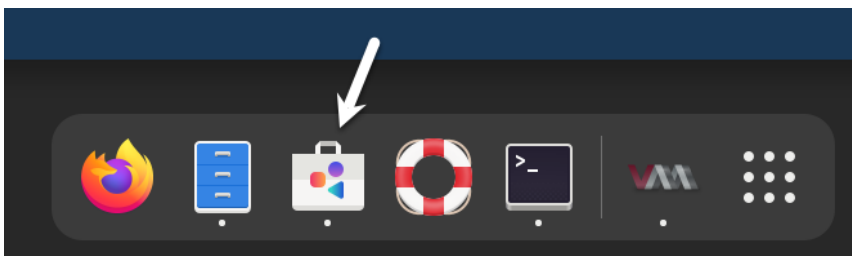


Figure 6-14

Once loaded, the Software app will appear as shown in Figure 6-15, allowing you to browse and install apps:

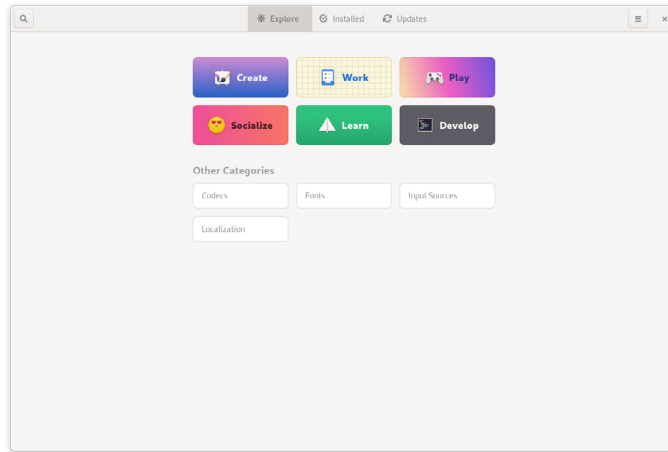


Figure 6-15

6.10 Summary

Rocky 9 includes the GNOME 40 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 40 is intended to provide a clean and easy-to-use windowing user interface. Key areas of the GNOME 40 desktop include the top bar, Activities overview, and dash. In addition, GNOME 40 supports multiple workspaces keeping running applications organized and the screen uncluttered. Various configuration options are available within the Settings app, including desktop background settings, audio, network configuration, and WiFi network selection.

15. Configuring SSH Key-based Authentication on Rocky Linux 9

When a Rocky Linux 9 system is first installed, it is configured by default to allow remote command-line access via Secure Shell (SSH) connections. SSH provides password-protected and encrypted access to the system for the root account and any other users added during the installation phase. However, this level of security is inadequate and should be upgraded to SSH key-based authentication as soon as possible.

This chapter will outline the steps to increase the security of a Rocky 9 system by implementing key-based SSH authentication.

15.1 An Overview of Secure Shell (SSH)

SSH allows secure remote access to systems to gain shell access and transfer files and data. As will be covered in “*Rocky Linux 9 Remote Desktop Access with VNC*”, SSH can also provide a secure tunnel through which remote access to the GNOME desktop can be achieved over a network connection.

A basic SSH configuration consists of a client (used on the computer establishing the connection) and a server (running on the system to which the connection is to be established). A user might, for example, use an SSH client running on a Linux, Windows, or macOS system to connect to the SSH server running on a Rocky 9 system to gain access to a shell command-line prompt or to perform file transfers. All communications between the client and server, including the password entered to gain access, are encrypted to prevent outside parties from intercepting the data.

The inherent weakness in a basic SSH implementation is that it depends entirely on the strength of the passwords assigned to the accounts on the system. If a malicious party is able to identify the password for an account (either through guesswork, deception, or a brute force attack), the system becomes vulnerable. This weakness can be addressed by implementing SSH key-based authentication.

15.2 SSH Key-based Authentication

SSH key-based authentication uses asymmetric public key encryption to add an extra layer of security to remote system access. The concept of public key encryption was devised in 1975 by Whitfield Diffie and Martin Hellman and is based on using a pair of private and public keys.

In a public key encryption system, the public key is used to encrypt data that can only be decrypted by the owner of the private key.

In the case of SSH key-based authentication, the host holds the private key on which the SSH

Configuring SSH Key-based Authentication on Rocky Linux 9

client is located, while the corresponding public key resides on the system on which the SSH server is running. Therefore, it is vital to protect the private key since ownership of the key will allow anyone to log into the remote system. As an added layer of protection, the private key may also be encrypted and protected by a password which must be entered each time a connection is established to the server.

15.3 Setting Up Key-based Authentication

There are four steps to setting up key-based SSH authentication, which can be summarized as follows:

1. Generate the public and private keys.
2. Install the public key on the server.
3. Test authentication.
4. Disable password-based authentication on the server.

The remainder of this chapter will outline these steps in greater detail for Linux, macOS, and Windows-based client operating systems.

15.4 Installing and Starting the SSH Service

If the SSH server is not already installed and running on the system, it can be added using the following commands:

```
# dnf install openssh-server
# systemctl start sshd.service
# systemctl enable sshd.service
```

15.5 SSH Key-based Authentication from Linux and macOS Clients

The first step in setting up SSH key-based authentication is to generate the key pairs on the client system. If the client system is running Linux or macOS, this is achieved using the *ssh-keygen* utility:

```
# ssh-keygen
```

This command will result in output similar to the following:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
```

Press the Enter key to accept the default location for the key files. This will place two files in the *.ssh* sub-directory of the current user's home directory. The private key will be stored in a file named *id_rsa* while the public key will reside in the file named *id_rsa.pub*.

Next, *ssh-keygen* will prompt for a passphrase with which to protect the private key. If a passphrase is provided, the private key will be encrypted on the local disk, and the passphrase will be required to access the remote system. Therefore, for better security, the use of a passphrase is recommended.

```
Enter passphrase (empty for no passphrase):
```

Finally, the *ssh-keygen* tool will generate the following output indicating that the keys have been generated:

```
Your identification has been saved in /home/neil/.ssh/id_rsa.
Your public key has been saved in /home/neil/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:FOLGWEEGFijWnCT5wtTOv5VK4hdimzWghZizUEMYbfo <username>@<hostname>
The key's randomart image is:
+---[RSA 2048]-----+
|.BB+=+*..          |
|o+B= * . .         |
|===.. + .          |
|*+ * . .           |
|.++ o   S          |
|..E+ * o           |
| o B *             |
|  + +              |
|   .                |
+-----[SHA256]-----+
```

The next step is to install the public key onto the remote server system. This can be achieved using the *ssh-copy-id* utility as follows:

```
$ ssh-copy-id username@remote_hostname
```

For example:

```
$ ssh-copy-id neil@192.168.1.100
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/neil/.ssh/
id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are
prompted now it is to install the new keys
neil@192.168.1.100's password:
```

```
Number of key(s) added: 1
```

Now try logging into the machine, with: `"ssh 'neil@192.168.1.100'"`
and check to make sure that only the key(s) you wanted were added.

Once the key is installed, test that the authentication works by attempting a remote login using the *ssh* client:

```
$ ssh -l <username> <hostname>
```

If the private key is encrypted and protected with a passphrase, enter the phrase when prompted to complete the authentication and establish remote access to the Rocky 9 system:

```
Enter passphrase for key '/home/neil/.ssh/id_rsa':
Last login: Fri Mar 31 14:29:28 2023 from 192.168.86.21
```

Configuring SSH Key-based Authentication on Rocky Linux 9

```
[neil@demosystem02 ~]$
```

Repeat these steps for any other accounts on the server for which remote access is required. If access is also required from other client systems, copy the *id_rsa* private key file to the *.ssh* sub-directory of your home folder on the other systems.

As currently configured, access to the remote system can still be achieved using less secure password authentication. Once you have verified that key-based authentication works, password authentication will need to be disabled on the system. To understand how to change this setting, begin by opening the */etc/ssh/sshd_config* file and locating the following line:

```
Include /etc/ssh/sshd_config.d/*.conf
```

This tells us that *sshd* configuration settings are controlled by files in the */etc/ssh/sshd_config.d* directory. These filenames must be prefixed with a number and have a *.conf* filename extension, for example:

```
01-permitrootlogin.conf
50-redhat.conf
```

The number prefix designates the priority assigned to the file relative to the other files in the folder, with 01 being the highest priority. This ensures that if a configuration file contains a setting conflicting with another file, the one with the highest priority will always take precedence.

Within the */etc/ssh/sshd_config.d* folder, create a new file named *02-nopasswordlogin.conf* with content that reads as follows:

```
PasswordAuthentication no
```

Save the file and restart the *sshd* service to implement the change:

```
# systemctl restart sshd.service
```

From this point on, it will only be possible to remotely access the system using SSH key-based authentication, and when doing so, you won't be required to enter a password.

15.6 Managing Multiple Keys

It is common for multiple private keys to reside on a client system, each providing access to a different server. As a result, several options exist for selecting a specific key when establishing a connection. It is possible, for example, to specify the private key file to be used when launching the *ssh* client as follows:

```
$ ssh -l neilsmlyth -i ~/.ssh/id_work 35.194.18.119
```

Alternatively, the SSH client user configuration file may associate key files with servers. The configuration file is named *config*, must reside in the *.ssh* directory of the user's home directory, and can be used to configure a wide range of options, including the private key file, the default port to use when connecting, the default user name, and an abbreviated nickname via which to reference the server. The following example *config* file defines different key files for two servers and allows them to be referenced by the nicknames *home* and *work*. In the case of the *work* system, the file also specifies the user name to be used when authenticating:


```
Host work
  HostName 35.194.18.119
  IdentityFile ~/.ssh/id_work
  User neilsmyth
```

```
Host home
  HostName 192.168.0.21
  IdentityFile ~/.ssh/id_home
```

Before setting up the configuration file, the user would have used the following command to connect to the work system:

```
$ ssh -l neilsmyth -i ~/.ssh/id_work 35.194.18.119
```

Now, however, the command can be shortened as follows:

```
$ ssh work
```

A full listing of configuration file options can be found by running the following command:

```
$ man ssh_config
```

15.7 SSH Key-based Authentication from Windows Clients

Recent releases of Windows include a subset of the OpenSSH implementation used by most Linux and macOS systems as part of Windows PowerShell. This allows SSH key-based authentication to be set up from a Windows client using similar steps to those outlined above for Linux and macOS.

On Windows, search for Windows PowerShell and select it from the results. Once running, the PowerShell window will appear as shown in Figure 15-1:

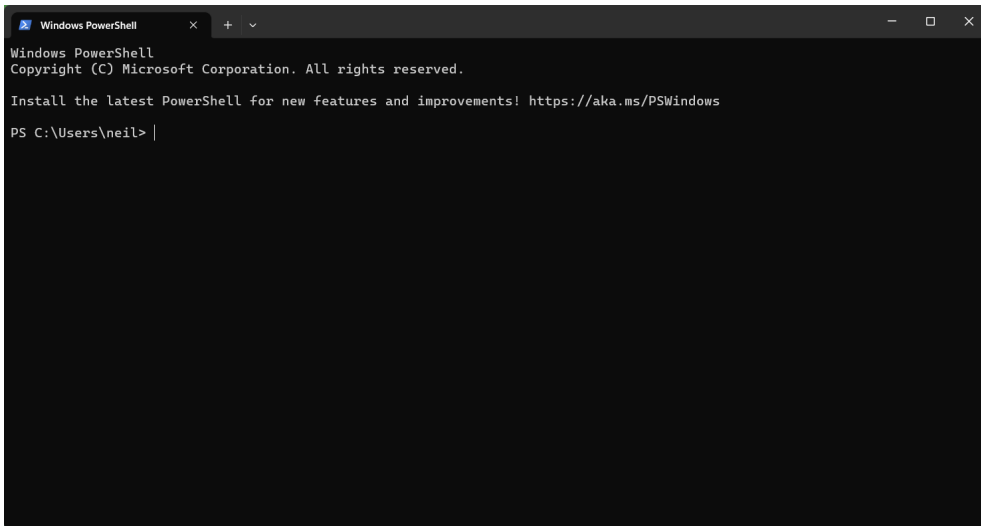


Figure 15-1

If you already have a private key from another client system, copy the *id_rsa* file to a folder named *.ssh* on the Windows system. Once the file is in place, test the authentication within the PowerShell

Configuring SSH Key-based Authentication on Rocky Linux 9

window as follows:

```
$ ssh -l <username>@<hostname>
```

For example:

```
PS C:\Users\neil> ssh -l neil 192.168.1.101
Enter passphrase for key 'C:\Users\neil\.ssh\id_rsa':
```

Enter the passphrase when prompted and complete the authentication process.

If the private key does not yet exist, generate a new private and public key pair within the PowerShell window using the *ssh-keygen* utility using the same steps outlined for Linux and macOS. Once the keys have been generated, they will again be located in the *.ssh* directory of the current user's home folder, and the public key file *id_rsa.pub* will need to be installed on the remote Rocky 9 system. Unfortunately, Windows PowerShell does not include the *ssh-copy-id* utility, so this task must be performed manually.

Within the PowerShell window, change directory into the *.ssh* sub-directory and display the content of the public key *id_rsa.pub* file:

```
PS C:\Users\neil> cd .ssh
PS C:\Users\neil\.ssh> type id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDFgx1vzu591116/uQw7FbmKVsq3fzLz9MW1fgo4sdsx
Xp8lwCHNAlqcjx1Pgr9BJPXWUMInQOi7BQ5I+vc2xQ2AS0kmq3ZH9ybWuQe/U2GjueXZd0FKrEXrT55wM
36Rm6Ii3roUCoGCzGR8mn95JvRB3VtCyDdzTWSi8JBpK5gV5oOxNTNPsew1LzouBlCT1qW3CKwEiIwu8S
9MTL7m3nrcaNeLewTTHevvHw4QDwzFQ+B0PDg96fzsYoTXVhzyHSWyo6H0gqrft7aK+gILBtEihWtkSVE
MAZylpiKtCr1IYTmVK6engv0aoGtMUq6FnOeGp5FjvKkF4aQkh1QR28r neil@DESKTOP-S8P8D3N
```

Highlight the file's content and copy it using the Ctrl-C keyboard shortcut.

Remaining within the PowerShell window, log into the remote system using password authentication:

```
PS C:\Users\neil\.ssh> ssh -l <username> <hostname>
```

Once signed in, check if the *.ssh* sub-directory exists. If it does not, create it as follows:

```
$ mkdir .ssh
```

Change directory into *.ssh* and check whether a file named *authorized_keys* already exists. If it does not, create it and paste the content of the public key file from the Windows system into it.

If the *authorized_keys* file already exists, it likely contains other keys. If this is the case, edit the file and paste the new public key at the end of the file. The following file, for example, contains two keys:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDCzRWH27Xs8ZA5rIbZXKgxFY5XXauMv+6F5PljBLJ6j
+9nkmykVe3GjZTp3oD+KMRbT2kTEPbDpFD67DNL0eiX2ZuEEiYsxZfGCRCrPBGYmQtFRHEAFnlS1Jx/
G4W5UNKvhAXWymwDEKiWvqTVy6syB2Ritoak+D/Sc8nJfIQ6dtw0jBs+S7Aim8TPfppi4p5XJGruXNRS
camk68NgnPfTL3vT726EuABck6C934KARD+/AXa8/5rNOh4ETPstjBRfFJ0tpmsWWhhNENwJRqS2LD0
ug7E3yFI2qsNKGEzvAYUC8Up45MRP7liR3aM1CBil1tsy9R+IB7oMEycZAe/qj neil@localhost.
localdomain
```

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDFgx1vzu591116/uQw7FbmKVsq3fzLz9MW1fgo4sdsx
Xp8lwCHNAlqcjx1Pgr9BJPXWUMInQOi7BQ5I+vc2xQ2AS0kmq3ZH9ybWuQe/U2GjueXZd0FKrEXrT55wM
```

Configuring SSH Key-based Authentication on Rocky Linux 9

```
36Rm6Ii3roUCoGCzGR8mn95JvRB3VtCyDdzTWSi8JBpK5gV5oOxNTNPsewLzouBlCT1qW3CKWeIiWu8S
9MTL7m3nrcaNeLewTTHevvHw4QDwzFQ+B0PDg96fzsYoTXVhzyHSWyo6H0gqrft7aK+gILBtEihWTkSVE
MAzylpiKtCr1IYTmVK6engv0aoGtMUq6FnOeGp5FjvKkF4aQkh1QR28r neil@DESKTOP-S8P8D3N
```

Once the public key is installed on the server, test the authentication by logging in to the server from within the PowerShell window, for example:

```
PS C:\Users\neil\.ssh> ssh -l neil 192.168.1.100
Enter passphrase for key 'C:\Users\neil\.ssh\id_rsa':
```

When key-based authentication has been set up for all the accounts and verified, disable password authentication on the Rocky 9 system as outlined at the end of the previous section.

15.8 SSH Key-based Authentication using PuTTY

For Windows systems that do not have OpenSSH available or as a more flexible alternative to using PowerShell, the PuTTY tool is a widely used alternative. The first step in using PuTTY is downloading and installing it on any Windows system that needs an SSH client. PuTTY is a free utility and can be downloaded using the following link:

<https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>

Download the Windows installer executable that matches your Windows system (32-bit and 64-bit versions are available), then execute the installer to complete the installation.

If a private key already exists on another system, create the `.ssh` folder in the current user's home folder and copy the private `id_rsa` key into it.

Next, the private key file needs to be converted to a PuTTY private key format file using the PuTTYgen tool. Locate this utility by typing “PuTTY Key Generator” into the search bar of the Windows Start menu and launch it:

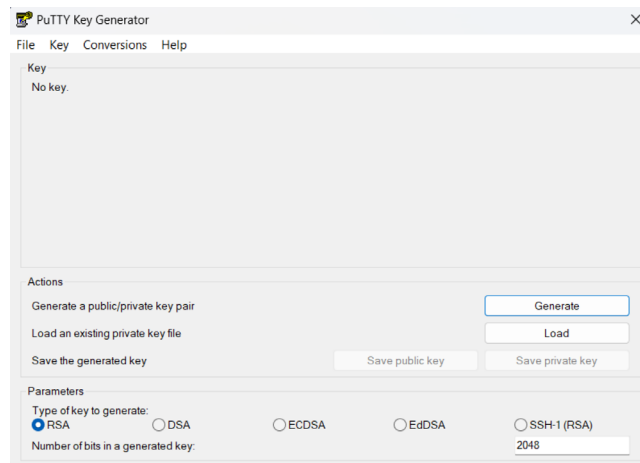


Figure 15-2

Once launched, click on the Load button located in the Actions section and navigate to the private key file previously copied to the `.ssh` folder (note that it may be necessary to change the file type

Configuring SSH Key-based Authentication on Rocky Linux 9

filter to All Files (*.*) for the key file to be visible). Once located, select the file and load it into PuTTYGen. When prompted, enter the passphrase used initially to encrypt the file. Once the private key has been imported, save it as a PuTTY key file by clicking the Save Private Key button. For consistency, save the key file to the `.ssh` folder but give it a different name to differentiate it from the original key file.

Launch PuTTY from the Start menu and enter the IP address or hostname of the remote server into the main screen before selecting the *Connection -> SSH -> Auth -> Credentials* category in the left-hand panel, as highlighted in Figure 15-3:

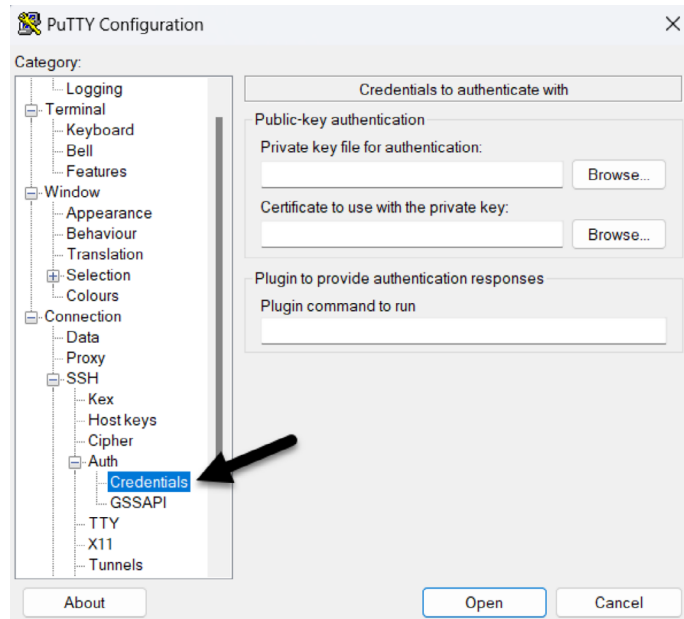


Figure 15-3

Click the Browse button next to the *Private key for authentication* field and navigate to and select the previously saved PuTTY private key file. Then, optionally, scroll to the top of the left-hand panel, select the Session entry, and enter a name for the session in the Saved Sessions field before clicking on the Save button. This will save the session configuration for future use without re-entering the settings each time.

Finally, click on the Open button to establish the connection to the remote server, entering the user name and passphrase when prompted to do so to complete the authentication.

15.9 Generating a Private Key with PuTTYgen

The previous section explored using existing private and public keys when working with PuTTY. If keys do not exist, they can be created using the PuTTYgen tool, which is included in the main PuTTY installation.

To create new keys, launch PuttyGen and click on the Generate button highlighted in Figure 15-4:

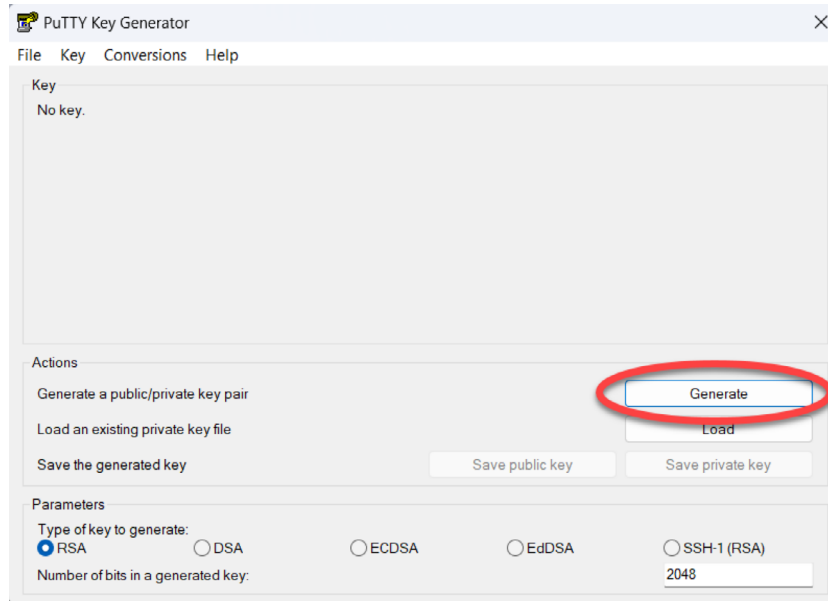


Figure 15-4

Move the mouse pointer to generate random data as instructed, then enter an optional passphrase to encrypt the private key. Once the keys have been generated, save the files to suitable locations using the Save public key and Save private key buttons. As outlined in the previous section, the private key can be used with PuTTY. To install the public key on the remote server, use the steps covered in the earlier section on SSH within PowerShell on Windows.

15.10 Summary

Any remote access to a Rocky 9 system must be implemented in a way that provides a high level of security. By default, SSH allows remote system access using password-based authentication. However, this leaves the system vulnerable to anyone who can guess a password or find out the password through other means. For this reason, key-based authentication is recommended to protect system access. Key-based authentication uses public key encryption involving public and private keys. When implemented, users can only connect to a server if they are using a client with a private key that matches a public key on the server. As an added layer of security, the private key may also be encrypted and password protected. Once key-based encryption has been implemented, the server system is configured to disable support for the less secure password-based authentication.

This chapter has provided an overview of SSH key-based authentication and outlined the steps involved in generating keys and configuring clients on macOS, Linux, and Windows, as well as installing and managing public keys on a Rocky 9 server.

18. Using NFS on Rocky Linux 9 to Share Files with Remote Systems

Rocky Linux 9 provides two mechanisms for sharing files and folders with other systems on a network. One approach is to use a technology called Samba. Samba is based on Microsoft Windows Folder Sharing and allows Linux systems to make folders accessible to Windows systems and access Windows-based folder shares from Linux. This approach can also be used to share folders between other Linux and UNIX-based systems as long as they have Samba support installed and configured. This is the most popular approach to sharing folders in heterogeneous network environments. Folder sharing using Samba is covered in *“Sharing Files between Rocky Linux 9 and Windows Systems with Samba”*.

Another option, explicitly targeted at sharing folders between Linux and UNIX-based systems, uses Network File System (NFS). NFS allows the file system on one Linux computer to be accessed over a network connection by another Linux or UNIX system. NFS was originally developed by Sun Microsystems (now part of Oracle Corporation) in the 1980s and remains the standard mechanism for sharing remote Linux/UNIX file systems.

NFS is very different from the Windows SMB resource-sharing technology used by Samba. This chapter will look at the network-based sharing of folders between Rocky 9 and other UNIX/Linux-based systems using NFS.

18.1 Ensuring NFS Services are running on Rocky Linux 9

The first task is to verify that the NFS services are installed and running on your Rocky 9 system. This can be achieved from the command line or the Cockpit interface.

Behind the scenes, NFS uses Remote Procedure Calls (RPC) to share filesystems over a network between different computers in the form of the rpcbind service. Begin by installing both rpcbind and the NFS service by running the following command from a terminal window:

```
# dnf install rpcbind nfs-utils
```

Next, configure these services so that they automatically start at boot time:

```
# systemctl enable rpcbind
# systemctl enable nfs-server
```

Once the services have been enabled, start them as follows:

```
# systemctl start rpcbind
# systemctl start nfs-server
```

Using NFS on Rocky Linux 9 to Share Files with Remote Systems

18.2 Configuring the Rocky Linux 9 Firewall to Allow NFS Traffic

Next, the firewall needs to be configured to allow NFS traffic. To achieve this, run the following *firewall-cmd* commands where *<zone>* is replaced by the appropriate zone for your firewall and system configuration:

```
# firewall-cmd --zone=<zone> --permanent --add-service=mountd
# firewall-cmd --zone=<zone> --permanent --add-service=nfs
# firewall-cmd --zone=<zone> --permanent --add-service=rpc-bind
# firewall-cmd --reload
```

18.3 Specifying the Folders to be Shared

Now that NFS is running and the firewall has been configured, we need to specify which parts of the Rocky 9 file system may be accessed by remote Linux or UNIX systems. These settings can be declared in the */etc/exports* file, which must be modified to export the directories for remote access via NFS. The syntax for an export line in this file is as follows:

```
<export> <host1>(<options>) <host2>(<options>)...
```

In the above line, *<export>* is replaced by the directory to be exported, *<host1>* is the name or IP address of the system to which access is being granted, and *<options>* represents the restrictions that are to be imposed on that access (read-only, read-write, etc.). Multiple host and options entries may be placed on the same line if required. For example, the following line grants read-only permission to the */datafiles* directory to a host with the IP address 192.168.2.38:

```
/datafiles 192.168.2.38(ro)
```

The use of wildcards is permitted to apply an export to multiple hosts. For example, the following line permits read-write access to */home/demo* to all external hosts:

```
/home/demo * (rw)
```

A complete list of options supported by the exports file may be found by reading the exports man page:

```
# man exports
```

For this chapter, we will configure the */etc/exports* file as follows:

```
/tmp          * (rw, sync)
/voll         192.168.86.42 (ro, sync)
```

Once configured, the table of exported file systems maintained by the NFS server needs to be updated with the latest */etc/exports* settings using the *exportfs* command as follows:

```
# exportfs -a
```

It is also possible to view the current share settings from the command line using the *exportfs* tool:

```
# exportfs
```

The above command will generate the following output:

```
/voll          192.168.86.42
/tmp           <world>
```


18.4 Accessing Shared Folders

The shared folders may be accessed from a client system by mounting them manually from the command line. However, before attempting to mount a remote NFS folder, the *nfs-utils* package must first be installed on the client system:

```
# dnf install nfs-utils
```

To mount a remote folder from the command line, open a terminal window and create a directory where you would like the remote shared folder to be mounted:

```
$ mkdir /home/demo/tmp
```

Next, enter the command to mount the remote folder using either the IP address or hostname of the remote NFS server, for example:

```
$ sudo mount -t nfs 192.168.86.24:/tmp /home/demo/tmp
```

The remote */tmp* folder will then be mounted on the local system. Once mounted, the */home/demo/tmp* folder will contain the remote folder and all its contents.

Options may also be specified when mounting a remote NFS filesystem. The following command, for example, mounts the same folder but configures it to be read-only:

```
$ sudo mount -t nfs -o ro 192.168.86.24:/tmp /home/demo/tmp
```

18.5 Mounting an NFS Filesystem on System Startup

It is also possible to configure a Rocky 9 system to automatically mount a remote file system each time it starts up by editing the */etc/fstab* file. When loaded into an editor, it will likely resemble the following:

/dev/mapper/rl-root	/	xfs	defaults	0 0
UUID=c4ba0b0f-777a-42a8	/boot	xfs	defaults	0 0
UUID=052D-19D8	/boot/efi	vfat	umask=0077,shortname=winnt	0 2
/dev/mapper/rl-swap	none	swap	defaults	0 0

To mount, for example, a folder with the path */tmp*, which resides on a system with the IP address 192.168.86.24 in the local folder with the path */home/demo/tmp* (note that this folder must already exist), add the following line to the */etc/fstab* file:

192.168.86.24:/tmp	/home/demo/tmp	nfs	rw	0 0
--------------------	----------------	-----	----	-----

Next time the system reboots, the */tmp* folder on the remote system will be mounted on the local */home/demo/tmp* mount point. All the files in the remote folder can then be accessed as if they reside on the local hard disk drive.

18.6 Unmounting an NFS Mount Point

Once a remote file system is mounted using NFS, it can be unmounted using the *umount* command with the local mount point as the command-line argument. The following command, for example, will unmount our example filesystem mount point:

```
$ sudo umount /home/demo/tmp
```

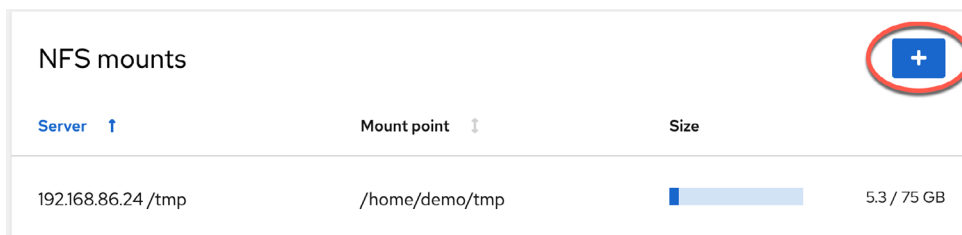
18.7 Accessing NFS Filesystems in Cockpit

In addition to mounting a remote NFS file system on a client using the command line, it is also possible to perform mount operations from within the Cockpit web interface. Assuming that Cockpit has been installed and configured on the client system, log into the Cockpit interface from within a web browser and select the Storage option from the left-hand navigation panel. If the Storage option is not listed, the *cockpit-storaged* package will need to be installed:

```
# dnf install cockpit-storaged
# systemctl restart cockpit.socket
```

Once the Cockpit service has restarted, log back into the Cockpit interface, at which point the Storage option should now be visible.

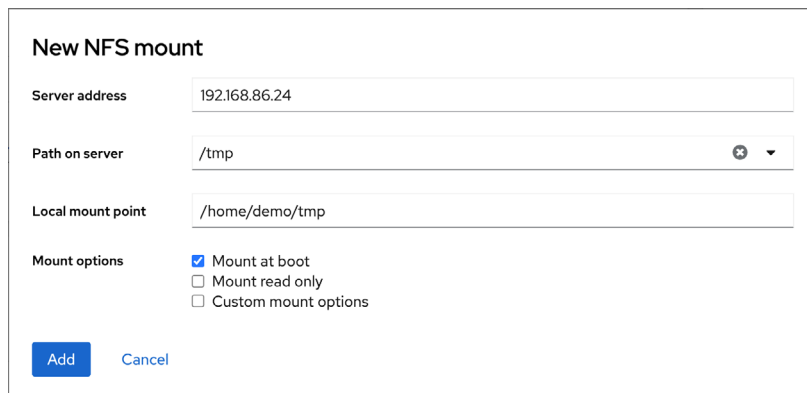
Once selected, the main storage page will include a section listing any currently mounted NFS file systems, as illustrated in Figure 18-1:



NFS mounts			
Server	Mount point		Size
192.168.86.24	/tmp	/home/demo/tmp	5.3 / 75 GB

Figure 18-1

To mount a remote filesystem, click on the ‘+’ button highlighted above and enter information about the remote NFS server and file system share together with the local mount point and any necessary options into the resulting dialog before clicking on the Add button:



New NFS mount

Server address: 192.168.86.24

Path on server: /tmp

Local mount point: /home/demo/tmp

Mount options:

- ☒ Mount at boot
- ☐ Mount read only
- ☐ Custom mount options

Add Cancel

Figure 18-2

To modify, unmount or remove an NFS filesystem share, select the corresponding mount in the NFS Mounts list (Figure 18-1 above) to display the page shown in Figure 18-3 below:

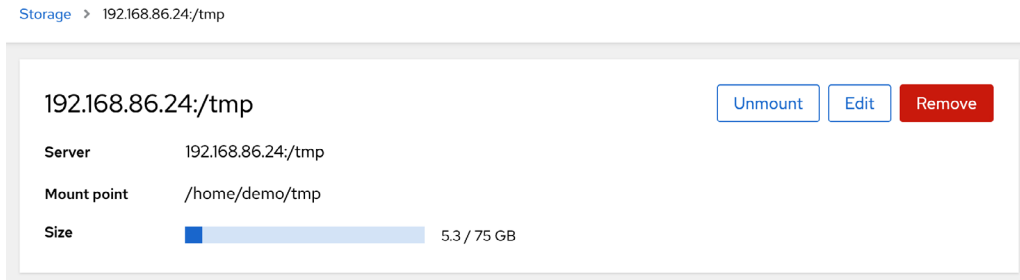


Figure 18-3

Within this screen, perform tasks such as changing the server or mount points or unmounting the file system. For example, the Remove option unmounts the file system and deletes the entry from the */etc/fstab* file so that it does not re-mount the next time the system reboots.

18.8 Summary

The Network File System (NFS) is a client/server-based system, originally developed by Sun Microsystems, which provides a way for Linux and Unix systems to share filesystems over a network. NFS allows a client system to access and (subject to permissions) modify files located on a remote server as though those files are stored on a local filesystem. This chapter has provided an overview of NFS and outlined the options for configuring client and server systems using the command line or the Cockpit web interface.

27. An Introduction to Linux Containers

The preceding chapters covered the concept of virtualization, emphasizing creating and managing virtual machines using KVM. This chapter will introduce a related technology in the form of Linux Containers. While there are some similarities between virtual machines and containers, key differences will be outlined in this chapter, along with an introduction to the concepts and advantages of Linux Containers. The chapter will also overview some Rocky Linux 9 container management tools. Once the basics of containers have been covered in this chapter, the next chapter will work through some practical examples of creating and running containers on Rocky Linux 9.

27.1 Linux Containers and Kernel Sharing

In simple terms, Linux containers are a lightweight alternative to virtualization. A virtual machine contains and runs the entire guest operating system in a virtualized environment. The virtual machine, in turn, runs on top of an environment such as a hypervisor that manages access to the physical resources of the host system.

Containers work by using a concept referred to as kernel sharing, which takes advantage of the architectural design of Linux and UNIX-based operating systems.

To understand how kernel sharing and containers work, it helps first to understand the two main components of Linux or UNIX operating systems. At the core of the operating system is the kernel. The kernel, in simple terms, handles all the interactions between the operating system and the physical hardware. The second key component is the root file system which contains all the libraries, files, and utilities necessary for the operating system to function. Taking advantage of this structure, containers each have their own root file system but share the host operating system's kernel. This structure is illustrated in the architectural diagram in Figure 27-1 below.

This type of resource sharing is made possible by the ability of the kernel to dynamically change the current root file system (a concept known as change root or chroot) to a different root file system without having to reboot the entire system. Linux containers are essentially an extension of this capability combined with a container runtime, the responsibility of which is to provide an interface for executing and managing the containers on the host system. Several container runtimes are available, including Docker, lxd, containerd, and CRI-O. Earlier versions of Rocky Linux used Docker by default, but Podman has supplanted this as the default in Rocky Linux 9

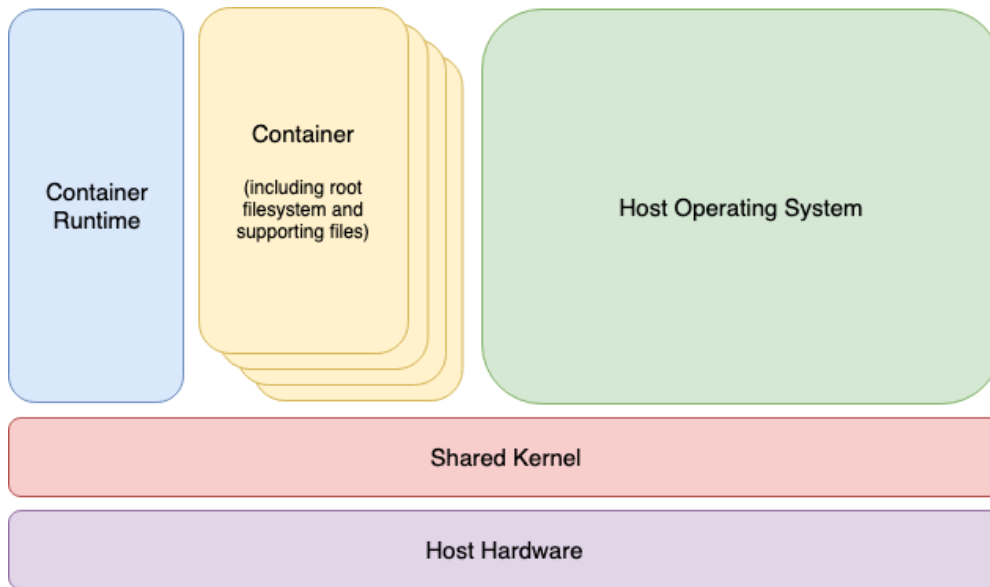


Figure 27-1

27.2 Container Uses and Advantages

The main advantage of containers is that they require considerably less resource overhead than virtualization allowing many container instances to be run simultaneously on a single server. They can be started and stopped rapidly and efficiently in response to demand levels. In addition, containers run natively on the host system providing a level of performance that a virtual machine cannot match.

Containers are also highly portable and can be easily migrated between systems. Combined with a container management system such as Docker, OpenShift, and Kubernetes, it is possible to deploy and manage containers on a vast scale spanning multiple servers and cloud platforms, potentially running thousands of containers.

Containers are frequently used to create lightweight execution environments for applications. In this scenario, each container provides an isolated environment containing the application together with all of the runtime and supporting files required by that application to run. The container can then be deployed to any other compatible host system that supports container execution and runs without any concerns that the target system may not have the necessary runtime configuration for the application - all of the application's dependencies are already in the container.

Containers are also helpful when bridging the gap between development and production environments. By performing development and QA work in containers, they can be passed to production and launched safely because the applications run in the same container environments in which they were developed and tested.

Containers also promote a modular approach to deploying large and complex solutions. Instead of developing applications as single monolithic entities, containers can be used to design applications

as groups of interacting modules, each running in a separate container.

One possible drawback of containers is that the guest operating systems must be compatible with the version of the kernel being shared. It is not, for example, possible to run Microsoft Windows in a container on a Linux system. Nor is it possible for a Linux guest system designed for the 2.6 version of the kernel to share a 2.4 version kernel. These requirements are not, however, what containers were designed for. Rather than being seen as limitations, these restrictions should be considered some of the key advantages of containers in providing a simple, scalable, and reliable deployment platform.

27.3 Rocky Linux 9 Container Tools

Rocky Linux 9 provides several tools for creating, inspecting, and managing containers. The main tools are as follows:

- **buildah** – A command-line tool for building container images.
- **podman** – A command-line based container runtime and management tool. Performs tasks such as downloading container images from remote registries and inspecting, starting, and stopping images.
- **skopeo** – A command-line utility used to convert container images, copy images between registries and inspect images stored in registries without downloading them.
- **runc** – A lightweight container runtime for launching and running containers from the command line.
- **OpenShift** – An enterprise-level container application management platform consisting of command-line and web-based tools.

All of the above tools comply with the Open Container Initiative (OCI), a set of specifications designed to ensure that containers conform to the same standards between competing tools and platforms.

27.4 The Docker Registry

Although Rocky 9 is provided with a set of tools designed to be used in place of those provided by Docker, those tools still need access to Rocky Linux images for use when building containers. For this purpose, the Rocky Enterprise Linux Foundation maintains a set of container images within the Docker Hub. The Docker Hub is an online container registry made of multiple repositories, each containing a wide range of container images available for download when building containers. The images within a repository are each assigned a repository tag (for example, 9.1, latest, etc.) which can be referenced when performing an image download. The following, for example, is the URL of the Rocky Linux 9.1 image contained within the Docker Hub:

```
docker://docker.io/library/rockylinux
```

In addition to downloading (referred to as “pulling” in container terminology) container images from Docker and other third-party hosts registries, you can also use registries to store your own

images. This can be achieved either by hosting your own registry, or by making use of existing services such as those provided by Docker, Amazon AWS, Google Cloud, Microsoft Azure, and IBM Cloud, to name a few of the many options.

27.5 Container Networking

By default, containers are connected to a network using a Container Networking Interface (CNI) bridged network stack. In the bridged configuration, all the containers running on a server belong to the same subnet and, as such, can communicate with each other. The containers are also connected to the external network by bridging the host system's network connection. Similarly, the host can access the containers via a virtual network interface (usually named `podman0`) which will have been created as part of the container tool installation.

27.6 Summary

Linux Containers offer a lightweight alternative to virtualization and take advantage of the structure of the Linux and Unix operating systems. Linux Containers share the host operating system's kernel, with each container having its own root file system containing the files, libraries, and applications. As a result, containers are highly efficient and scalable and provide an ideal platform for building and deploying modular enterprise-level solutions. In addition, several tools and platforms are available for building, deploying, and managing containers, including third-party solutions and those provided by the Rocky Linux project.

Index

Symbols

! 61
 #! 65
 >> 63
 | 63
 \$DISPLAY variable 136
 50-redhat.conf 135
 .bashrc 65
 /etc/containers/networks 216
 /etc/default/grub 27
 /etc/exports 142
 /etc/fstab 29, 31, 143, 145, 241
 /etc/gdm/custom.conf 135
 /etc/group 69
 /etc/httpd 223
 /etc/passwd 69
 /etc/samba/smb.conf 148
 /etc/shadow 69
 /etc/sshd_config 135
 /etc/ssh/sshd_config.d 135
 /etc/sudoers 69
 /etc/systemd/system 86, 128
 /etc/yum.repos.d/ directory 76
 /home 67
 /proc/swaps 255
 .requires 87
 .ssh 116, 118
 /usr/lib/systemd/system 86
 /var/log/maillog 234
 .wants 86

A

aarch64 10
 ACC Corporation 6
 Activity Overview 38

adduser 67
 AIX 5
 alias 63
 Aliases 63
 AMD-V 162
 Andrew S. Tanenbaum 5
 Apache
 mod_ssl 225
 Apache web server 221
 Application Stream 75
 AppStream 75, 77
 modules 77
 packages 77
 profiles 77
 AppStream repository 75
 ARM64 10
 authorized_keys file 120

B

BaseOS repository 75, 76
 Bash
 scripts 65
 Bash shell 59
 aliases 63
 .bashrc 65
 chmod 66
 command-line editing 60
 do loops 65
 echo 64
 environment variables 64
 filename completion 62
 Filename shorthand 62
 for loop 65
 history 61
 HOME 64
 input and output redirection 62
 PATH 64
 path completion 62
 pipes 63

Index

- sh 66
- stderr 63
- stdin 62
- stdout 62
- basic.target 82
- Bell Labs 59
- Boot ISO 9
- Bourne Again SHell 59
- Bourne shell 59
- Brian Fox 59
- buildah 205
 - containers 212
 - from registry 212
 - from scratch 212
 - installroot 213
 - run 213
 - umount 213
- Buildah 207

C

- CA 224
- cat 62
- certbot 225
- Certificate Authority 224
- change root 203
- chmod 66
- chroot 203
- cifs filesystem 157
- CNI 206, 213
- Cockpit 87
 - accessing 48
 - account management 52
 - applications 53
 - cockpit-machines 169
 - cockpit-storaged 144
 - create VM 169
 - Drives 242
 - enabling 48
 - extensions 47, 53
 - firewall management 112
 - installing 48

- logs 50
- Multiple Servers 55
- networking 51
- NFS 144
- overview 47
- persistent metrics 56
- Podman Containers 216
- port 48
- select bridge 193
- services 52, 87
- storage 51, 241
- system 49
- systemd 87
- terminal access 54
- user management 69
- virtual machines 53
- volume groups 251
- cockpit-machines 169
- cockpit.socket 48
- cockpit-storaged 144, 251
- Compressed X11 Forwarding 136
- Connection Profiles 94
- containerd 203
- Container Networking Interface 206, 213
- Containers
 - attaching to an image 210
 - bridge networking 213
 - buildah 212
 - networking 213
 - Networking Interface 213
 - overview 203
 - pulling an image 207
 - removing an image 211
 - running an image 209
 - saving to an image 211
 - stopping 210
- context labels 151
- CRI-O 203
- C shell 59

D

- daemon 81
- dash 39
- David Korn 59
- dd 11
- DDNS 221
- Default Boot Option 27
- default.target 84
- df 250, 266
- discretionary access control 150
- disk drive
 - detecting 237
- disk I/O 267
- Disk partition
 - formatting 34
- disk usage 266
- diskutil 11
- DISPLAY variable 136
- dmesg 10
- dnf 22
 - groupinfo 126
 - groupinstall 126
 - grouplist 126
- dnf.conf file 76
- DNS 102
- DNS MX Records 233
- Docker 203, 204
- do loops 65
- Domain Name Server 102
- dual boot 31
- Dual Booting 25
- DVD ISO 9
- Dynamic DNS 221
- DynDNS 221

E

- echo 64
- Email Server 229
- encryption
 - disk 20
- Environment Variables 64
- EPEL 28

- Errata 4
- export 64
- exportfs 142
- ext2 251
- ext3 251
- ext4 251
- Extra Packages for Enterprise Linux 28

F

- FAT16 25
- FAT32 25
- fdisk 29, 31, 238, 249
 - create partition 238
 - list partitions 238
- Fedora Linux 7
- Fedora Media Writer 12
- Fedora Project 7
- Filename Shorthand 62
- File System
 - creating 239
 - mounting 240
- File Transfer (Control) 101
- File Transfer Protocol (Data) 101
- findmnt 10
- Firewall
 - Interfaces 107
 - overview 99, 105
 - Port Forwarding 111
 - Ports 107
 - Services 107
 - web server settings 222
 - Zones 105
- firewall-cmd 101, 108, 128
 - enable NFS 142
 - mail settings 231
 - web server settings 222
- firewall-config 113
- firewalld
 - default zone 108
 - display zone information 108
 - firewall-cmd 108

Index

- firewall-config 113
- ICMP rules 111
- interfaces 105, 107
- list services 109
- overview 105
- permanent settings 108
- port forwarding 111
- port rules 109, 110
- ports 105, 107
- reload 108
- runtime settings 108
- services 105, 107
- status 107
- zone creation 110
- zone/interface assignments 110
- zones 105
- zone services 109

for 65

ForwardX11Trusted 136

free 256

- s flag 266

Free Software Foundation 6

fsck 240

fstab 143, 241

FTP 99, 101

Full Virtualization 162

Fuse NTFS driver 28

G

GDM 22

gedit 136

getfacl 172

GNOME

- apps 44

- Software app 44

GNOME Desktop 37

- Activity Overview 38

- dash 39

- gnome-tweaks 44

- settings 43

- starting 37

- switcher 41

- tweaks 44

- windows 41

- Workspaces 41

GNOME Desktop Environment 125

GNOME Display Manager 22

gnome-system-monitor 263

gnome-tweaks 44

GNU/Linux 6

GNU project 7

graphical.target 82

groupadd 68

groupdel 68

groups 68

grub2-set-default 28

GRUB_SAVEDFAULT 27

Guest OS virtualization 159

H

Hardware Virtualization 162

Hewlett-Packard 5

history 60, 61

HOME 64

HP-UX 5

HTTP 103, 224

httpd 221, 225

httpd.conf 223, 225

httpd-le-ssl.conf 226

HTTPS 99, 104, 107, 224

hypercalls 161

Hypertext Text Transfer Protocol 103

Hypertext Transfer Protocol Secure 104

Hypervisor 160

- hypercalls 161

- type-1 160

- type-2 160

Hypervisor Virtualization 160

I

IBM 5

ICMP 111

id_rsa file 116, 118, 119

id_rsa.pub file 116, 120

if statements 65

IMAP4 103

Input and Output Redirection 62

installation

disk partitioning 18

Installation

clean disk 9

install packages

listing 76

Intel VT 162

Intel x86

Rings 161

Internet Control Message Protocol 111

Internet Message Access Protocol, Version 4 103

internet service provider 221

I/O redirection 63

iotop 267

installing 267

IPSets 114

iptables 99, 101, 105

rules 100

tool 100

ip tool 92

ISO image

Boot 9

DVD 9

Minimal 9

write to USB drive 10

ISP 221

J

journalctl 128, 133

Journalized File Systems 240

K

Kdump 17

kernel 5

kill 262

-9 flag 262

Korn shell 59

Kubernetes 204

KVM

hardware requirements 165

installation 166

overview 165

virt-manager 166

kvm_amd 167

kvm_intel 167

KVM virtualization 163

L

LE 246

Let's Encrypt 225

libvirt 177

libvirt daemon 167

Linus Torvalds 6

Linux Containers. *See* Containers

Logical Extent 246

Logical Volume 246

Logical Volume Management 245

loopback interface 91

lost+found 240

ls 60, 63

lscpu 165

lsmod 166

LV 246

lvdisplay 247, 250, 257

LVM 245

lxd 203

M

macOS

writing ISO to USB drive 11

MacVTap 163

Mail Delivery Agent 229

Mail Exchanger 233

Mail Transfer Agent 229

Mail User Agent 229

main.cf 231

man 60

Index

mandatory access control 150
Marc Ewing 6
Martin Hellman 115
MDA 229
Minimal ISO 9
MINIX 5
mkfs.xfs 34, 239
mkswap 256, 257
mod_ssl 225
mount 29, 143, 156, 240, 246
MTA 229
MUA 229
multi-user.target 82
MX 233

N

NAT 163, 171
NetBIOS 153
NetBIOS nameservice 153
Network Address Translation 163
Networked Bridge Interface 189
Network File System 104
NetworkManager
 Connection Profiles 94
 enabling 90
 installing 90
 permissions 98
Network News Transfer Protocol 103
Network Time Protocol 103
NFS 104
 Cockpit 144
 firewall settings 142
nfs-client.target 82
NFS service 141
nfs-utils 143
NMB 153
nmcli 89, 191
 activate connection 93
 add bridge 191
 add connections 95
 command line options 90

 deactivate connection 93
 delete connection 95
 device status 91
 general status 90
 interactive 96
 modify connection 94
 permissions 98
 reload 94
 show connections 92
 switch connection 92
 wifi scan 93

nm-connection-editor 89

 create bridge 195

nmtui 89

NNTP 103

NTFS 25

NTP 103

O

OCI 205

Open Container Initiative 205

OpenShift 204, 205

OSI stack 103

P

Paravirtualization 161, 162

Partition

 mounting 34

passwd 67

PATH 64

Path Completion 62

PE 246

Physical Extent 246

Physical Volume 246

Pipes 63

podman 205

 attach 210

 commit 211

 exec 210, 211

 images 209, 211

 inspect 209

- network commands 215
- network connect 215
- network create 215
- network disconnect 215
- network inspect 215
- network ls 214
- network rm 216
- pause 211
- ps -a 210
- pull 208
- rm 212
- run 209
- search 208
- stop 210
- unpause 211
- Podman 203, 207
- POP3 103
- Port Forwarding 111, 222
- Ports
 - securing 99
- Postfix 229, 230
 - configuring 231
 - installing 231
 - main.cf 231
 - postmap 234
 - sasl_passwd 234
 - starting 233
 - testing 233
- postmap 234
- Post Office Protocol 103
- poweroff.target 81
- PowerShell 119
- ppc64le 10
- private key 115
- ps 63, 151, 261
 - a flag 261
 - aux flags 262
 - H flag 263
 - TERM signal 262
 - u flag 261
- public key 115

- public key encryption 115
- PuTTY 121
 - secure tunnel 131
 - X11 Forwarding 138
- PuTTYgen 121
- PuTTY Key Generator 121
- PV 246
- pvccreate 250, 259
- pvdisplay 248
- pwd 60

Q

- QEMU 178
- QEMU/KVM Hypervisor 167
- Qmail 230

R

- RealVNC 127
- reboot.target 82
- Red Hat, Inc. 5
- Red Hat Package Manager 76
- Red Hat Support 6
- Remote Desktop Access
 - insecure 125
 - secure 125
- remote-fs.target 82
- remote installation 16
- Remote Procedure Calls 141
- Repositories 75
- rescue.target 81
- resize2fs 251
- restorecon 152
- RHGB 27
- Richard Stallman 6
- Rocky Linux
 - history of 5
- root password
 - specifying during installation 21
- root user 1
- RPC 141
- rpcbind 141

Index

rpm 76
RPM 76
runc 205

S

s390x 10
Samba 141, 148
 add user 152
 firewall settings 148
 installing 148
 NetBIOS 153
 samba_share_t 151
 SELinux 150
 smbclient 153, 156
 smb.conf 148
 smbpasswd 152
 smb_t 151
 testparm 153
Samba Client 148
samba_share_t 151
sasl_passwd 234
SATA 237
Secure File Transfer Protocol 101
Secure Shell 102, 115
Secure Socket Layer 224
Secure Sockets Layer 104
Secure Tunnel 131
SELinux
 context labels 151
 restorecon 152
 Samba 150
 sestatus 150
 type enforcement 151
SELinux
 enforcing mode 151
 permissive mode 151
Sendmail 229, 230
Server Message Block 147
Server with GUI 125
Services
 securing 99

sestatus 150
setfacl 172
Settings App 71
 users 71
SFTP 101
sh 66
Shell Scripts 65
Simple Mail Transfer Protocol 102
Simple Mail Transport Protocol 230
Simple Network Management Protocol 104
skopeo 205, 207
Skopeo 207
SMB 147
smbclient 153, 156
smb.conf 148
 testing 152
 testparm 152
smbpasswd 152
smb_t 151
SMTP 99, 102, 107, 230
SMTP Relay 230, 234
SNMP 104
sockets.target 82
Solaris 5
spawning 263
ssh
 -C flag 136
 X11 Forwarding 136
 -X flag 136
SSH 99, 102, 129, 115
 Microsoft Windows 119
 Multiple Keys 118
 VNC 129
ssh client 117
ssh-copy-id 117, 120
sshd_config.d 135
sshd_config.d directory 118
sshd_config file 118
sshd service 118
ssh-keygen 116
SSH Service

- installing 116
- starting 116
- SSH tunnel 130
- SSL 104, 224
- SSL certificate 224, 225
- SSL Labs 227
- startx 37, 127
- stderr 63
- stdin 62
- stdout 62
- storage devices
 - identify 10
- Storage Pools 173
- Storage Volumes 173
- su - command 1
- sudo 1
 - wheel group 68
- SunOS 5
- Superuser 1
- Swap 255
 - current size 255
 - free 256
 - logical volume 257
 - lvdisplay 257
 - mkswap 256, 257
 - /proc/swaps 255
 - pvccreate 259
 - recommendations 255
 - swapoff 256, 259
 - swapon 256
- swapoff 256, 259
- swapon 256
- system
 - units 84
 - unit types 84
- systemctl 83
 - daemon-reload 128
- systemd 81
 - services 81
 - targets 81
- System Monitor 263

- system processes 261

T

- TCP/IP 99, 107
 - Well-Known Ports 101
- Telnet 102
- Terminal window 2
- TERM signal 262
- testparm
 - smb.conf 152
- TFTP 102
- TigerVNC 127, 129
 - installation 127
 - viewer 127
- TightVNC 127
- TLS 224
- top 265
 - u flag 266
- Transport Layer Security 224
- Trivial File Transfer Protocol 102
- Trusted X11 Forwarding 136
- Type-1 hypervisors 161
- Type-2 hypervisors 161
- type enforcement 151

U

- UDP 101
- umount 11, 143
- UNIX 5, 59
 - origins of 5
- update packages 77
- USB drive
 - device name 10
- userdel 67
- usermod 68
- user processes 261
- Users and Groups 67

V

- VcXsrv 137
- VG 245

Index

vgdisplay 246
vgextend 250
vgs 247, 259
virbr0 189, 190
virsh 185, 187, 192, 199
 destroy 187, 201
 dumpxml 187
 edit 194
 help 200
 list 200
 reboot 202
 restore 201
 resume 201
 save 201
 setmem 202
 setmemmax 202
 shell 199
 shutdown 187, 201
 start 187, 201
 suspend 201
virt-install 169, 185, 193
virt-manager 166, 177, 193
 installation 166
 New VM wizard 178
 storage pools 180
VirtualBox 165
Virtualization 159
 AMD-V 162
 full 162
 guest 159
 hardware 162
 hypercalls 161
 hypervisor 160
 Intel VT 162
 KVM virtualization 163
 MacVTap 163
 Type-1 160
 Type-2 160
 virt-manager 166
Virtual Machine Networking 163
Virtual Network Computing 125

virt-viewer 172, 186
VMware 165
VNC 125
 firewall settings 128
 installation 127
 PuTTY 131
 secure 130
 server shutdown 132
 start server 128
 troubleshooting 132
 vncpasswd 127
 vncserver 128
 vncserver@.service 128
vncpasswd 127
vncserver 128
VNC Server
 configuring 127
vncserver@.service 128
vncviewer 130
Volume Group 245

W

Wayland 135
wc 63
Web Server 221
 testing 222
Well-Known Ports 101
wheel group 68
which 60
Whitfield Diffie 115
wildcard character 62
wildcards 62
Windows
 accessing partition from Linux 28
 Disk Management 25
 Dual Booting 25
 shrink volume 26
 writing ISO to USB drive 12
Windows partition
 reclaiming 31
 unmounting 31

Windows PowerShell 119
wipefs 250
Workspaces 41
Workstation 125

X

X11 Forwarding 135
 compressed 136
X11Forwarding 135
x86 family 161
Xen 166
XFS file system 239
XFS filesystem 250
xfs_growfs 250
XLaunch 137
X.org 135
X Window System 135

Y

yum.repos.d directory 76

