Red Hat Enterprise Linux 9 for SysAdmins

A technical guide for building secure production systems using RHEL 9 administration

Jerome Gotangco

Luca Berton

www.bpbonline.com

First Edition 2025

Copyright © BPB Publications, India

ISBN: 978-93-65891-171

Dedicated to

My family and closest friends - they are the foundation of who I am today;

***Jose**, **Nidia**, **Joseph**, **Joshua**, **Gigi**, **Veronica**, **Imman**, **Valentine**, **Pauline**, and **Elnaz**;*

Thank you for the continued support and encouragement.

-Jerome Gotangco

*My son **Filippo** – the joy of my life*

-Luca Berton

About the Authors

- **Jerome Gotangco** is a technology consultant with extensive experience in the IT Industry. Currently an OpenShift Specialist at Red Hat, he champions the Open Hybrid Cloud approach for enterprises using open-source software platforms. Jerome has a strong background in open source, previously involved in OpenStack and Ubuntu development. Prior to Red Hat, he served as an Azure consultant at Microsoft and held technology consultant roles at Accenture and Hewlett Packard Enterprise. Beyond his technical expertise, Jerome enjoys reading, writing essays and poems, painting miniature figurines, and maintaining an active lifestyle with fitness and golf.

- **Luca Berton** is a seasoned IT professional with over 18 years of experience in Linux system administration, cloud engineering, and automation. Renowned as an IT Automation expert, Luca has authored several best-selling books, including *Ansible for VMware by Examples, Ansible for Kubernetes by Examples, Red Hat Ansible Automation Platform, and Hands-on Ansible Automation.*

  Luca's dedication to open-source projects and his commitment to sharing knowledge are reflected in his creation of the Ansible Pilot project, which serves as a resource for automation enthusiasts worldwide. His expertise extends to presenting at international conferences, contributing to the growth of the Ansible community, and mentoring aspiring IT professionals.

  As a thought leader and practitioner, Luca combines a deep technical understanding with practical insights, making his work a trusted resource for IT professionals globally. Through this book, he aims to equip readers with the skills and confidence to excel in their careers and beyond.

Acknowledgements

- As a first-time author, this book took some time to complete, even though I had numerous tips and advice received to write as fast as I can and refine later, I made sure every chapter written didn't just get the approval of the editing team involved – I had to convince myself that each command worked to my personal standard.

  I thank my family for their continued support of my career and my aspirations in life. Even though I embarked on a fulfilling career in IT, writing has been close to my heart, be it creative or technical.

  I also extend my thanks to my fellow associates at Red Hat for their support on this book project, most especially my former managers, Anirban Mukherjee and Lennie Tan, for their approval and support to me when I took on this book writing project. Without them, this book wouldn't even be possible.

  Finally, to my co-author, Luca Berton, for his guidance and most important, his patience on agreeing to see this book from the beginning to what you have in your hands today.

  - Jerome Gotangco

- Creating this book has been a journey of collaboration, effort, and learning. I am profoundly grateful to all who made this possible.

  First, I want to express my heartfelt thanks to my family for their unending support and patience during this journey. Their encouragement has been my foundation.

  I am deeply appreciative of the guidance provided by my mentors and colleagues in the IT and open-source communities. A special acknowledgment goes to the Ansible community for nurturing the growth of automation enthusiasts like me.

  Thank you to my publishers for their trust, vision, and professionalism in transforming this work into reality.

  Lastly, to the readers, thank you for choosing this book. I hope it serves as a valuable guide on your path forward in your professional career and beyond.

  - Luca Berton

Preface

First, we wish to thank you for getting this book, be it for your continued learning using Linux distributions as a hobby, or just simply interested in learning a new operating system. Second, this book has been written for you, dear system administrators of the world, to enable and empower you to be productive with the latest edition of Red Hat's flagship Linux distribution Red Hat Enterprise Linux 9 (RHEL 9). This book is written in such a way that it can be used as a handy reference whenever you need to do something with my RHEL 9 system, like installing a database service, spinning up a single node instance for Kubernetes, and more! Treat it like a cookbook if you will – always ready to give you the quickest and best way to cook up a recipe. In fact, it is the way the book is structured!

Who is this book for and how to use the cookbook?

As the title suggests, **Red Hat Enterprise Linux 9 for SysAdmins** is written for System Administrators or SysAdmins who are responsible for installing, configuring, and managing an RHEL 9 server in their respective IT environments, be it deployed in a data center rack server, on-premises in the company as a stand-alone server, a virtual machine running in a hypervisor, or even in public and private cloud as a virtual machine deployment. From a day-to-day point of view, SysAdmins are responsible for ensuring computer systems, such as servers, are running in peak condition and tuned to handle volumes of transactions. SysAdmins also do proactive planning and monitoring; these systems are accessed securely and don't affect the productivity of users relying on the services running in the systems being managed by SysAdmins. In short, they are the unsung heroes of today's hyper-connected world, making sure the applications that we use every day, be it a social network, a digital wallet, or an online banking application, just among others, are always available and fast for our day-to-day transactions.

This book is written for junior to mid-level SysAdmins who have managed a Linux server before and need to become productive with RHEL 9 in a short amount of time. The book contains over 107 recipes of tasks in building and managing a RHEL 9 system along with applications – from installing RHEL 9, building a database server, designing a cluster for high availability, and creating a platform for artificial intelligence, in the style of a cookbook. Each chapter will focus on a particular topic and will contain at least 5 recipes to get your task done in a shorter amount of time. The table of contents is structured this way along to make it easy to go back to your favorite recipes anytime!

We have written this book in a cookbook style and below are just some of the great things you'll learn in each chapter:

Chapter 1: Introducing Red Hat Enterprise Linux 9 - You will learn what Red Hat Enterprise Linux (or RHEL), its origin, and how critical this operating system is in the enterprise. You will also get additional information on getting support for RHEL, as well as known alternatives to RHEL.

Chapter 2: Setting Up RHEL 9 - You will get information on downloading RHEL 9 and creating bootable media to be used for the installation. When you get to do an installation, you will learn the basics as well as advance lessons on disk partitioning and automating the RHEL 9 installation for mass deployments using Kickstart.

Chapter 3: Establish RHEL 9 on Cloud - Cloud computing is a great way to learn about RHEL and you will learn how to deploy and configure RHEL 9 on the major cloud providers: AWS, Microsoft Azure, and Google Cloud Platform. You will also learn how to manage your RHEL 9 deployments with the Red Hat Hybrid Cloud Console and use existing subscriptions to save money on cloud deployments with Red Hat Cloud Access.

Chapter 4: Miscellaneous Configurations of RHEL 9 - When you get to install and set up a RHEL 9 system, you will need to do some customizations and configurations to make it usable and secure. This chapter will guide you through basic and essential configurations in RHEL 9.

Chapter 5: Managing RHEL 9 Subscriptions - Running RHEL 9 requires a subscription from Red Hat and you will learn in this chapter how to obtain a no-cost developer subscription for your learning journey, as well as registering your RHEL 9 systems to receive software updates and send diagnostic reports to Red Hat to get technical support.

Chapter 6: Configuring Software Repositories and RHEL 9 Updates - You will learn to add third-party software repositories to be able to install new software as well as update already installed software.

Chapter 7: Managing RHEL 9 with GNOME Desktop - RHEL 9 includes the GNOME Desktop and you will learn how to configure and navigate with this simple graphical user interface to use and manage your RHEL 9 system.

Chapter 8: Managing Infrastructure and Databases - RHEL 9 provides a large repository of enterprise-grade open-source infrastructure and database software that you can install and configure in this chapter.

Chapter 9: Administration of Virtualization Workloads - You will learn how to create virtual machines and manage them in RHEL 9.

Chapter 10: Create, Manage, and Monitor Containers - RHEL 9 has robust container support with Podman and Buildah and you will have a good grasp of the tools by creating and managing container workloads in RHEL 9.

Chapter 11: Working Around Networks, Files, and Storage Services - This chapter covers the essentials of networking, file systems, and storage in RHEL 9. You will learn to configure and manage network interfaces, implement firewall rules, and set up DNS and DHCP services. Additionally, you will delve into managing file systems, including creating and mounting file systems, configuring NFS and SMB for file sharing, and working with advanced storage solutions like Logical Volume Manager (LVM) and Stratis.

Chapter 12: Source Codes, DevOps Pipelines, and Application Development - This chapter explores how RHEL seamlessly integrates with modern development practices. It teaches you to set up development environments, manage source code repositories with Git, and build CI/CD pipelines using tools like Jenkins, Ansible, and Podman. You will also explore RHEL debugging utilities and performance profilers to streamline application development and deployment.

Chapter 13: Administration of Clusters and Servers - RHEL is a trusted choice for enterprise-grade clustering and server management. This chapter focuses on creating and managing high-availability clusters with tools like Pacemaker and Corosync. You will learn to configure load balancers, optimize server performance, and maintain uptime for critical services. You will also explore distributed file systems and shared storage for clusters.

Chapter 14: Security Hardening of RHEL - Securing systems is vital in the enterprise landscape. This chapter provides comprehensive guidance on hardening your RHEL installations. Topics include SELinux configuration, managing firewalls with FirewallD, implementing audit rules, and enforcing security policies. Learn about Red Hat Insights, the SCAP Security Guide, and automating security compliance checks to ensure your systems meet enterprise security standards.

Chapter 15: Capacity Planning, Log Analysis, and System Audits - Efficient resource management and monitoring are crucial for RHEL administrators. In this chapter, you will explore tools like top, vmstat, and iotop to monitor system performance and plan for future capacity needs. You'll also learn to manage and analyze logs using Rsyslog and Logrotate and conduct system audits using tools like Auditd to maintain compliance and traceability.

Chapter 16: Artificial Intelligence and Machine Learning - Discover how RHEL supports AI/ML workloads. This chapter introduces you to deploying and managing AI frameworks and libraries like TensorFlow, PyTorch, and Scikit-learn on RHEL. Learn to optimize hardware for AI/ML, and leverage containerized environments for machine learning pipelines to accelerate AI/ML application development.

This is all you need to use this cookbook and go over through the recipes!

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/ba22ae

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Red-Hat-Enterprise-Linux-9-for-SysAdmins**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Table of Contents

Chapter 1
Introducing Red Hat Enterprise Linux 9

Introduction

Welcome to **Red Hat Enterprise Linux** (**RHEL**) 9 for SysAdmins!

This chapter gives you a brief introduction to RHEL 9 — what it is, why it was created, what is its intended purpose, and who is the target audience for this book. You will also learn where to gather additional information on RHEL 9 outside the scope of this book, as well as other Linux distributions that have deep ties with RHEL 9 and known derivatives and alternatives that you might want to consider if you are seeking to experience the cutting edge of open-source Linux distributions and software. So, with that, let us begin.

Structure

In this chapter, we will discuss the following topics:

- Understanding Red Hat Enterprise Linux
- Getting additional information on RHEL
- Alternatives to RHEL

Objectives

By the end of this chapter, you will understand what RHEL is and its history, along with learning how Red Hat became one of the biggest names in the IT industry. This chapter will also build up the skills required for your day-to-day tasks in managing a RHEL 9 environment like getting up-to-date information about RHEL 9, and browsing through some alternative platforms that have close ties to RHEL 9.

With the objectives set in place, let us start learning.

Understanding Red Hat Enterprise Linux

RHEL, is a commercial Linux distribution developed and distributed by Red Hat specifically for the enterprise. By enterprise, we mean that Red Hat provides commercial support for the use of RHEL throughout its supportable lifecycle. While RHEL is considered a commercial Linux distribution, it is released as open-source, which means Red Hat provides RHEL to users and grants the rights to use the software, inspect and study the source code of the software, make modifications to the source code, and even distribute the modifications made. This open-source model of software development promotes collaboration, transparency, and sharing. Many of today's modern software development practices are based on the open-source model and released under various licenses compatible with the said model.

RHEL 9 was officially announced at the *Red Hat Summit* on May 10, 2022, with the theme, *Delivering Hybrid cloud innovation from the Datacenter to the Edge*. RHEL 9 was assigned the codename Plow, released to the public on May 17, 2022, and shipped with the Linux kernel 5.14.0 and the GNOME 40 desktop environment. Supported architectures include:

- AMD and Intel 64-bit architectures (x86-64-v2)
- 64-bit ARM architecture (ARMv8.0-A)
- IBM Power Systems little-endian (POWER9)
- 64-bit IBM Z and IBM LinuxONE

RHEL 9, along with succeeding versions in the series (9.x), are shipped with a 10-year lifecycle support. That means we expect the RHEL 9.x series to be officially supported until May 2032. As of this writing, the latest edition of RHEL 9 is version 9.4, was released on April 30, 2024, with updated software packages along with the Linux kernel with support to the hardware architectures listed from the previous paragraph. Succeeding versions of RHEL 9 from 9.1 to 9.10 are considered minor releases with various level of support policies.

Other versions of RHEL still being supported and maintained by Red Hat include the following:

- RHEL 8 (codename Ootpa), was released in May 2019, and supported until May 2029
- RHEL 7 (codename Maipo), was released in June 2014, and supported until June 2024)

Releases prior to RHEL 7 have either been retired or are currently about to end their Extended lifecycle support. We will discuss more about the RHEL lifecycle support in detail in *Chapter 5, Managing RHEL 9 Subscriptions*.

While RHEL 9 is provided as open-source software, Red Hat holds the trademark to its brand, logo, and related properties. These trademarked assets do not fall under the open-source licenses that RHEL 9 source code gets distributed and cannot be re-mixed, modified, nor re-distributed due to their intellectual property and trademarks applied. However, for the rest of what RHEL 9 provides as open-source, it is fair game and there are other Linux distributions available that have used and exercised the freedoms that open-source software provides such as CentOS and Rocky Linux. These RHEL based distributions have proven to be viable alternatives to RHEL for workloads and use cases that do not require the enterprise-grade support that Red Hat provides for RHEL. We will discuss more about these alternatives in the later part of this chapter.

Who is Red Hat?

RHEL is no doubt, a great product, but who exactly is Red Hat, the company that created it, and how did it get started? RHEL got its humble beginnings from 1994 when *Marc Ewing*, then an engineer, started the *Red Hat Linux Project*. It was called Red Hat because *Marc* was known to wear a red lacrosse hat while attending college at the *Carnegie Mellon University* (*CMU*). Red Hat Linux was distributed and up for sale then as **compact discs** (**CDs**), which prompted *Bob Young*, a budding software entrepreneur, to buy much of the CDs in bulk and sell them on retail. This proved to be a lucrative business venture for both. In 1995, the two combined forces and formed *Red Hat Software* with *Young* as the CEO.

Red Hat Linux became so successful in the market that it became a formidable challenger to the operating system business which was then dominated by *Microsoft* with its Windows operating system that spanned across desktop and server environments. It was also a time when the internet started to become more pervasive globally, and this contributed to the adoption growth of Linux with Red Hat becoming the de facto distribution favored among corporate and enterprise organizations on running their platforms and systems online. Beyond the boxed Red Hat Linux CD distribution business, the Red Hat brand was popular on merchandise, with T-shirts, stickers, caps, and even red fedoras! Red Hat continues this tradition of selling merchandise today through the Red Hat *Cool Stuff Store*. You can check out the Red Hat Cool Stuff Store at **https://coolstuff.redhat.com/**. From the store, you can browse through and order various Red Hat branded merchandise like shirts, caps, bags, stickers, mugs, and other desktop toys!

Due to the rapid business growth of Red Hat, it became a public company in 1999 and broke IPO records during that time. Red Hat became the biggest name and the market leader in open-source, and its technology, along with the adopted philosophies of open-source software became mainstream and cemented the company's brand as one of the top technology companies in the world. Being a publicly traded company allowed Red Hat to transition its business model from a boxed software provider to one that focuses on enabling open-source technologies for the enterprise. The following years after the IPO saw Red Hat build its portfolio of products and services that revolved not just with the Linux business but to other areas such as middleware, containers, and security. Below were some notable companies that Red Hat acquired along with the technologies that are now part of the company's product and services portfolio:

- Cygnus Solutions (software tools, acquired January 2000)
- JBoss (middleware, acquired June 2006)
- Gluster (storage, acquired October 2011)
- Ansible (automation management, acquired October 2015)
- CoreOS (containers, acquired January 2018)
- StackRox (container security, acquired January 2021)

On October 28, 2018, *IBM* announced its intentions to acquire Red Hat for $34 billion USD to become an independent subsidiary as part of IBM's Hybrid cloud business. The said acquisition was approved and closed on July 8, 2019.

To date, Red Hat has over 19,000 employees globally, and the business has gone beyond RHEL. Much of the portfolio of enterprise open-source products and services provided by Red Hat revolve around their positioning of the open hybrid cloud, its strategy of enabling enterprises to run and manage software applications regardless of deployment model, be it on-premises, on public cloud, private cloud, and even on edge. Red Hat now focuses on the following core products and services across various industries:

- **RHEL**: The de facto standard for building infrastructure and platforms offering a stable operating system and ecosystem support from various application and platform vendors
- **Red Hat OpenShift**: An enterprise grade Kubernetes container platform positioned for end-to-end application development and delivery across hybrid and multi-cloud environments.
- **Red Hat Ansible Automation Platform**: An enterprise grade platform for automating and orchestrating your IT infrastructure and platform landscape
- **Red Hat Application Foundations**: Delivered as a set of integration tools and platforms that support cloud-native application delivery.

There are more applications and platforms that Red Hat caters to the market but covering some RHEL 9 operations into these other Red Hat platforms, but to get back on topic, let us learn more about RHEL, its importance in your day-to-day IT infrastructure operations and management, and why it is vital to have this solid foundation set. We will use these learnings as our base to build up the skills and knowledge needed as we go along with this book.

Getting additional information on RHEL

Red Hat frequently updates not just the code of RHEL through patches and releases, but also documentation, whitepapers, and a comprehensive knowledge base of use cases, fixes, and alternative methods on configuring and managing a RHEL 9 environment. The portal pages and links below are some of the key resources you will frequently access once you have your RHEL 9 system running:

- **RHEL 9 official documentation**: The RHEL 9 documentation contains detailed information on the release as well as last minute updates, documentation errata, and known bugs of the release. This documentation portal is always updated, especially when a minor release is introduced (case in point, the release of RHEL 9.4). The documentation portal also contains previous versions of RHEL up to RHEL 4. You can access the official documentation pages at **https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9**
- **RHEL 9 trial portal**: You can download a trial subscription of RHEL 9 as you are your organization. This is meant for testing and evaluation prior to getting a full subscription from Red Hat. The trial is meant to last 60 days and is self-supported, meaning Red Hat support is not included here. However, the trial also provides access to other tools like Red Hat Insights and Red Hat Smart Management, which are very helpful when deploying and managing RHEL 9 at scale. If you wish to use RHEL 9 as a developer for application development, we will discuss the Red Hat Developer program in a later chapter of this book. To get access to a trial subscription of RHEL 9, along with the documentation for the trial, go to this page from the Red Hat product website: **https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux/server/trial**
- **Red Hat Customer Portal**: This portal is meant for new and existing customers of Red Hat to get product and documentation updates as well as some basic how to and whitepapers. To gain access to the Red Hat Customer Portal, you need to create a Red Hat account that allows you to check up-to-date information on the product, published errata, **knowledge base** (**KB**), as well as tips on using your RHEL 9 system. You can create your Red Hat account at no-cost here: **https://access.redhat.com/**.
- **Red Hat Developer portal**: If you are into application development, you are in for a treat. The Red Hat Developer portal provides access to application development resources, documentation, and even a sandbox environment for you to test your

code on various Red Hat products including RHEL. You can also get access to a no-cost, self-supported RHEL subscription as an application developer and we will discuss this in detail at a later chapter. To sign up for a Red Hat Developer account, head to this portal page: **https://developers.redhat.com.**

Alternatives to RHEL

In this section, we will discover some well-known Linux distributions that are considered viable alternatives to RHEL depending on the use case and application workloads to run. We will focus on distributions that have a direct connection or relation to RHEL be it on a development perspective or have lineage to RHEL through its source code.

Fedora Linux

Fedora Linux is a community driven Linux distribution that is used as the development starting point for RHEL. In Linux distribution parlance, Fedora Linux is the upstream source, meaning that the source code of Fedora Linux is picked up by Red Hat, stabilized, and made secure to build the RHEL product. Fedora Linux development, release, and support is community driven, and is led and managed by the *Fedora Project*, comprised of volunteers contributing code, documentation, marketing, and artwork. To manage the Fedora Project organization, *The Fedora Council* was established to coordinate the work done by the volunteer community. The project was founded on September 22, 2003, and has been active since. The project by itself, has no legal identity and directly sponsored by Red Hat. The company provides support by providing a pool of dedicated developers, resources needed for development, build tools and infrastructure, as well as marketing oversight. Thus, Red Hat is legally responsible for the directions and actions being adopted and executed by the project.

Because the focus of Fedora Linux is bringing the latest and the best of what a Linux distribution can offer, a new version is targeted for release every six months. Primary architectures supported include x86_64 and ARM aarch64 while alternative architectures provided are PowerPC64LE (IBM Power), and s390x (IBM z Systems). The latest version of Fedora as of this writing is Fedora 37 and was released November 15, 2022, with 5 different versions depending on the deployment use case:

- **Fedora Workstation**: Targeted for desktop and laptop users, the latest version as of this writing (Fedora 40), includes the GNOME 46 desktop environment. It is suitable for Linux hobbyists, application developers, and even power users. A typical Fedora 40 desktop interface experience is shown in *Figure 1.1*:

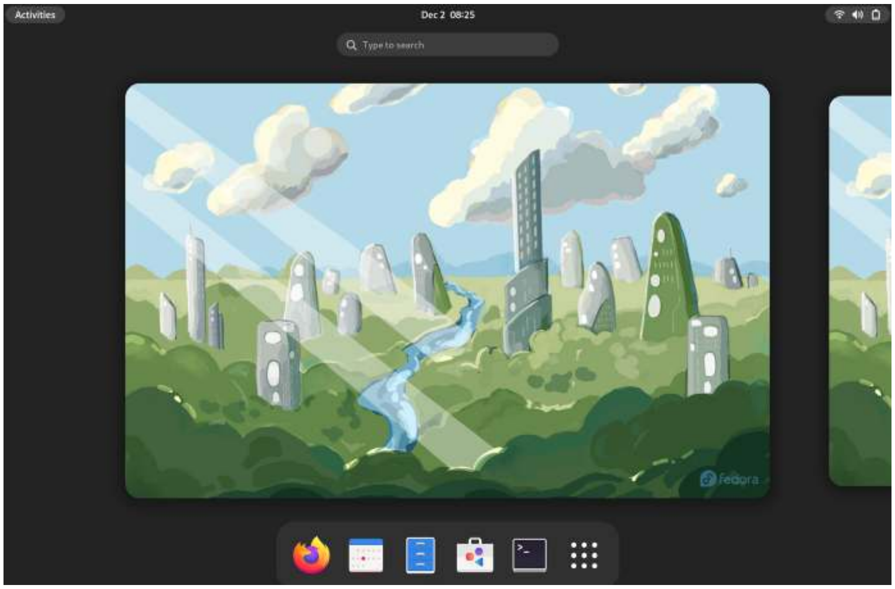


***Figure 1.1**: The author's Fedora 40 desktop with GNOME 46*

- **Fedora Server**: This release does not include a desktop environment and is meant for use on deploying to bare metal servers or as a virtual machine on-premises or on a public cloud. This is a great option for learning system administration with the **command line interface** (**CLI**).
- **Fedora IoT**: A baseline Fedora setup designed for deployment on edge computing platforms and **Internet of Things** (**IoT**) devices.
- **Fedora Cloud**: A minimal Fedora release designed for deployment on a private cloud setup or with public cloud providers such as Amazon Web Services, Microsoft Azure and Google Cloud Platform. On public cloud, Fedora Cloud is provided as an operating system image that can be deployed in a virtual machine instance.
- **Fedora CoreOS**: This release is purpose-built for running containerized workloads.

You can try downloading these Fedora releases as you go through the detailed documentation at **https://getfedora.org/.**

CentOS Linux and CentOS Stream

Started in 2004, **CentOS Linux** is a community driven distribution by *The CentOS Project* that aims to provide the stability of a Linux system that is completely binary compatible with RHEL as it is the upstream source of the distribution. That means CentOS Linux is essentially RHEL without the Red Hat branding, trademarks, and other properties that Red as the upstream source), CentOS Linux became a favorite alternative distribution among organizations that have the capabilities to build and manage a solution stack and seek the rock-solid reliability of RHEL without paying for a subscription to Red Hat for commercial support. In 2014, The CentOS Project became affiliated with Red Hat as an independent project (like Fedora) with its own *CentOS Governing Board* responsible for managing, curating the activities, and marketing The CentOS Project. This model of development for CentOS Linux continued until December 2020, when the CentOS Governing Board announced that development model of CentOS Linux will discontinue at the end of 2021 and introduce a new distribution called **CentOS Stream**. The last supported version of CentOS Linux is version 7 which was based on RHEL 7 and released in July 2014. CentOS Linux 7 inherited the 10-year support cycle for updates and is expected to have maintenance updates until June 30, 2024. As of this writing, the final release of CentOS Linux is version 8 which reached **end of life** (**EoL**) on December 31, 2021. After this, all CentOS Project development focused on CentOS Stream.

CentOS Stream started after the final release of CentOS Linux 8 to continuously deliver the innovations of a Linux distribution and serve as the new upstream source for Red Hat Enterprise Linux development. That means CentOS stream sits in between Fedora and RHEL. This is a radical change from the previous model of development wherein CentOS Linux used RHEL as its upstream source, which made it popular among users and allowed them to get the best of what RHEL provides without the cost of a Red Hat subscription. While this change upset many end users of CentOS Linux, the introduction of CentOS Stream provided an opportunity for organizations, especially those who are invested in the RHEL ecosystem, to become part of the development community that would work and influence on future changes and directions of RHEL development which includes thousands of **independent software vendors** (**ISVs**), hardware **original equipment manufacturers** (**OEMs**), and Red Hat ecosystem partners. As the new upstream source for RHEL, CentOS Stream gives these organizations a platform for building and testing their solution offerings to become compatible with future versions of RHEL.

CentOS Stream 9, the latest release as of this writing, used Fedora as the new upstream source for development. RHEL 9, along with succeeding minor versions (RHEL 9.1, 9.2, onwards), will be utilizing CentOS Stream 9 as an upstream resource. What does this mean for you the sysadmin? CentOS Stream 9 is very much aligned to RHEL 9 (and succeeding versions) development; hence, the expected full support lifecycle is expected to be like that of RHEL 9 until 2027. You can download both CentOS Linux and CentOS Stream installation and source code images at **https://www.centos.org/downloads**.

Rocky Linux and AlmaLinux OS

The discontinuation of CentOS Linux development and the introduction of CentOS Stream resulted to a major backlash among the CentOS community and among them, *Greg Kurtzer*, who happened to be a co-founder of The CentOS Project. *Greg* announced that he would be starting a new project called **Rocky Linux** that was true to the original goals and aspirations of CentOS Linux. The name Rocky Linux was a tribute to early CentOS co-founder *Rocky McGaugh*. Rocky Linux is rebuilt from RHEL sources meaning it is binary compatible and would provide the same reliability and stability of a RHEL release. To date, Rocky Linux has two official releases, Rocky 8 (derived from RHEL 8), with planned EoL by May 31, 2029, and Rocky 9 (derived from RHEL 9), with planned EoL by May 31, 2032. You can check out these Rocky Linux releases at **https://rockylinux.org**.

Like Rocky Linux, **AlmaLinux OS** is a community driven and governed Linux distribution project that aims to provide an enterprise grade, binary compatible with RHEL distribution. Development and management of AlmaLinux OS is owned by the AlmaLinux OS Foundation, a non-profit organization. The foundation owns all assets related to AlmaLinuxOS. Similar to Rocky Linux, AlmaLinux OS is backed by big companies that previously benefited from the success of CentOS Linux, like AWS, Microsoft, AMD, and ARM. AlmaLinux OS release versions are very similar to that of RHEL, and you can download installation images for x86_64, aarch64, and other architectures at **https://almalinux.org**.

Oracle Linux

One of the biggest enterprise software providers in the world, Oracle, introduced in 2006 a 100% binary compatible alternative to RHEL called **Oracle Linux**. This binary compatibility resulted by adopting a similar model of development that was done with CentOS Linux, and with Oracle's depth and reach in the enterprise market, able to provide 24/7 global professional technical support to customers who needed that level of coverage. Having Oracle Linux as part of the Oracle portfolio enabled the company to manage the technology stack and direction from hardware, operating system, and applications. As a RHEL compatible distribution, Oracle can provide an enterprise support path for CentOS Linux users that did not want to move to a Red Hat subscription.

Oracle Linux is made available with two versions based on Linux kernel tuning:

- **Oracle Linux with Red Hat Compatible Kernel (RHCK)**: Uses 100% RHEL sources and is considered binary compatible with all supported RHEL releases.
- **Oracle Linux with Unbreakable Enterprise Kernel (UEK)**: Like the RHCK release but utilizes newer Linux kernel sources with Oracle enhancements applied aligned to Oracle products (hardware and software). These enhancements are heavily used in Oracle's Engineered Systems (full-stack solutions designed to run the Oracle Database and supported applications for mission-critical workloads) and the **Oracle Cloud Infrastructure** (**OCI**).

Oracle Linux versioning is like RHEL version conventions, making it easy for users to immediately check compatibility and changes made from RHEL upstream (especially UEK versions). Users can download the distributions at no cost and, as an option, subscribe to professional technical support for an additional fee. You can check out more information and download Oracle Linux RHCK and UEK versions and go through the documentation, as well as a testing and evaluation guide at **https://www.oracle.com/linux/**.

Conclusion

In this chapter, we learned what RHEL is, what you can do with it as a sysadmin, and the history of Red Hat and its importance in the overall open-source ecosystem. We have also touched on topics on where to get more information about RHEL, along with a list of various RHEL-based alternatives that you can try as we begin our learning journey. Now that we have gone through some basics and importance of RHEL 9, let us discuss the recipes and go into detail to start building your computing environment.

In the next chapter, we will install and set up your RHEL 9 system.

Chapter 2
Setting Up RHEL 9

Introduction

Picking up from our progress in *Chapter 1, Introducing Red Hat Enterprise Linux 9*, we will now start to do some hands-on exercises on installing and configuring a RHEL 9 system. In this chapter, we will start with downloading the RHEL 9 image, decide the target platform for installation, and go through some download options depending on your connection and the level of understanding with regards to the installation options. Next, we will do a step-by-step walkthrough of the installation process and finish the chapter by going through the advanced installation options using the Kickstart tool.

Structure

In this chapter, we will discuss the following topics:

- Downloading and creating bootable media for RHEL 9 installation
- Choosing RHEL 9 installation options
- Disk partitioning and set up for RHEL 9
- Configurations for the RHEL 9 installation
- Automating RHEL 9 installation and configuration with Kickstart

Objectives

In this chapter, we set the following learning objectives. First, you will get the appropriate RHEL 9 image file based on the intended computing architecture that you will install it with. Once downloaded, you will create a bootable USB stick and run this to your computer or virtual machine. Once RHEL 9 is booted, you will configure installation options and proceed with the installation based on your intended use case. You will also learn how to do an automated installation for RHEL 9, which would be useful for multiple installation setups.

In the next section, you will download the RHEL 9 image and set it up as bootable media that we will use for our installation.

Download the RHEL 9 installation image

There are several types of installation methods that we can use for our system, and we will go through them in detail. But first, you need to download an ISO image. An **ISO image** is a disc image based on the ISO 9660 file system that is widely used in the CD-ROM format. DVD images use the **Universal Disk Format** (**UDF**) to accommodate the larger capacity of the media. But in the case of our RHEL 9 image, all of them are available as an ISO image download.

In *Chapter 1, Introducing Red Hat Enterprise Linux 9*, we discussed about the architectures that RHEL 9 supports which include:

- AMD and Intel 64-bit architectures (x86-64)
- 64-bit ARM architecture
- IBM Power Systems (POWER9)
- 64-bit IBM Z and IBM LinuxOne

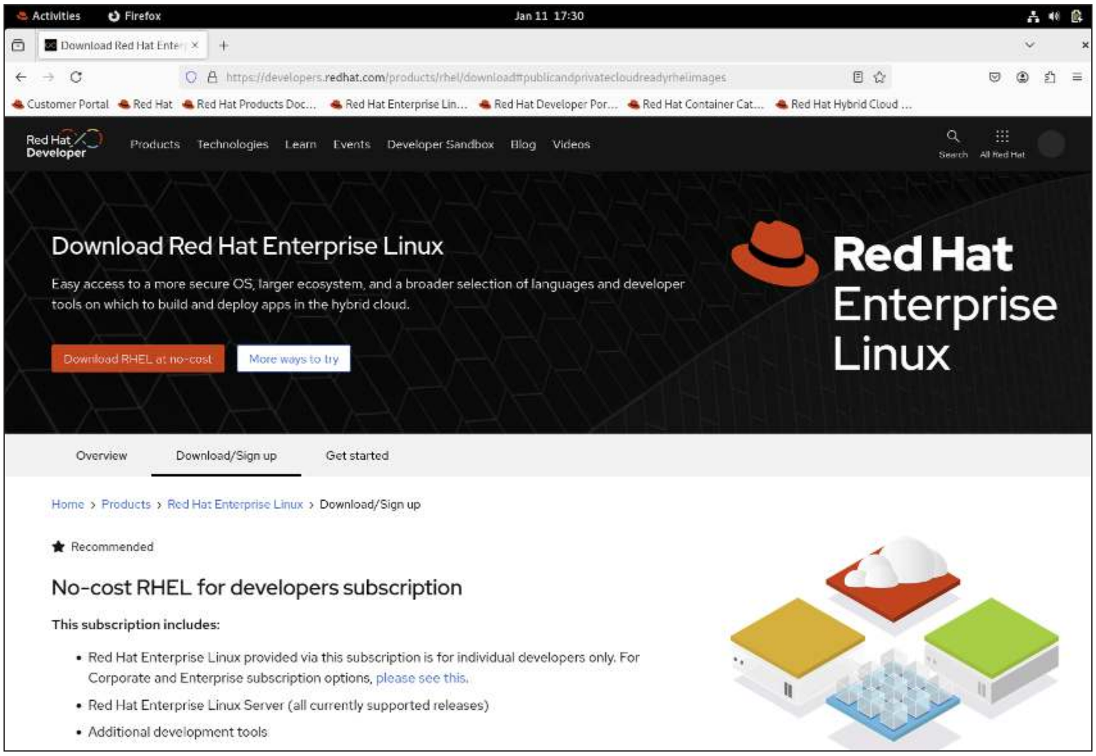
In this book, we will cover only the AMD and Intel 64-bit architectures (x86-64) as it is the most widely used and accessible architecture that we can use. You can also use a virtual machine setup for installation if your host machine has a 64-bit x86 CPU.

But before we continue with downloading anything. Make sure your setup (be it a desktop or laptop computer, server, or virtual machine) have the following minimum requirements:

CPU	Memory	Storage
AMD and Intel 64-bit architectures (x86_64), note that RHEL 9 does not include support for 32-bit x86 CPUs	1.5 GiB for an iso installation; at least 3 GiB for a network installation.	10 GB minimum, 20 GB recommended.
64-bit ARM architecture (AArch64 - ARMv8.x and ARMv9.x)	1.5 GiB for an ISO installation; at least 4 GiB for a network installation.	10 GB minimum, 20 GB recommended.

Table 2.1

Going back to our RHEL 9 image download, open a web browser and head to **https://developers.redhat.com/products/rhel/download**. You will also see some sections in this page asking you to activate your RHEL 9 subscription, but let us ignore this for now as we will go through this in detail in *Chapter 5, Managing RHEL 9 Subscriptions*. Make sure to download only from the official **Red Hat** page and download mirrors to ensure your ISO image is authentic. The download page would look like *Figure 2.1* below:



Download Red Hat Enterprise Linux

Easy access to a more secure OS, larger ecosystem, and a broader selection of languages and developer tools on which to build and deploy apps in the hybrid cloud.

No-cost RHEL for developers subscription

This subscription includes:

- Red Hat Enterprise Linux provided via this subscription is for individual developers only. For Corporate and Enterprise subscription options, please see this.
- Red Hat Enterprise Linux Server (all currently supported releases)
- Additional development tools

***Figure 2.1**: Download page for Red Hat Enterprise Linux*

From the download page, both x86_64 and aarch64 iso image files are available. Focus on the x86_64 architecture for our setup. For each architecture, you have two options to select: a DVD iso or a Boot iso.

A DVD iso supplies the full installation of RHEL 9. The installation process can be graphical, or text based. Much of our installation and configuration recipes will be using the DVD iso. This should be your default download image for many of the recipes in this book.

A Boot iso is a minimal image and used for installing RHEL 9 from a network. This can be used for an advanced installation over the network. We will focus on installing RHEL 9 from the network in *Recipe #4*.

Both DVD and Boot iso images can also be used to fix a RHEL 9 setup in rescue mode (more on this later). So, for this exercise, download the x86_64 DVD iso image of RHEL 9.

The DVD iso image would have the filename `rhel-baseos-9.0-x86_64-dvd.iso` and this would be around 8GB in size. Once you have downloaded it, let us continue to *Recipe #1*!

Recipe #1: Create a bootable USB stick for RHEL 9

For our first recipe, let us build a bootable media from the DVD iso we just downloaded from the Red Hat website. We will be using a USB stick as the installation media for RHEL 9, and we need to write the DVD iso image to it, so it becomes bootable on your target computer or server. Make sure your USB stick is dedicated for this exercise as it will erase everything! Also ensure that it is at least 16 GB in size so that we have enough room to write the DVD iso image. Once you have your USB stick handy, go ahead to *Step 1*.

Step 1: Install the Fedora Media Writer application

The quickest way to do create a bootable USB stick is to use the Fedora Media Writer application to write the DVD iso. You can download the app at **https://getfedora.org**. From that website, select the Fedora Workstation version and from there, the Fedora Media Writer download link for both Windows and macOS computers will be available. Install the version that is right for your use. *Figure 2.2* shows how the **Fedora Media Writer** looks like on a Windows 11 setup:

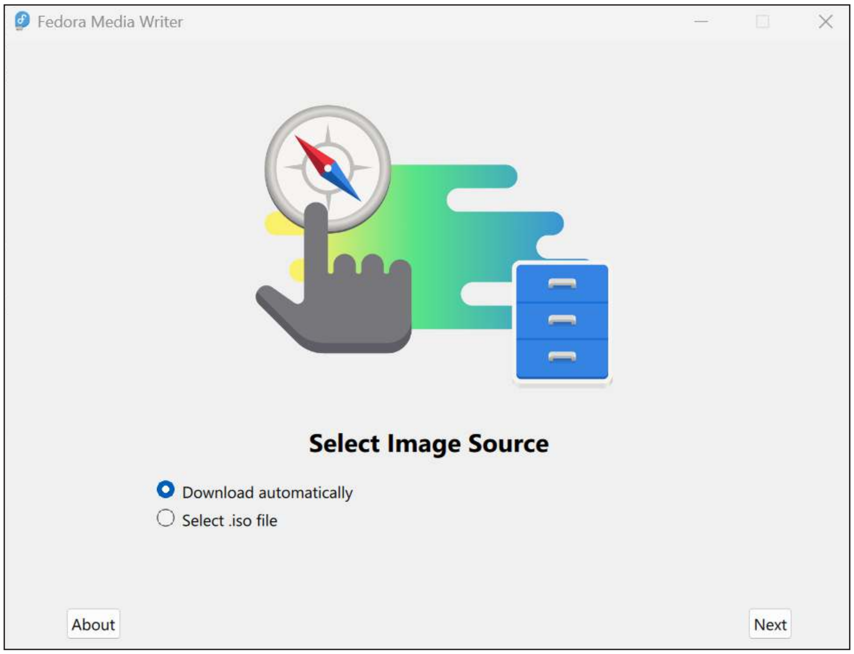


Figure 2.2

You can also use Fedora Media Writer on a Linux desktop. It is already part of the default installation on Fedora 37 Workstation and would look like the Windows and macOS version. If for some reason Fedora Meda Writer is not installed, you can use the **`dnf`** command and install the **`mediawriter`** application. After installing, run **`mediawriter`** from the terminal.

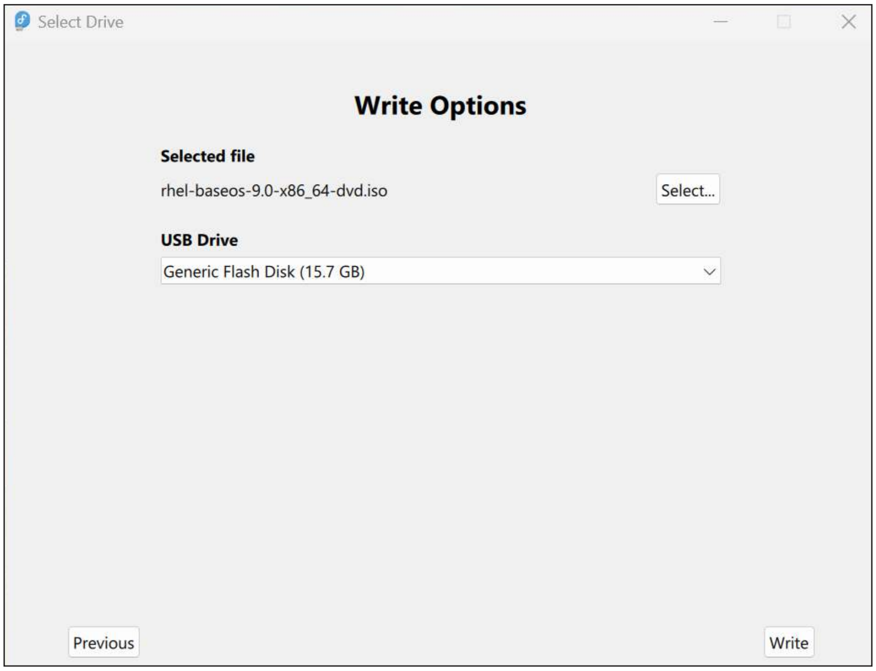
```
1. [jsg@rhel9_bpb ~]$ sudo dnf install mediawriter
2. [jsg@rhel9_bpb ~]$ mediawriter
```

You can also run Fedora Media Writer by clicking on the icon from the list of Applications in your GNOME Desktop.

Tip: You may have noticed that you can use Fedora Media Writer to download different releases of Fedora Linux (Workstation, Server, Emerging Editions, etc.). This was the original intent of the application. The fact that you can use the same for creating your RHEL 9 USB installer is a bonus! You can also use Fedora Media Writer to reformat your USB stick when you do not need the RHEL 9 installer anymore.

Step 2: Create the RHEL 9 bootable USB

The next step is to write your DVD iso image to the USB stick with the Fedora Media Writer. From Select Image Source, choose Select .iso file and make sure to use the **`rhel-baseos-9.0-x86_64-dvd.iso`** image. Also, select the USB drive you want to have the ISO image written on. When done, click on **Write** to start writing to your USB stick.



Write Options

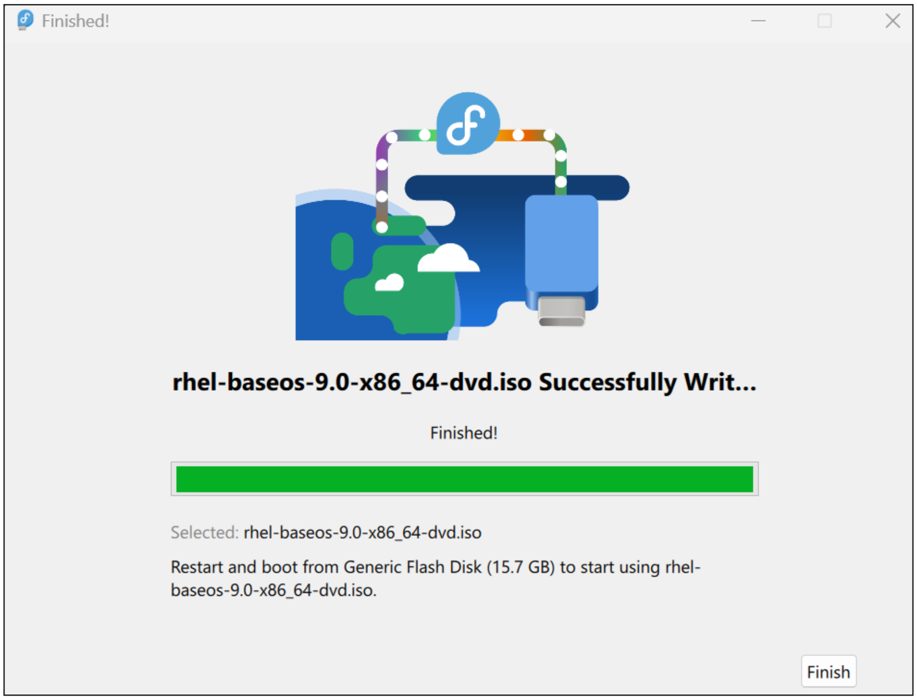
Selected file
rhel-baseos-9.0-x86_64-dvd.iso

USB Drive
Generic Flash Disk (15.7 GB)

Figure 2.3

Fedora Media Writer will then start the process of writing the DVD iso image to your USB stick and this would take a few minutes as it writes the RHEL 9 image then checks the written data (you can grab a cup of coffee while waiting for this to complete). On the background, the USB stick is being resized to a smaller capacity and this will be a dedicated USB installer for RHEL 9.

When the writing and checking process is completed, you will see the status as **Finished!** as shown in *Figure 2.4*:



rhel-baseos-9.0-x86_64-dvd.iso Successfully Writ...

Finished!

Selected: rhel-baseos-9.0-x86_64-dvd.iso

Restart and boot from Generic Flash Disk (15.7 GB) to start using rhel-baseos-9.0-x86_64-dvd.iso.

***Figure 2.4**: Writing and checking of RHEL 9 installer now finished in Fedora Media Writer*

You now have a bootable RHEL 9 installer in a USB stick, and this will be used in the next recipe!

Recipe #2: Choosing between the installation options for RHEL 9

Now that your RHEL 9 USB stick installer is ready, let us start preparing the target computer to have RHEL 9 installed. Since you have a bootable USB stick, you first need to set the computer to boot from USB. This is done by setting up the boot options of your computer in the **Unified Extensible Firmware Interface** (**UEFI**) firmware settings. Some still refer to this as the BIOS but newer computers use UEFI as standard nowadays as it a graphical interface compared to BIOS, so it is much more user-friendly. Setting up the UEFI boot priority of your computer is not covered in this book, but this can be checked with the documentation from the manufacturer of your computer.

Another option for installing RHEL 9 would be as a **virtual machine** (**VM**) in a hypervisor host. You will have multiple options for this and can be done with any operating system. Good choices for a Windows hypervisor host would be Oracle VM VirtualBox, Hyper-V, and VMware vSphere. In a Linux server, you can use virt-manager. But for this exercise, we will be using a simple VM manager application called **Boxes** and is installed in a Fedora Linux workstation as default. Boxes is simple enough to be used in most of recipes in this book and can be used as a lightweight alternative for the recipes in this book. Boxes does not supply advanced options to configure a VM compared to the likes of Oracle VM VirtualBox or VMware vSphere, but it gets the job done. You can have Boxes boot the USB stick installer we made or just load the DVD iso we downloaded from *Recipe #1*. *Figure 2.5* shows how **Boxes** look like in Fedora 37 and loading the RHEL 9 DVD iso image:

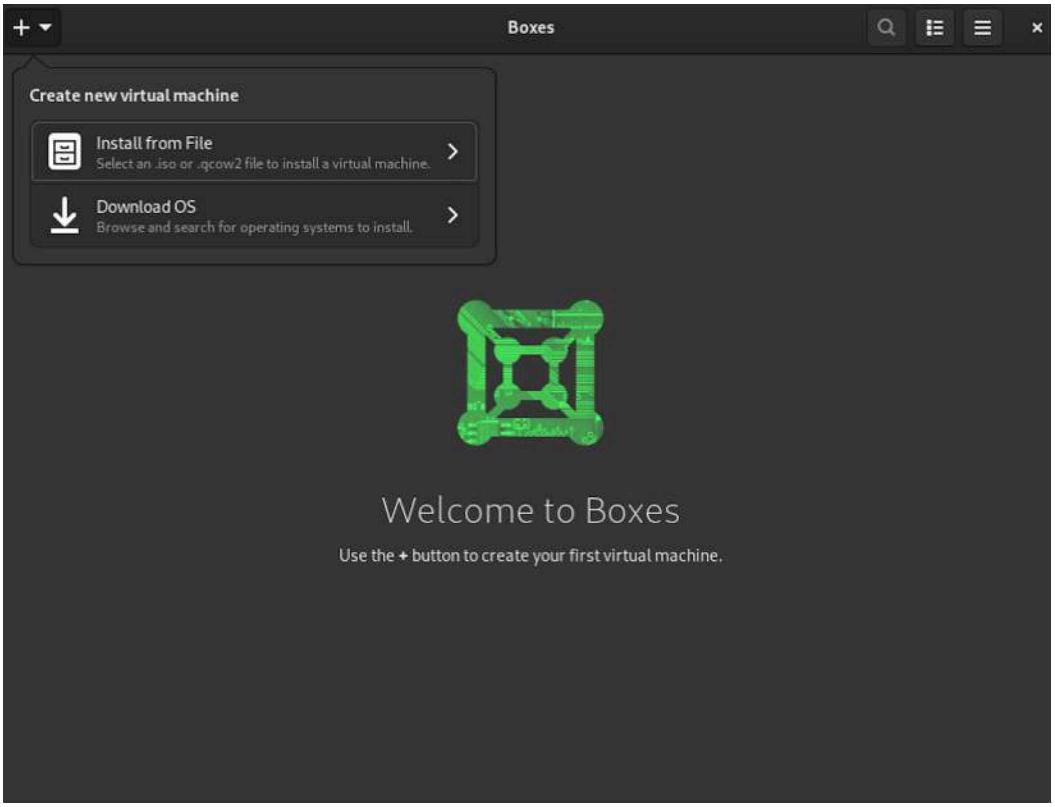
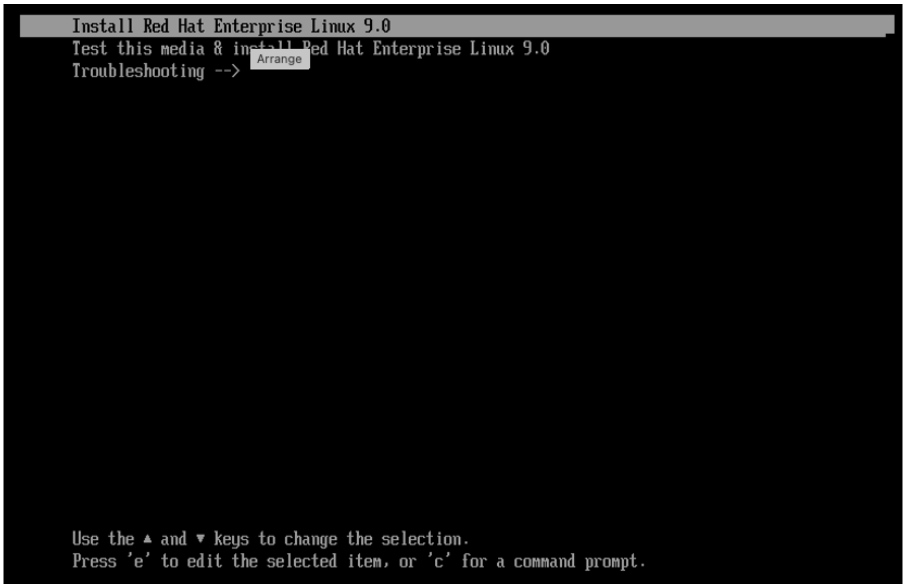

Boxes
Create new virtual machine
Install from File
Select an .iso or .qcow2 file to install a virtual machine.
Download OS
Browse and search for operating systems to install.
Welcome to Boxes
Use the + button to create your first virtual machine.

***Figure 2.5**: Boxes with the RHEL 9 DVD iso image loaded and ready for installation*

This **Boxes** VM setup will be used throughout this chapter but the steps we will discuss would have the same instructions and processes even if you use a different hypervisor host or use the USB stick installer we made. So, let us start!

Step 1: Run the Anaconda installer

RHEL 9 uses Anaconda to install the operating system. Anaconda supplies a text-based or a GUI method for installing RHEL 9. In most cases, you will be using the Anaconda as a GUI much of the time. Every time you run the RHEL 9 installer, you will be greeted with a text-based selection interface called **GRand Unified Bootloader** (**GRUB**) to start the installation along with other options and would look like *Figure 2.6:*

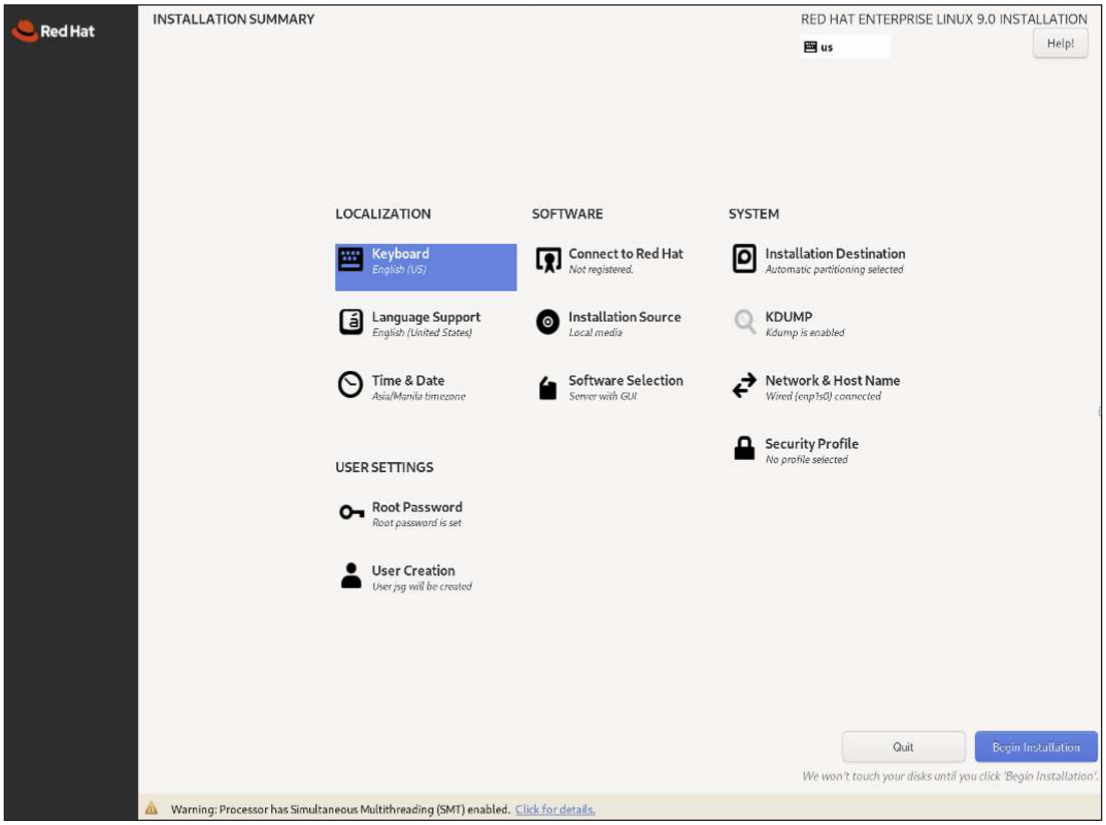


```
Install Red Hat Enterprise Linux 9.0
Test this media & install Red Hat Enterprise Linux 9.0
Troubleshooting -->

Use the ▲ and ▼ keys to change the selection.
Press 'e' to edit the selected item, or 'c' for a command prompt.
```

***Figure 2.6**: GRUB with selections for installing RHEL 9, testing the media, and troubleshooting the set up*

You can also do a test on the installer or image that you would be using for RHEL 9 to ensure that the installer and the packages are not corrupted. The troubleshoot option provides added tools to install RHEL 9 using text mode (instead of GUI), rescue an existing RHEL 9 installation, and run a memory test prior to installing. We will go through these options in later sections of this chapter.

To start the installation, select Install Red Hat Enterprise Linux 9.0 and Anaconda will load and the graphical installer for RHEL 9 will start. Then select the preferred language that you will use during the installation process. An installation summary will then be presented, and each choice can be changed based on our needs. We will go through these options for localization, software, system, and user settings in the succeeding steps. The installation summary page would look like *Figure 2.7:*



INSTALLATION SUMMARY

RED HAT ENTERPRISE LINUX 9.0 INSTALLATION

us

LOCALIZATION

Keyboard
English (US)

Language Support
English (United States)

Time & Date
Asia/Manila timezone

SOFTWARE

Connect to Red Hat
Not registered.

Installation Source
Local media

Software Selection
Server with GUI

SYSTEM

Installation Destination
Automatic partitioning selected

KDUMP
Kdump is enabled

Network & Host Name
Wired (enp1s0) connected

Security Profile
No profile selected

USER SETTINGS

Root Password
Root password is set

User Creation
User jsg will be created

We won't touch your disks until you click 'Begin Installation'.

Warning: Processor has Simultaneous Multithreading (SMT) enabled. Click for details.

***Figure 2.7**: RHEL 9 installation summary*

Now, let us go through some of the setup options that you would need to configure prior to installing RHEL 9. Some of the options need to be set up to have a successful install, while others can be configured post installation.

Step 2: Set localization options

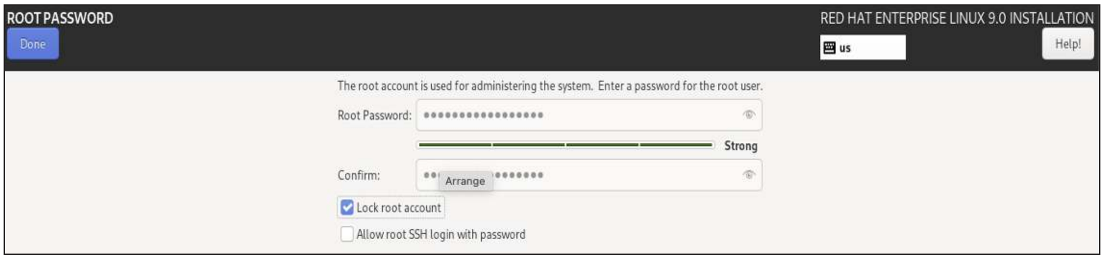
Setting the localization options is straightforward. You can change the following based on your preferences:

- **Keyboard**: RHEL 9 detects your keyboard type at the start but if you wish to add added keyboard layouts, you can select from the options based on your keyboard and preferences.
- **Language support**: You can select more languages aside from the default. You can also do this after the installation.
- **Time & date**: This is also detected based on your UEFI or hypervisor host settings, but you can make changes as well if you wish to use a different time zone. You can also make this change after.

Step 3: Set user settings

From here, we will be setting up the root account password as well as the first user of your RHEL 9 system. We also need to set some options to ensure the accounts are secured by default.

- **Root password**: The root account is used for managing the RHEL 9 set up. You will need to set a password for the root user. Ensure that this is a strong password.
    - To further secure the root user, make sure **Allow root SSH login with password** still is unchecked. Another way to secure the root account is to lock it, so it does not get used for logging into the system directly.
    - Check the **Lock root account** option and click **Done**. *Figure 2.8* shows the root user set up page:



The root account is used for administering the system. Enter a password for the root user.

Root Password: ••••••••••••••••••

Strong

Confirm: ••••••••••••••••••

☑ Lock root account

☐ Allow root SSH login with password

***Figure 2.8**: Creating the root account*

- **User creation**: You can create your first user by adding the Username (the name you will use to login to RHEL 9) and the Password. Like the root account, make your password as strong as possible.
    - If you wish to make this user a sudo user (a user with administrator rights), check Make this user administrator option. This will be the account you will use to login to RHEL 9 after installation.
    - You can also set some advanced options for your user such as:
        - **The home directory**: The default location for your personal files.
        - **User and group IDs**: Useful when you wish to set specific IDs for your user, and which group this user belongs to. This is useful when you set up a multi-user environment and require some access control.
        - **Group membership**: You can specify existing groups where you user belong to or create a new one.

We will discuss in detail about further securing your root and user accounts in *Chapter 14, Security Hardening of RHEL 9*.

Step 4: Set software options

The next step is to set up the software selection that you will install in RHEL 9. This will depend on your use case, but we will go through the options that you get to select. It is also in this section that you need to set up and register your RHEL 9 system, so you can get updates from Red Hat.

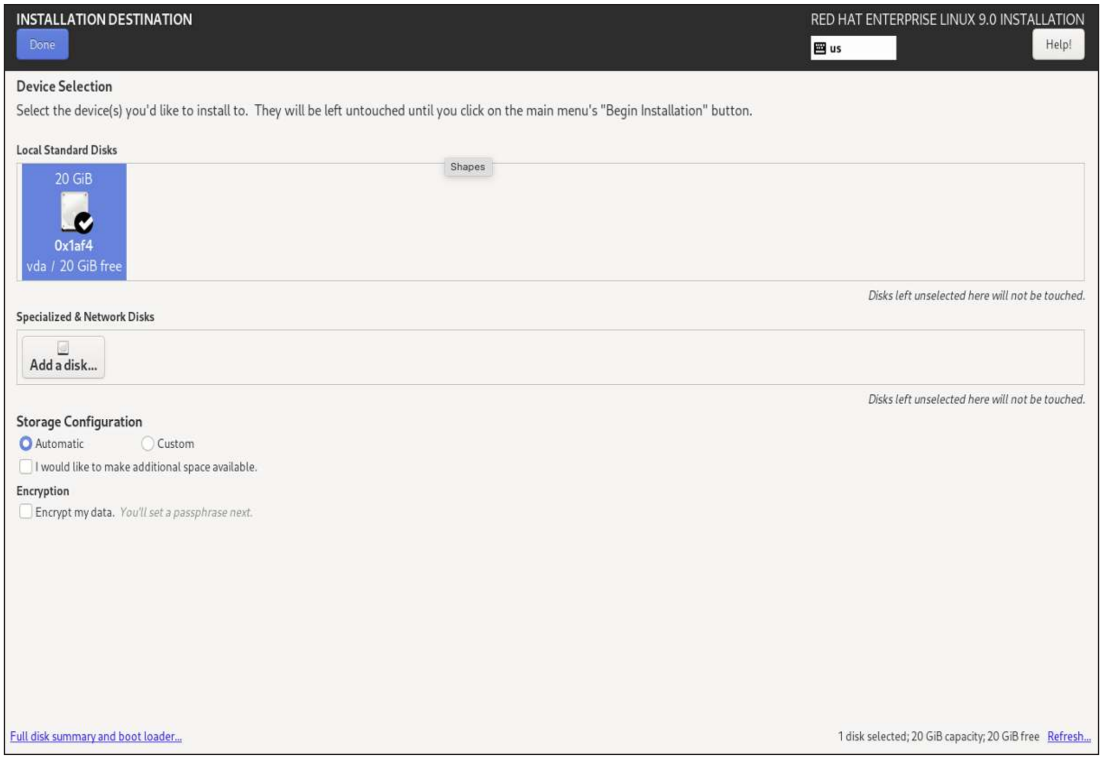
- **Connect to Red Hat**: As discussed in earlier chapters, RHEL is a subscription-based offering that provides enterprise-level support and ensures the continuous development of RHEL along with its surrounding community projects like Fedora Linux and CentOS. You can obtain a no-cost developer subscription of RHEL 9 that gives you the opportunity to develop and test your platforms and software in a testing or non-production environment. If you need to run RHEL 9 in a production environment, you will need to get a commercial subscription from Red Hat. We can opt not to register this RHEL 9 installation and do it after the installation as will go through this in detail in Chapter 5, *Managing RHEL Subscriptions*.
- **Installation source**: This is used for finding the packages for installation in RHEL 9. By default, Anaconda detects the installation media which we used to boot up the installer (in our case, the DVD iso). You can also use this section to test the installer media prior to the actual installation. It is also possible to install RHEL 9 by way of:
    - **Red Hat CDN**: Directly through Red Hat's servers over the internet through its **content delivery network** (**CDN**).
    - **On the network**: If the software packages you need to install are available in a different software download repository. You will need to know the details of the sources in advance prior to using this choice.
- **Software selection**: In this section, you would be able to select the base environment of your RHEL 9 set up based on the requirement that you would be using it for. From the base environment, you can also select more software to install. You can select the base environment with the following use cases:
    - **Server with GUI**: An integrated server with a graphical user interface (this defaults to GNOME). This is the default choice.
    - **Server**: An integrated server with no graphical user interface.
    - **Minimal install**: A basic RHEL 9 system is installed.
    - **Workstation**: A configuration suitable for laptops and desktop PCs.
    - **Custom operating system**: Like the *Minimal install* that allows you to customize the software selection further.
    - **Virtualization host**: this configuration set up ideal for running multiple guest virtual machines.

The software selection page gives a lot of options for installing more software. But this can also be done after the installation so we can focus on installing the base environment that we need to complete this step. We will go through adding more software packages in the succeeding chapters along with the recipes.

Step 5: Set system options

The final set of options to configure is for the storage:

- **Installation destination**: This is one of the most critical portions to configure as we will select the storage device where we will install RHEL 9. In this step, we only have one local disk of 20 GiB in size as part of the virtual machine setup we created in Boxes. Anaconda will detect the local disks installed and gives you the choice to configure the storage automatically. For this step, we will go through the default selections and configure the single disk allocated. Make sure the disk you select will be dedicated to RHEL 9 as it will reformat the whole disk and configure the partitions automatically as shown in *Figure 2.9*. If you wish to customize your disk set up and partitioning, we will go through the process in detail with *Recipe #3: Configure disk partitions for RHEL 9*.



INSTALLATION DESTINATION

RED HAT ENTERPRISE LINUX 9.0 INSTALLATION

Device Selection

Select the device(s) you'd like to install to. They will be left untouched until you click on the main menu's "Begin Installation" button.

Local Standard Disks

20 GiB 0x1af4 vda / 20 GiB free

Disks left unselected here will not be touched.

Specialized & Network Disks

Disks left unselected here will not be touched.

Storage Configuration

[✓ Automatic] Custom

☐ I would like to make additional space available.

Encryption

☐ Encrypt my data. *You'll set a passphrase next.*

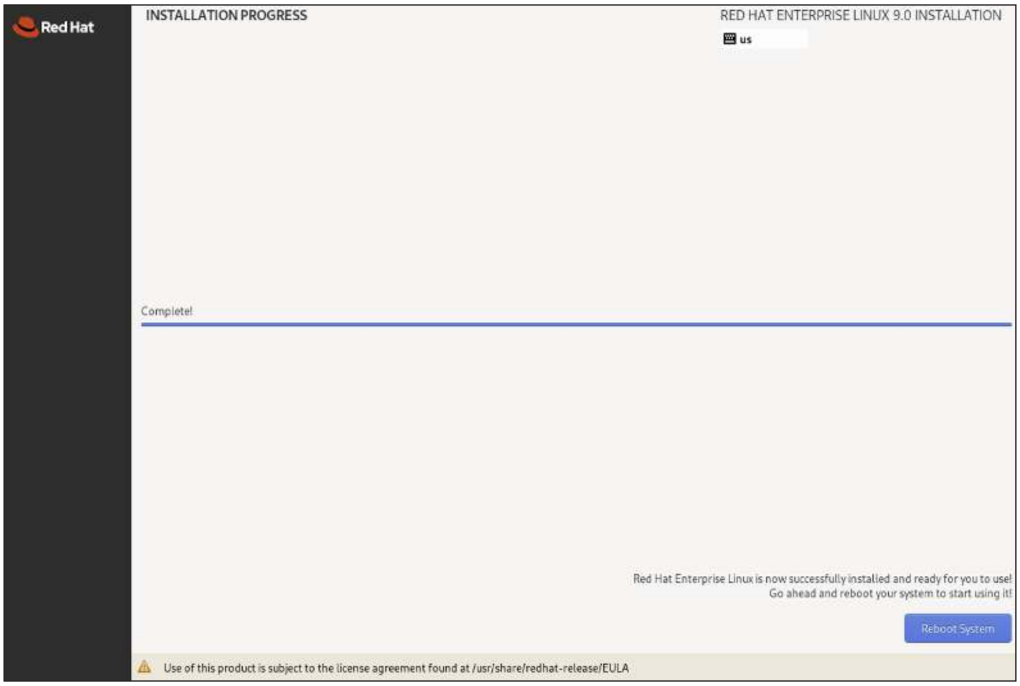
1 disk selected; 20 GiB capacity; 20 GiB free

***Figure 2.9**: Selecting the default disk for RHEL 9 installation.*

- **KDUMP**: The kdump service is installed by default in all RHEL installations. The service allows you to save the contents of the memory for postmortem analysis in the event your RHEL 9 system crashes. This is useful when doing a rescue of the system. This is optional but it is advisable to turn on mission critical servers.
- **Network and host name**: Anaconda will detect existing network interfaces installed on your computer and will default to an interface with an active connection with your network. If you have multiple network interfaces, you can enable or disable the ones you need to configure. You will need to have the details of your network such as the IPv4 or IPv6 addresses, default routes, and the Domain Name System, or DNS. If your network supports **Dynamic Host Configuration Protocol** (**DHCP**), the interface will get configured based on the DHCP settings set. You can add more network interfaces depending on your need such as a **virtual local area network** (**VLAN**) or a Bridge, but you will also need to include more details for Anaconda to set the network settings in the installation. We will use the default network interface detected by Anaconda from the Boxes set up.
- **Security profile**: Some industries require systems to be configured for specific security profiles based on your industry or need. An example would be the Financial Services Industry (like banking and finance) which require systems to adhere to PCI-DSS standards. You can select security profiles based on the standards provided or select none if you plan to implement this later.

Step 6: Complete the RHEL 9 installation

Once all the options you selected in *Step 5* have been applied and confirmed, click on Begin Installation and Anaconda will start installing the RHEL. The installation will happen in the background, and this would be a suitable time to have a little celebration, grab a cup of tea, or coffee, or any beverage of your choice! When the installation progress has completed, you can click **Reboot System** as shown in *Figure 2.10:*



Complete!

Red Hat Enterprise Linux is now successfully installed and ready for you to use!
Go ahead and reboot your system to start using it!

Use of this product is subject to the license agreement found at /usr/share/redhat-release/EULA

***Figure 2.10**: Installation of RHEL 9 completed*

Once the system has rebooted, the GRUB bootloader will load the default RHEL 9 kernel and go ahead with the boot process. After a few minutes, you will be presented with the GNOME login screen and from here, just use the Username and Password you set earlier from *Step 3*. Upon login, you will see a big **9** on your screen like in *Figure 2.11*, and a GNOME welcome notification and from here, do a short tour of the interface:

Figure 2.11

Congratulations! You have just completed the full installation of RHEL 9!

Recipe #3: Configure disk partitions for RHEL 9

From *Recipe #2*, we configured our single disk automatically using Anaconda. But what if we want to configure this manually? Anaconda allows us to customize the disk and create partitions based on recommended configurations.

But before we partition the disk, let us do a quick review of a typical directory structure that you would see in a RHEL 9 system:

- **Root directory (/)**: Pretty much everything in RHEL 9 falls under the **`root`** directory. From /, the directory branches further and takes shape depending on how the partitioning is done.
- **Boot (/boot)**: The **`boot`** directory contains all the files and executables needed by RHEL 9 to start, like the Linux kernel that is needed load RHEL 9.
- **Home (/home)**: All your RHEL 9 local users will have their files and personal directories inside **`/home`**. Example would be **`/home/jsg`** where all my files are inside this home folder.
- **Root user (/root)**: The **`root`** user itself will be in its own folder instead of being part of **`/home`**.
- **Essential binaries (/bin)**: This directory contains all the essential executable files or binaries shared across users in your RHEL 9 system. For example, the bash shell inside the terminal.
- **User binaries and read-only files (/usr)**: Some personal applications of users are located here. An example of this would be the applications installed by a user in GNOME.
- **Lib (/lib)**: These contain libraries needed by applications in **`/bin`** to execute.
- **Var (/var)**: These contain variable data files like log files written by applications.
- **Opt (/opt)**: Optional software packages are installed here. Proprietary applications are generally installed in this location. Let us say you install Microsoft Edge on Linux, the Edge browser executable will be in **`/opt/microsoft/msedge`**.
- **Lost + found (/lost+found)**: In the event of a system crash, files recovered will be placed here.
- **Swap space**: The swap space is created and used as contingency to move inactive resources when the amount of physical memory or RAM in the system is full.

the amount of swap space is the same as that of the physical memory. However, the swap space performance would not be the same compared to adding more physical memory in your system.

Anaconda will create `/`, `/boot`, and a swap space, to install RHEL 9 as the rest of the directories will be created as well during the installation. If you are installing RHEL 9 with multiple disks, it is possible to place certain directories in specific disks, especially those that would potentially grow over time like the `/home` directory.

Another factor to consider when doing a manual partition is the method or type of partition to do. In RHEL 9, we can do the following:

- **Logical Volume Manager (LVM)**: By default RHEL 9 uses LVM to create a partition. Let us say you have 4 physical drives with 1 TB capacity for each drive. LVM will treat the 3 drives as one single volume with a capacity of 4 TB. The physical drives are designated as **physical volumes** (**PV**) and the PVs are then added to a **volume group** (**VG**). Our partitioning and directory structure then will be created in the VG. What is nice about LVM is that we can add and manage more PVs that allow us to increase the size of the VG, making us sysadmins think of total capacity rather than individual disk capacity and management. Thus, RHEL 9 makes disk management easier with LVM.
- **LVM thin provisioning**: This takes a step further to the LVM concept (which is already nice for starters). With LVM thin provisioning, it is possible to create a bigger volume size than the actual volume on the back end. But the partition size is not fully allocated at the start, rather the partition only grows at the time data is written on it. This is done by creating a thin pool in the logical volume. The thin pool can then be expanded based on the growth of storage use. The only downside? You as the sysadmin will have to be proactive in monitoring the use of thinly provisioned storage to ensure that users will have the capacity that you over committed with this method.

  Tip: A good candidate for using LVM thin provisioning is the `/home` directory where users can have bigger storage allocated than what is available in the back end, but they would not be aware until they reach full capacity of their allocation, but by that time you would have already added more PVs to the VG.

- **Standard partitioning**: By making a standard partition, we lock the size of the directory we assign to it. For example, if we allocate 20 GB to `/home`, users will have to share that space and limit it to the total size assigned. A Standard partition is ideal when we consider assigning a physical drive to a particular directory. It is a great way to manage storage, but not the most flexible way to do it.

One last thing to consider about disk partitioning, the file system to use in your set up. By default, RHEL 9 uses the XFS file system. XFS has many features over the other file systems such as:

- Setting directory quotas
- Defragmenting and enlarging an active volume
- Quicker crash recovery
- Support for large file systems, up to 500 TB with a file size up to 16 TB
- Support for LVM and standard partitioning

And these are just a few of what XFS does and makes LVM the logical choice (pun intended) in setting up your disk partition.

Now, let us go through some steps on customizing the storage partitioning in your RHEL 9 system.

Step 1: Customize the storage configuration

After you have selected the disk, you would like to install RHEL 9, from Storage Configuration, select Custom, then click Done.

Step 2: Do a manual partition of the disk

You will then proceed with manual partitioning of the disk.

There are two ways to do the manual partitioning:

- **Automatic creation**: Anaconda will automatically create 3 partitions: The root (**`/`**), boot partition (**`/boot`**), and swap space. You add more partitions by clicking the plus (+) sign at the bottom and select the mount point (such as **`/home`**, **`/var`**, etc.) and input the desired capacity of the mount point. If you wish to restart again, you can either click Discard All Changes at the bottom, then select Reset selections, which brings you back to the previous view, or select the mount point you wish to change or delete then click the minus (**`-`**) sign at the button. When you are satisfied with the partition layout and sizing, just click the blue Done button at the top.

- **Step by step creation**: For this, you will create each mount point from scratch by clicking on the plus (+) button and selecting the mount point you wish to create along with the desired size. This is like the custom partitioning we did after Anaconda did the automatic partitioning. Make sure you have **`/`**, **`/boot`**, and swap space at the very least. As mentioned at the beginning of this recipe, all other mount points will also be created by Anaconda if you only created the three essential mount points mentioned.

Once you are done in creating your partition and mount points, just click the blue Done button at the top of the page and Anaconda will do a check if the mount points and partitioning created are valid. If you are not sure if what you are doing is right, you can always do an automatic partition of which Anaconda will determine the best sizing for your mount points and swap space.

Recipe #4: Conduct an automated installation of RHEL 9 using Kickstart

In the previous recipes, we have installed and configured RHEL 9 with an interactive interface using Anaconda. That method is the most common way to install RHEL 9 in your system and gives you a solid foundation in building the skills needed for succeeding recipes in this book. But what if we had to do the same thing over again to multiple systems? Using Anaconda for every system we install would be cumbersome as you must accomplish every step. For that reason, you need to automate the installation process and RHEL 9 provides a great tool for that called Kickstart.

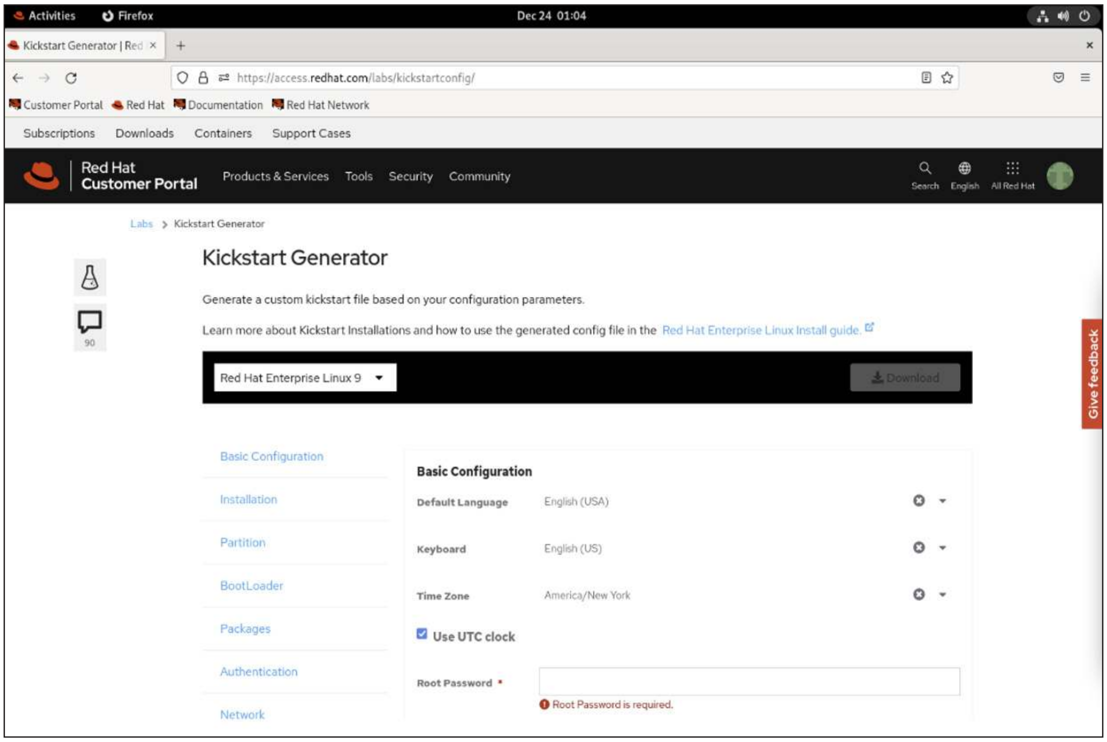
Kickstart allows you to automate the RHEL 9 installation process from configuring system settings, drive partitioning, software to be installed, just to name a few. A Kickstart configuration file can be used by multiple systems at the same time, allowing for RHEL 9 installations at scale. So as a sysadmin, learning how to do automated installation and configuration of RHEL 9 with Kickstart is a valuable timesaver!

A Kickstart installation can be done using the DVD iso we created in previous recipes. We will focus on an automated DVD iso install in this recipe. The other ways we can use Kickstart installation is through the network, by way of HTTPS, FTP, or NFS (we will go through the HTTPS process in the next recipe).

Here are the steps on creating a Kickstart file:

Step 1: Access the Kickstart Generator tool

The fastest way to create a Kickstart file is through the Kickstart Generator tool provided by Red Hat which can be accessed at **https://access.redhat.com/labs/kickstartconfig/** (you may need to create a Red Hat account to access the tool). The tool provides an easy way to create the file by providing answers to the configuration settings that is normally done in the manual install. Head to the link provided and you will be forwarded to the tool interface like *Figure 2.12* below:



Kickstart Generator

Generate a custom kickstart file based on your configuration parameters.

Learn more about Kickstart Installations and how to use the generated config file in the Red Hat Enterprise Linux Install guide.

Red Hat Enterprise Linux 9

Basic Configuration

Default Language English (USA)

Keyboard English (US)

Time Zone America/New York

☑ Use UTC clock

Root Password *

Root Password is required.

Figure 2.12: Kickstart Generator tool

Step 2: Create the Kickstart file with the Kickstart Generator

Now, let us go through some details for creating the Kickstart file:

1. From the drop down box, select **Red Hat Enterprise Linux 9**. You will notice that you can also create Kickstart files from previous versions of RHEL up to version 5.

2. In **Basic Configuration**, set the **Root Password** and ensure that this password is secure, similar to how you did the manual installation previously. You also need to set the **Default Language**, **Keyboard**, and **Time Zone**. By default, the Kickstart installation will restart once completed. You also have the option to perform the installation in text mode (the default mode is Graphical, like the one we did previously).

3. From **Installation**, select the DVD option. If you wish to install via network, NFS, FTP, and HTTP options are also provided.

4. In **Partition**, ensure that Remove all existing partitions is selected. This will erase everything in the target disk where RHEL 9 will be installed (again like how we

did in the previous recipes). Select Clear Master Boot Record, Initialize the Disk Label, and Use automatic partitioning.

5. In **Packages**, select the x86_64 architecture, then select the environment you wish your RHEL 9 installation would be set up. By default, Kickstart would do a Minimal Install which has no GUI. For this recipe, select Server with GUI, like the one we did previously in the step-by-step install. From this section, you can go deeper into installing additional and optional packages, but we will stick with the default we selected.
6. In **Networking**, ensure Use default networking (DHCP) is selected.
7. In **Security**, set SE Linux as Disabled. We will go through SE Linux in detail in *Chapter 14*. If you wish to install a firewall, ensure Enable Firewall option is checked and select the protocol that the firewall would allow (ex. HTTP, FTP, etc.). Note that we can also set up the firewall post installation.
8. For Display, keep this unchecked, as we are installing a server with GUI as selected in Packages.
9. For the remaining options, Pre-installation Script, and Post-installation Script, have this unchecked. These are for advanced installation setup and not covered in this recipe.
10. And once you have completed all the options, the Download button at the top right will become blue and you can download the Kickstart file. Save the file in your system. By default, it will be named **`kickstart.cfg`**. Rename this file to **`ks.cfg`**.

If you open the generated **`ks.cfg`** file from a text editor, it will look like the code snippet below:

```
lang en_US
keyboard --xlayouts='us'
timezone America/New_York --utc
rootpw $2b$10$eDi3thpYopH1vJa4PfpY1uBBtKk3a.oX0Nmtqi680Xmz2OxpF3P4O --iscrypted
reboot
cdrom
bootloader --append="rhgb quiet crashkernel=1G-4G:192M,4G-64G:256M,64G-:512M"
zerombr
clearpart --all --initlabel
autopart
firstboot --disable
```

```
selinux --disabled
%packages
@^graphical-server-environment
%end
```

We can now use this **`ks.cfg`** file for an automated install with Kickstart.

Recipe #5: Install RHEL 9 with Kickstart

Now that we have our generated Kickstart file **`ks.cfg`**, let us now do an automated RHEL 9 installation with our DVD iso.

Step 1: Run the Kickstart file from the boot prompt

1. Boot the system using the *DVD* iso. Then press *ESC* for the boot menu.
2. From **`boot:`**, input the path of the **`ks.cfg`** file depending on the location of the file:
    a. **CD-ROM: `inst.ks=cdrom`**
    b. **ISO/HD: `inst.ks=hd<device>:<path>`**
    c. **URL/Network: `inst.ks=https://<host>/<path>`**
3. Confirm the installation by pressing your *Enter* key.

The Kickstart installation should start, and it would be automated moving forward based on the parameters set in the **`ks.cfg`** file. The installation time would be like the manual installation we did in previous recipes and after a few minutes, RHEL 9 would reboot, and you will be presented again with the login screen.

If you did not get it for the first time, you can try to do the following:

1. Check the generated Kickstart file for possible errors using the **`ksvalidator`** tool from the command line.
2. Make sure the **`inst.ks`** path from **`boot:`** is correct; the installation would not advance if it could not find the **`ks.cfg`** file.

The Kickstart process can be replicated across systems and the advantage of doing this method instead of going through each process is that all your RHEL 9 installations would be consistent throughout.

Conclusion

In this chapter, we went through the end-to-end process of installing and configuring a RHEL 9 system. We started with downloading the right ISO images and then created a bootable USB drive that we used for installing RHEL 9. We also went through the steps of installing RHEL 9 using the Anaconda tool and configured the key parameters needed to complete the install such as setting the root password, choosing the software packages for installation, as well as partitioning the disk. We also went through the basics of building an automated installation and configuration set up using the Kickstart tool and completed an installation with the generated Kickstart file.

In the next chapter, we will learn on how RHEL 9 can be installed and configured on public cloud which would help on accelerating our installations and can even be done on scale, consistently, and securely.

Points to remember

- RHEL 9 installations can be done either using the Anaconda tool for graphical installations or a text-based installation. The process and options for installation would be similar.
- The default installation with Anaconda is Server with GUI. You can start with a Minimal install if you wish to customize your RHEL 9 setup after completing the installation process.
- You can do detailed partitioning using the installation tool, but remember, all the partitions fall under root (**/**) and are created automatically. Much of the demand in storage would come from **/home** and this can be created and managed through LVM thin provisioning.

Chapter 3
Establish RHEL 9 on Cloud

Introduction

In *Chapter 2, Setting Up RHEL 9*, we installed and configured RHEL 9 in a standalone virtual machine. The same process can also be applied to installing it in a bare metal server, desktop, or even a laptop computer if the required minimum specifications of the target hardware that you would use is available. In this chapter, we will focus on deploying a RHEL 9 on a public cloud which accelerates deployment and makes setup and configuration easier as the infrastructure needed is already set up and ready for use. Cloud computing is now the norm in enterprise computing and many, if not all enterprises in the world consider the public cloud as a viable platform to run applications and services. The availability of compute, storage, network, and application services in a pay-as-you-go or consumption model of usage allowed for enterprise platforms and projects to be deployed and configured in a rapid manner, in a safe, secure, and trusted delivery platform. In a public cloud deployment model, companies and enterprises are provided a committed **service-level agreement** (**SLA**) to ensure the computing infrastructure used is always running and available. For enterprises, this model of computing has tremendously increased business activity and rapid deployment of resources means faster time to market for services to the public. Combined with the security and resiliency of RHEL 9, learning how to implement and manage RHEL 9 deployed on a public cloud is a must learn skill for sysadmins.

Structure

In this chapter, we will discuss the following topics:

- Deploying RHEL 9 on the public cloud with AWS, Microsoft Azure, and Google Cloud Platform
- Manage your RHEL 9 deployments with the Red Hat Hybrid Cloud Console
- Migrating your existing Red Hat subscriptions to Microsoft Azure with Red Hat Cloud Access

Objectives

In this this chapter, you will learn to securely deploy RHEL 9 as a virtual machine in a public cloud service of your choice. After deploying, you will get introduced to the Red Hat Hybrid Cloud Console which would enable you to manage your RHEL 9 systems. Finally, you will learn how to migrate your current active RHEL 9 subscriptions and move them to Microsoft Azure with no additional cost to your entitlement.

Recipe #6: Deploy a RHEL 9 virtual machine on Amazon Web Services (AWS)

In this recipe, we will deploy a baseline RHEL 9 system in a VM in AWS. The VM service in AWS is called the **Elastic Compute Cloud** (**EC2**). In AWS, VMs are referred to as instances and we will be using the term throughout the book as a system that is deployed as a VM in a public cloud. Now, let us go through the steps in creating a RHEL 9 VM in AWS:

1. If you do not have an AWS account, yet, sign up for one at **https://aws.amazon.com/**. You will also need to set your billing details to use AWS resources and get RHEL 9 as part of the set up.
2. If you are using a corporate account, check with your cloud administrator to provide you access as your company may already have a set AWS landing zone that defines the governance, security, and cost management requirements of your company.
3. From the AWS console, navigate to Compute and select EC2.
4. Click Launch instance and select an **Amazon Machine Image** (**AMI**) which is the VM image you need to deploy to launch a VM instance.
5. Search for RHEL and select the version that you would like to deploy on EC2. Make sure you are getting official images from AWS so that you would be able to connect them to your Red Hat account for Cloud Access benefit and have the VM monitored by Red Hat Insights.

6. Next, select the size of the VM by going through the Instance types available.
7. Once you have selected the Instance type, configure the basic details such as network setup. You also need to add storage as the VM will only contain the boot disk for RHEL 9 to run on.
8. You can add a Tag to properly manage and identify your RHEL 9 machine from the rest of your AWS resources.
9. Configure the firewall rules for the RHEL 9 VM by setting up a Security group by identifying the ports and IP addresses allowed to access the VM. If your AWS landing zone already has set security groups, choose the appropriate one for your instance.
10. To be able to access the RHEL 9 VM after deployment through a **Secure Shell** (**SSH**), generate a keypair and you will be able to download the private key that you will need to access the RHEL 9 VM over SSH.
11. Finally, review all the configurations you have just set before launching the EC2 instance. After a few minutes, your RHEL 9 VM in EC2 would be available.

Recipe #7: Deploy a RHEL 9 virtual machine on Microsoft Azure

For this recipe, we will deploy a RHEL 9 system as a VM on Microsoft Azure with a standalone or personal account. The steps below should also apply if you are using a corporate account or have an existing **Enterprise Agreement** (**EA**) with *Microsoft*. Work with your cloud administration team if you are going to deploy RHEL 9 with a corporate account as this entails cost, network, and security considerations. Your company may have already set up an Azure landing zone to ensure that all resources deployed in Azure are managed and secured, with proper cost management gates and rules set in place.

1. If you do not have access to Azure yet, create one for free and for personal use at **https://azure.microsoft.com/en-us/**. Once created, login using your Azure credentials.
2. From the Azure portal, create a new VM. You can type `virtual machine` in the search bar at the top, or if an icon of a VM is available, click on it to proceed.
3. Click Create and select Azure Virtual Machine.
4. Select the Subscription that you would use to pay for the Azure resources. Depending on your use, this could either be a pay-as-you-go, a personal subscription (like a Visual Studio Enterprise benefit), or a corporate Enterprise Agreement with Microsoft.

5. For Resource group, create a new one for this exercise. If you have existing resource groups already set, select the one which you want your RHEL 9 Azure VM to be deployed in.

6. Create a unique virtual machine name. The virtual machine name is used in both managing the Azure resource and creating the host name. Make sure this is something you make final as you would not be able to make changes after deployment, although you can change the hostname in RHEL afterwards.

7. Select the Region where you wish to deploy the RHEL 9 VM. This would depend on your location and requirement as you must consider latency issues if the VM is quite far from your physical location.

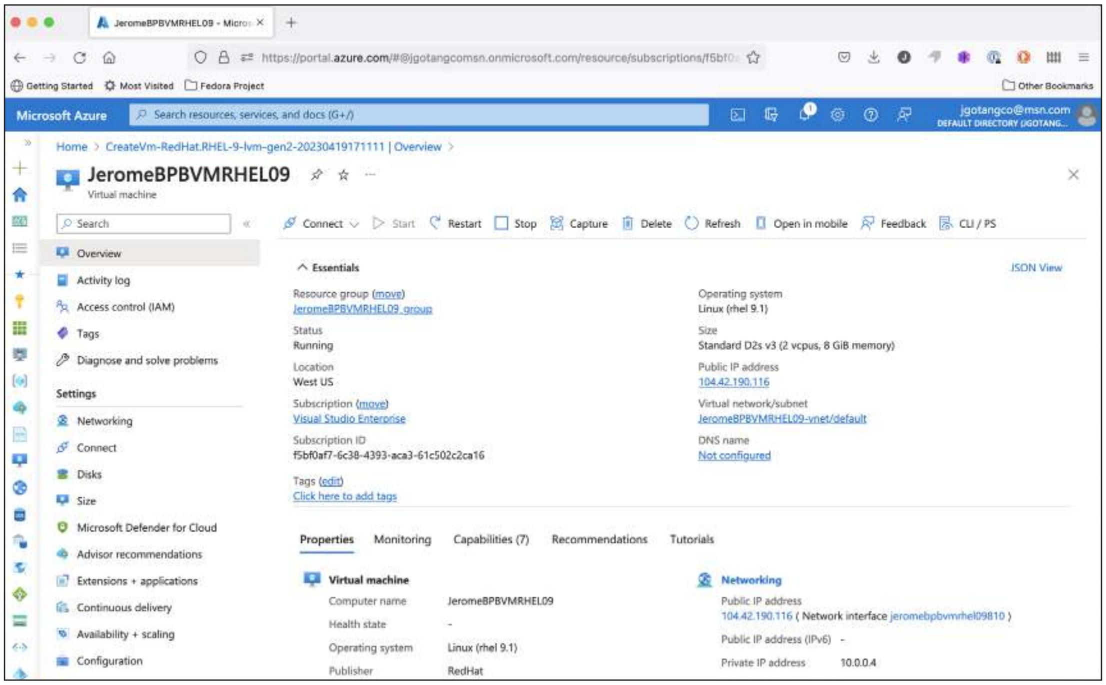
8. For the Image, click See all images and search for the latest RHEL 9 image available. You will be provided a set of recommended VM sizes that you can use for your RHEL 9 system. Select the most appropriate for your use.

9. For the Administrator account, select SSH public key authentication method. Azure will generate a new key pair for the default username azureuser. Keep the private key generated as you will use this to login to RHEL 9 on Azure.

10. For Inbound port rules, use the default port 22 to allow for SSH login. While this may not be the most secure option, for the purpose of the exercise, you can use this method.

11. Once completed, click Review + create. For this recipe, we will use the default settings that would allow us to deploy a basic RHEL 9 server as a VM in Azure. You will be provided with a summary of how RHEL 9 will be deployed on Azure and once validation of settings has completed, click Create.

12. Azure will start deployment of RHEL 9 as an Azure Virtual Machine. Once completed, you can check the newly deployed system like *Figure 3.1* below:



Essentials

Resource group (move): JeromeBPBVMRHEL09_group
Status: Running
Location: West US
Subscription (move): Visual Studio Enterprise
Subscription ID: f5bf0af7-6c38-4393-aca3-61c502c2ca16
Tags (edit): Click here to add tags
Operating system: Linux (rhel 9.1)
Size: Standard D2s v3 (2 vcpus, 8 GiB memory)
Public IP address: 104.42.190.116
Virtual network/subnet: JeromeBPBVMRHEL09-vnet/default
DNS name: Not configured

Virtual machine
Computer name: JeromeBPBVMRHEL09
Health state: -
Operating system: Linux (rhel 9.1)
Publisher: RedHat

Networking
Public IP address: 104.42.190.116 (Network interface jeromebpbvmrhel09810)
Public IP address (IPv6): -
Private IP address: 10.0.0.4

Figure 3.1: *RHEL 9 deployed as an Azure Virtual Machine*

13. When you are done with the exercise and do not need the VM anymore (also to save on cost), just click **Delete** and confirm to proceed with the action. Be mindful of what you are deleting as this would also delete storage, network interfaces, and IP addresses that were configured as part of the deployment.

Recipe #8: Deploy a RHEL 9 virtual machine on Google Cloud Platform

To deploy RHEL 9 on **Google Cloud Platform** (**GCP**) is like how we deploy a virtual machine from the other major public cloud providers like AWS and Azure. GCP provides the **Google Compute Engine** (**GCE**) where you can deploy various types of VM sizes along with operating systems ranging from Linux distributions, container optimized systems, and even Windows server machines. You need to create a GCP account and set up your billing management settings to be able to deploy resources with GCP.

1. From a web browser, head to **https://console.cloud.google.com** and login with your *Google* credentials. Go through the setup process in creating your Google Developers account as well as billing details to be able to provision RHEL 9 and the required resources needed to run the system.

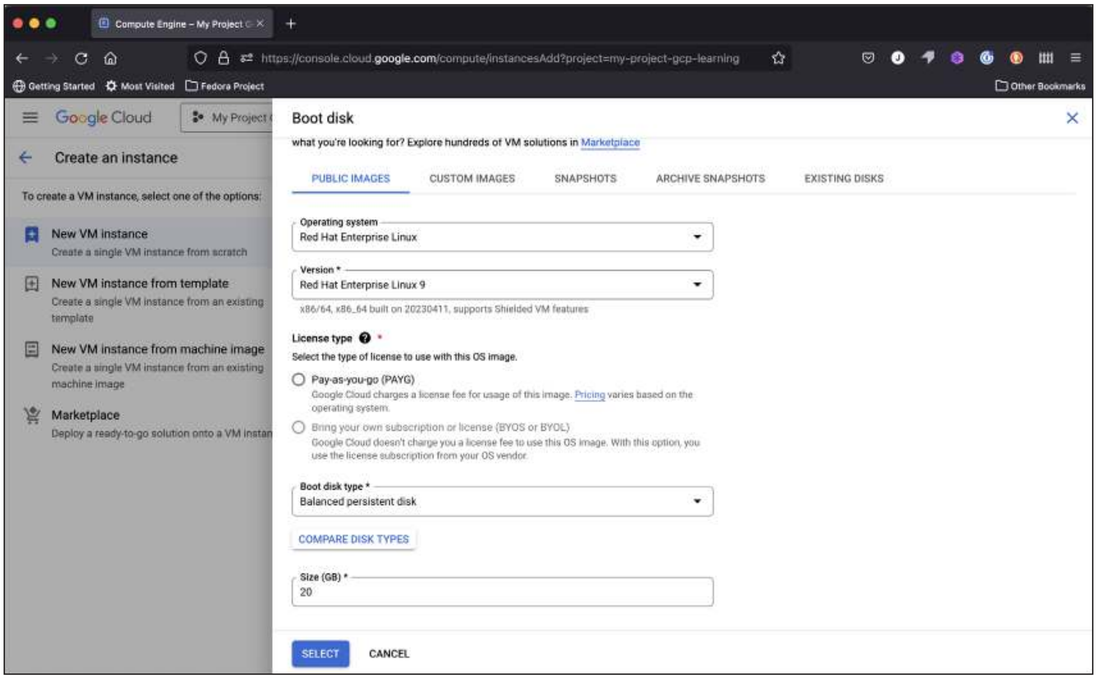
2. From the GCP console, navigate to Compute Engine and select VM instances.

3. Click Create instance and set the details such as instance name, region, and machine size. You will be able to select preconfigured sizes for VMs as well as create a custom one by putting the desired number of cores and memory.

4. In the **Boot disk** section, select **`Red Hat Enterprise Linux`** from the drop down of public images. GCE provides Arm64 and x86 images for RHEL 9.

5. For the **License type**, select **Pay-as-you-go (PAYG)**. RHEL is considered a premium image and the PAYG price will vary depending on the size of the VM instance. Do go through the GCP documentation for more details on premium image pricing.

6. If you wish to use your current subscription and move to GCP, you need to add your GCP account details in the Red Hat Hybrid Cloud Console and activate the Cloud Access benefit. Go straight to *Recipe #10* to get more details if you want to proceed with this. Otherwise, just use PAYG (your credit card or billing account will be charged to both VM resources and RHEL 9 subscription). See *Figure 3.2* below to see the options in using RHEL 9 in GCP:



https://console.cloud.google.com/compute/instancesAdd?project=my-project-gcp-learning

Create an instance

To create a VM instance, select one of the options:

New VM instance
Create a single VM instance from scratch

New VM instance from template
Create a single VM instance from an existing template

New VM instance from machine image
Create a single VM instance from an existing machine image

Marketplace
Deploy a ready-to-go solution onto a VM instan

Boot disk

what you're looking for? Explore hundreds of VM solutions in Marketplace

Operating system: Red Hat Enterprise Linux

Version *: Red Hat Enterprise Linux 9

x86/64, x86_64 built on 20230411, supports Shielded VM features

License type *

Select the type of license to use with this OS image.

○ Pay-as-you-go (PAYG)
Google Cloud charges a license fee for usage of this image. Pricing varies based on the operating system.

○ Bring your own subscription or license (BYOS or BYOL)
Google Cloud doesn't charge you a license fee to use this OS image. With this option, you use the license subscription from your OS vendor.

Boot disk type *: Balanced persistent disk

Size (GB) *: 20

***Figure 3.2**: Selecting between PAYG and BYOS for RHEL 9 in the GCP console*

7. In **Boot disk type** and Size, select the type of boot disk you would like this RHEL 9 VM to run on and the size of the boot disk.

8. Set the Identity and API access, Firewall, and Advanced options (for networking, storage, security, etc.) depending on your requirements. You can select the defaults at

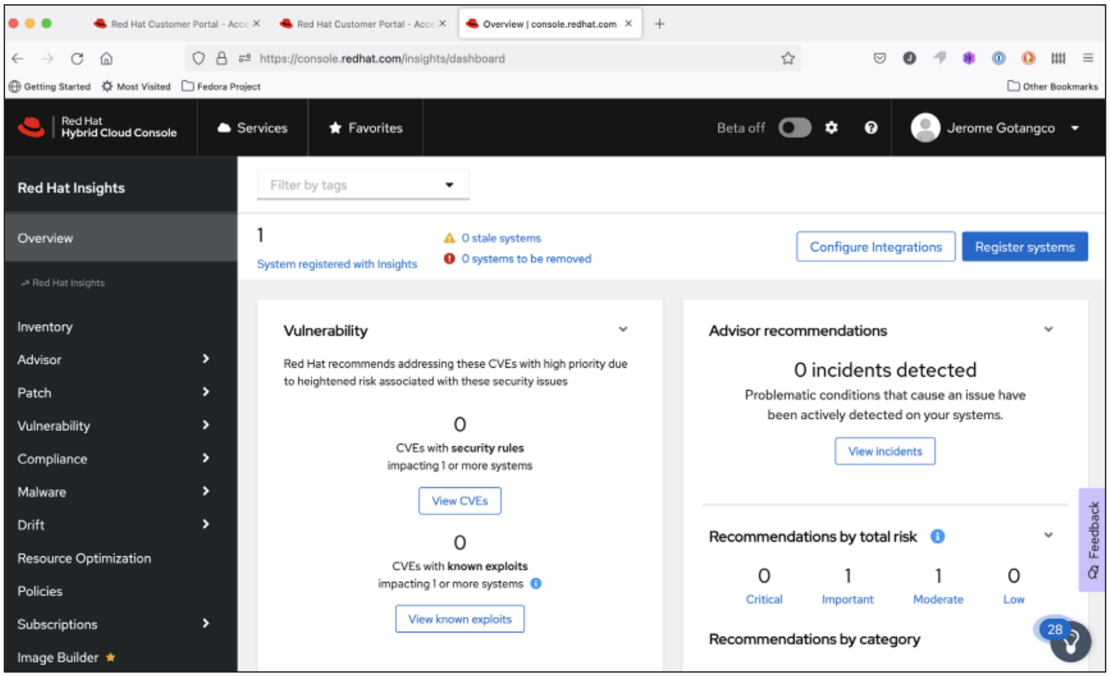
9. Click Create and after a few minutes, a RHEL 9 system would be deployed in GCP. You can check the status of the deployment through the GCP console and even SSH on the VM through the browser.

10. Once you are done in the testing, you can stop or delete the VM instance by selecting it. If you select stop, the VM session will be terminated but will still be available for later use as the resources are kept (boot disk, network settings, etc.). If you select Delete, all resources used in creating the VM instance will be shut down and permanently deleted. So, keep this in mind when managing your VM instances in GCP.

Recipe #9: Manage your RHEL 9 systems with the Red Hat Hybrid Cloud Console

A useful tool to have a holistic view of all your RHEL 9 instances and deployments would be through the **Red Hat Hybrid Cloud Console** which can be accessed online at **https://console.redhat.com**. Let us go through some steps on using the Red Hat Hybrid Console and check the status of our deployed RHEL system registered on Red Hat Insights.

1. To access the **Hybrid Cloud Console**, login using your Red Hat account click on Red Hat Enterprise Linux and select **Red Hat Insights**. The console would look like *Figure 3.3* below:



1
System registered with Insights
0 stale systems
0 systems to be removed

Vulnerability
Red Hat recommends addressing these CVEs with high priority due to heightened risk associated with these security issues
0
CVEs with **security rules** impacting 1 or more systems
0
CVEs with **known exploits** impacting 1 or more systems

Advisor recommendations
0 incidents detected
Problematic conditions that cause an issue have been actively detected on your systems.

Recommendations by total risk

Critical	Important	Moderate	Low
0	1	1	0

Recommendations by category

Figure 3.3

2. As you can see from *Figure 3.3*, Insights provides a complete view of the status of your RHEL systems from needed patches, known vulnerabilities, compliance, etc. If there are no RHEL 9 systems shown in Insights, you will have to register the systems to make it visible to the Hybrid Cloud Console. To do this, click on the **Register systems** button.
3. Select the version of RHEL that you would like to register. For RHEL 9, select RHEL 8.4+ and select how the systems are managed. You can choose between Red Hat Subscription Manager, Red Hat Satellite, and a RHEL deployed in public cloud. Each selection will give you different instructions on registering your RHEL system.
4. For this exercise, since we have been using Red Hat Subscription Manager, we will use that option. Log in to the RHEL 9 system that you need to connect to Insights and open up a console and do the following command as a **`sudo`** user.

```
[jsg@rhel9_bpb ~]$ sudo insights-client --register
```

5. Once completed, refresh the Red Hat Hybrid Cloud Console on your browser, head to Insights, and select Inventory. Your RHEL 9 system should now be available for viewing and auditing using the tools provided by Insights.
6. If you would like to connect all your systems in one command, do the following command instead using the Red Hat connector command:

```
[jsg@rhel9_bpb ~]$ sudo rhc connect -u <username> -p <password>
```

7. Then refresh again the Hybrid Cloud Console and your RHEL 9 systems should be available in the Inventory tab.

Recipe #10: Migrate your current Red Hat subscription to cloud with Red Hat Cloud Access

The Red Hat Cloud Access program is an additional subscription benefit provided by Red Hat that allows you to move your active RHEL subscriptions to cloud. Red Hat Cloud Access is already included in your existing active subscription at no additional cost, plus you get the flexibility and freedom to choose how and where to use your RHEL product, be it on-premises, on public cloud, or on a supported service provider. You still get full support from Red Hat will the subscription is active.

Before using the Red Hat Cloud Access benefit, Red Hat provides some guidance to determine eligibility of your subscription to avail yourself of the benefit. These include:

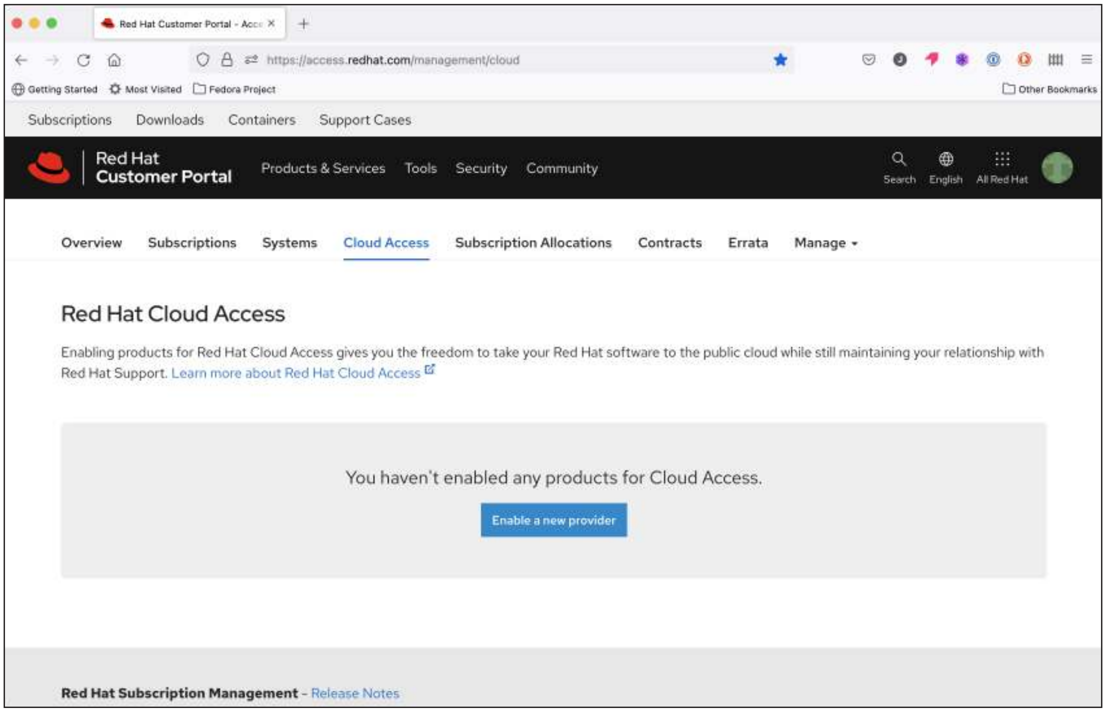
- An active RHEL subscription
- The subscription is currently not used or deployed

- The RHEL product is compatible for use in the supported target cloud environment

Red Hat provides a complete documentation on the program overview and benefits of the Red Hat Cloud Access Program at the Red Hat Customer Portal and can be accessed at **https://docs.redhat.com/en/documentation/subscription_central/1-latest/html-single/red_hat_cloud_access_reference_guide/index**.

Now let us go through the steps on getting your RHEL subscriptions enabled with Cloud Access benefit:

1. With your Red Hat account, head to the Red Hat Cloud Access portal at **https://access.redhat.com/management/cloud** and login with your Red Hat credentials. When logged in, the **Red Hat Cloud Access** portal would be like *Figure 3.4* below:



Red Hat Cloud Access

Enabling products for Red Hat Cloud Access gives you the freedom to take your Red Hat software to the public cloud while still maintaining your relationship with Red Hat Support. Learn more about Red Hat Cloud Access

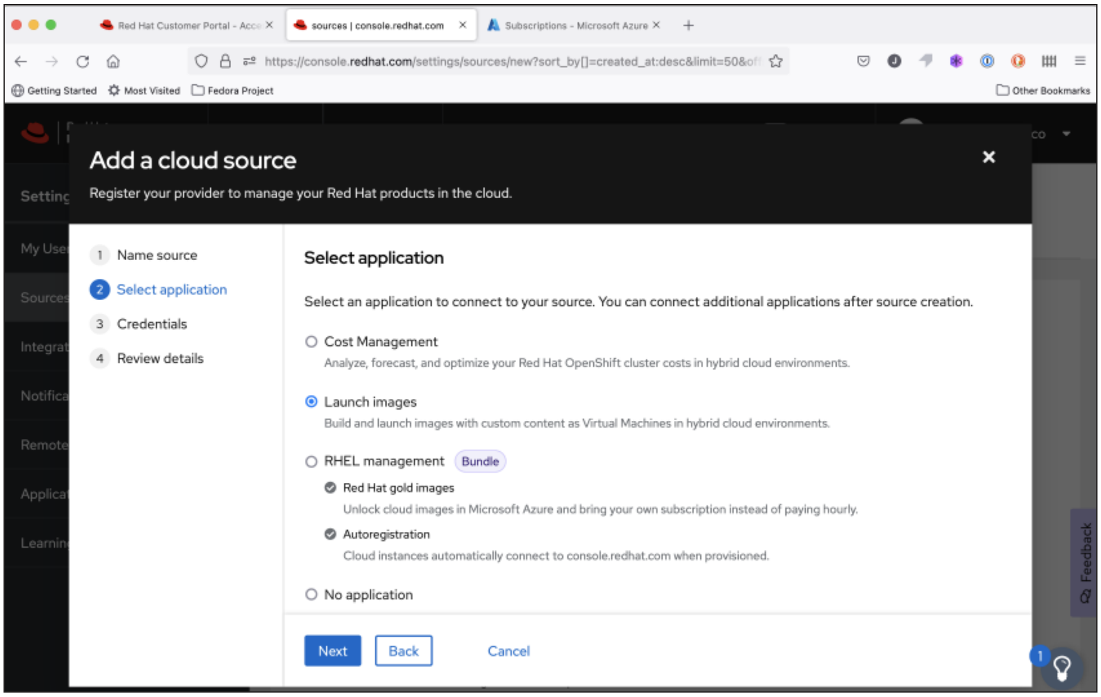
You haven't enabled any products for Cloud Access.

***Figure 3.4**: Red Hat Cloud Access in the Red Hat Customer Portal*

2. Click the **Enable a new provider** button and select from the drop down box on the certified cloud and service provider that you wish to use the Cloud Access benefit.

3. Depending on the provider you have selected, you have choices on how to connect your current Red Hat subscription to the service provider. In the case of AWS and Microsoft Azure, you can either connect your cloud provider through the Red Hat Hybrid Cloud Console at **https://console.redhat.com/settings/sources**, or you can manually add your accounts. For GCP, you must manually add your Google

guide you through the steps as shown in *Figure 3.5* below for connecting an Azure subscription to the Red Hat Hybrid Cloud Console:


Red Hat Customer Portal - Acc
sources | console.redhat.com
Subscriptions - Microsoft Azure
https://console.redhat.com/settings/sources/new?sort_by[]=created_at:desc&limit=50&of
Getting Started
Most Visited
Fedora Project
Other Bookmarks
Feedback

Add a cloud source

Register your provider to manage your Red Hat products in the cloud.

1 Name source
2 Select application
3 Credentials
4 Review details

Select application

Select an application to connect to your source. You can connect additional applications after source creation.

- Cost Management
  Analyze, forecast, and optimize your Red Hat OpenShift cluster costs in hybrid cloud environments.
- [✓ Launch images]
  Build and launch images with custom content as Virtual Machines in hybrid cloud environments.
- RHEL management Bundle
  - Red Hat gold images
    Unlock cloud images in Microsoft Azure and bring your own subscription instead of paying hourly.
  - Autoregistration
    Cloud instances automatically connect to console.redhat.com when provisioned.
- No application

Next Back Cancel

Figure 3.5: *Adding a Microsoft Azure subscription in the Red Hat Hybrid Cloud Console*

4. To manage your RHEL instances across cloud environments, select **Launch images** to be able to build and launch images with custom content and deployed as virtual machines. You must add the relevant subscription details from your cloud provider to complete this step.

5. Once completed, the subscription details and credentials will be validated and once completed, you will be able to utilize your active RHEL subscriptions on cloud using Red Hat Cloud Access and you will only pay for the virtual machine resources and other applications used in your deployment.

Conclusion

In this chapter, we have learned that RHEL 9 can also be deployed and configured for the public cloud, and this allows for more flexible use of the platform for your applications as it gives you choice on the public cloud provider, cost, and even a huge selection of instance sizes that you can use for deployment. The flexibility of using RHEL 9 on cloud is further strengthened with the option of using your current RHEL 9 subscriptions and activating the Cloud Access benefit that would allow you to offset the cost of the operating system and just pay for the use of cloud resources deployed.

By adding your deployed RHEL 9 instances on Red Hat Insights, this gives you complete coverage to ensure that your systems and applications deployed would be continuously monitored and secured by providing proactive guidance, patches, and errata.

Points to remember

- Always check the configurations and settings you have applied prior to deploying or launching your RHEL 9 system on cloud as some of the settings such as the VM name and storage cannot be changed after unless you will have to redeploy again which would cost you money.
- If you are using your company's public cloud resources, check with your cloud administrator if there are available resources and budget for building your VM instances in the public cloud. Doing this first will avoid potential overspend of your company's IT operating budget, at the same time, will ensure that you are launching and deploying cloud resources as prescribed in your company's landing zone policy.
- Take advantage of the Red Hat Cloud Access benefit and add your supported providers in the Red Hat Hybrid Cloud Console so you can use your existing subscriptions on cloud and just pay for the cloud resources you use.

Chapter 4
Miscellaneous Configurations of RHEL 9

Introduction

In *Chapters 2* and *3,Setting Up RHEL,* and *Establish RHEL 9 on Cloud,* we have set up and installed RHEL 9 based on the deployment model needed. First, we went through the process of implementing RHEL when done on-premises, be it on bare metal or as a VM. Then we also learned about the other methods we can implement RHEL 9 on the major public cloud providers, AWS, Microsoft Azure, and Google Cloud Platform. In most cases, the default settings after installation will work for common workloads as Red Hat provides a secure, enterprise grade working environment by default. But there are times that we must do some changes or further improve the set up beyond the default settings. We will go through some miscellaneous, but essential configurations in RHEL 9 to improve your set up.

Structure

In this chapter, we will discuss the following topics:

- Upgrading a current RHEL system to RHEL 9
- Managing user accounts and permissions
- Configuring the system date and time

- Setting up a firewall
- Enabling and disabling system services

Objectives

In this chapter, we will start by going through the detailed steps in upgrading an existing RHEL system to RHEL 9. After which, we will login to the system and manage user accounts and roles. Depending on the installation setup, we may have to configure the proper date and time settings of your RHEL 9 system. We will also go through the basic steps in setting up the network interfaces and connecting your RHEL 9 system to your network. To secure the system, we will also go through the process of setting up a basic firewall service in RHEL 9, as well as enabling and disabling the necessary services depending on your need.

Now let us go through these recipes to have your RHEL 9 system ready for workloads!

Recipe #11: Upgrade from a previous version of RHEL

It is possible to upgrade an existing RHEL installation to RHEL 9 if it still supported by Red Hat. However, certain conditions exist:

- **RHEL 8**: You can do an in-place upgrade to RHEL 9 using the LEAPP utility. LEAPP is a tool accessible via CLI and is a framework for modernizing operating systems and applications.
- **RHEL 7 and below**: there is no direct upgrade path to RHEL 9 with these versions. To upgrade to RHEL 9, you must do an in-place upgrade to RHEL 8 (where supported), prior to doing another upgrade to RHEL 9, like how it is described in *#1*.

For this recipe, let us go through the steps in upgrading an existing RHEL 8 installation to RHEL 9:

1. **Check the system requirements**: Make sure that your system meets the minimum requirements for RHEL 9. To go through the system requirements, head back to *Chapter 2, Setting Up RHEL 9,* and review the specifications needed to do an upgrade.
2. **Do a data backup**: It is important to backup all important and relevant data and configurations prior to doing the upgrade to RHEL 9. This includes files in your **`/home`** directory to ensure that the data is secure in case the upgrade fails. You can use the **`tar`** utility to do a simple backup of the **`/home`** directory with the command:

```
[jsg@rhel9_bpb ~]$ tar czf /mnt/backup/mybackup.tar.gz /home/
```

The **tar** utility will create a compressed tar archive called **mybackup.tar.gz** located in **/mnt/backup** and will consist of files and directories from the **/home/** directory.

3. **Update the RHEL 8 system prior to upgrade**: Make sure that your RHEL 8 system has all the latest, supported security and software updates prior to the upgrade. From the console, do a **dnf** update:

```
[jsg@rhel9_bpb ~]$ sudo dnf update
```

4. **Disable third-party software repositories**: To ensure a smooth upgrade process, temporarily disable third party repositories with this **subscription-manager** command:

```
[jsg@rhel9_bpb ~]$ sudo subscription-manager repos --disable=*
```

5. **Install the LEAPP utility and run the pre-upgrade check**: The recommended tool to upgrade RHEL 8 to RHEL 9 is **leapp**. Install this tool with **dnf**:

```
[jsg@rhel9_bpb ~]$ sudo dnf install leapp
```

Once installed, do a pre-upgrade check:

```
[jsg@rhel9_bpb ~]$ sudo leapp preupgrade
```

During the pre-upgrade check, **leapp** will collect data in the RHEL 8 system, check upgradeability, and generate a report located at **/var/log/leapp**. The report will provide a list of issues found and provide solutions to fix the issues. It will also give a recommendation whether to proceed with the upgrade or not.

6. **Upgrade to RHEL 9 using LEAPP**: Once your RHEL 8 system has passed the **leapp** pre-upgrade check, you can proceed with the upgrade to RHEL 9 with the **leapp upgrade** command:

```
[jsg@rhel9_bpb ~]$ sudo leapp upgrade --target 9.0
```

During the upgrade, **leapp** will download all the necessary packages and executes the rpm upgrades in the background. Once completed, you can manually reboot to refresh the system. You can also opt to include a **--reboot** option from the **leapp** upgrade command so that the system will reboot automatically upon completion.

7. **Check and confirm that the upgrade is successful**: You can do the following commands to verify that you have successfully upgraded your RHEL 8 system to RHEL 9:

```
[jsg@rhel9_bpb ~]$ sudo cat /etc/redhat-release
```

The output of the **cat** command should show that you are now on RHEL 9. Check if your RHEL 9 system is connected to the network as well by doing some basic **ping** commands from the terminal or connect to a nearby server via **ssh** to validate the connection as well. Once confirmed, update your new RHEL 9 system to be get all newly released software and security updates.

Congratulations on completing an in-place upgrade to RHEL 9!

Recipe #12: Manage user accounts and access permissions

In this recipe, we will go through the basics of creating and managing user accounts and permissions in RHEL 9. These series of task are essential for sysadmins as they are part of the security foundation in managing the system. Now, let us go through some tools and commands available to manage user accounts and access permissions:

- **Creating user accounts**

  The **`useradd`** command is used to create a new user account in RHEL 9. To use it, open a console session and create a new user account as a **`sudo`** user with the desired username account:

  ```
  [jsg@rhel9_bpb ~]$ sudo useradd jeromeg
  ```

  In the example above, the **`jeromeg`** user home directory will be created at **`/home/jeromeg`** by default. To specify a different home directory for the user account, add the *-d* flag option followed by the path to the desired directory.

  An alternative tool that can be used to create a new user is the **`adduser`** command. The main difference of **`adduser`** is that it allows you to set the password, other user settings such as groups, and automatically creates the home directory:

  ```
  [jsg@rhel9_bpb ~]$ sudo adduser jeromeg
  ```

- **Setting user passwords**

  With the **`useradd`** command, we created a new user **`jeromeg`** but we did not set a password yet. Use the **`passwd`** command in the console to set a password for the new user:

  ```
  [jsg@rhel9_bpb ~]$ sudo passwd jeromeg
  ```

  For the **`adduser`** command, the new password will be asked during the user creation process, so we do not need to set this unless we want to change it again.

- **Managing user accounts**

  To modify an existing user account, use the **`usermod`** command. In the example below, we will add the user **`jeromeg`** to the **`marketing`** group, enter the following command:

  ```
  [jsg@rhel9_bpb ~]$ sudo usermod -aG marketing jeromeg
  ```

  The **`-aG`** flag adds the user **`jeromeg`** to the **`marketing`** group and replaces the previous group membership of the user.

To remove a user account, use the **`userdel`** command. In the example below, we are deleting the user **`jblack`** in the console:

```
[jsg@rhel9_bpb ~]$ sudo userdel jblack
```

Be careful when using the **`userdel`** command for deleting user accounts — all files and directories owned by the user will also be deleted and the only way to recover them is if you have created a backup of their home directory!

- **Managing access permissions**

  Next, we must set the access permissions control which users and groups can access specific files and directories on the system. To modify access permissions for a file or directory, use the **`chmod`** command followed by the desired permission settings and the path to the file or directory. In the example below, we are giving the owner of the file **`myfile01`** read, write, and execute permissions:

  ```
  [jsg@rhel9_bpb ~]$ sudo chmod u+rwx myfile01
  ```

  The **`u+rwx`** flag gives the owner read, write, and execute permissions.

Managing user accounts and access permissions in RHEL 9 is a critical task for system administrators The commands described in this recipe follow best practices for securing user accounts and access permissions to maintain the security of your system.

Recipe #13: Configure date and time settings

To ensure the system clock in RHEL 9 is accurate, we need to learn how to configure the time and date settings. This ensures that the system clock is accurate and synchronized with other devices and servers on the network. RHEL 9 uses the **`timedatectl`** command to manage time and date settings. This command is part of the systemd system and service manager, which is responsible for controlling and managing the system's processes and services.

Now let us configure time and date settings in RHEL 9:

1. **Check the current time and date settings**

   Before we make any changes, let us check the current time and date settings. To do this, run the **`timedatectl`** command in the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo timedatectl
   ```

   The **`timedatectl`** command will display the current time, date, and time zone of your RHEL 9 system.

2. **Set the new time zone**

   If the time zone of the system is not set correctly, the time will be incorrect and may lead to issues in the applications you use. We will still use the **`timedatectl`**

```
[jsg@rhel9_bpb ~]$ sudo timedatectl list-timezones
```

The **`list-timezones`** option will provide a list of available time zones that can be set in your RHEL 9 system. Now let us set the time zone to New York:

```
[jsg@rhel9_bpb ~]$ sudo timedatectl set-timezone America/New_York
```

The RHEL 9 system will now be set to the **`America/New_York`** time zone.

3. **Enable automatic time synchronization**

   RHEL 9 can automatically synchronize the system clock with a remote network time server using the **Network Time Protocol** (**NTP**). To enable automatic time synchronization, run the following command in the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo timedatectl set-ntp yes
   ```

   The RHEL 9 system clock will now synchronize to the default NTP servers online and will reference the time zone set earlier.

4. **Manually set the time and date**

   If you do not want to use an NTP server to synchronize the system clock, you can manually set the time and date with the following:

   ```
   [jsg@rhel9_bpb ~]$ sudo timedatectl set-time "2023-04-23 12:30:00"
   ```

   The format you must use is YYYY-MM-DD HH:MM:SS which pertains to the year, month, day, hours, minutes and seconds you wish to set.

   After making any changes in your date and time, verify that the settings have been applied correctly. Run the **`timedatectl`** command again from the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo timedatectl
   ```

   The updated time, date, and time zone settings should be displayed.

Recipe #14: Configure network interfaces

Configuring and managing network interfaces is a common post-deployment task to ensure that your RHEL 9 system is properly set to communicate in your network. Network interfaces can be configured and managed using the NetworkManager service or by manually editing configuration files. Let us go through the steps on configuring and managing network interfaces in RHEL 9 using both methods:

1. **Configuring NetworkManager**

   The NetworkManager service is the default network management tool in RHEL 9. It provides a GUI as well as console options for configuring and managing network interfaces.

Configure using the graphical interface:

To configure network interfaces using the graphical interface, follow these steps:

1. Click on the Activities menu on the top left of the screen.
2. Type `Settings` in the search bar and click on the Settings icon.
3. Click on the Network icon.
4. Click on the + button to add a new connection.
5. Choose the type of connection you want to configure (choose between the options provided such as Ethernet, Wi-Fi, VPN, etc.).
6. Follow the on-screen prompts to configure the connection.

Configure using the console:

To configure network interfaces using console, follow these steps:

1. Open a console session.
2. Use the `nmcli` command as a `sudouser` to manage network interfaces.
3. Use the `nmcli connection add` command to add a new connection.
4. Use the appropriate options and arguments to configure the connection (e.g., `--type`, `--con-name`, `--autoconnect`, etc.).
5. Use the `nmcli connection modify` command to modify an existing connection.
6. Use the `nmcli connection delete` command to delete a connection.

2. **Manual configuration**

Network interfaces in RHEL 9 can also be configured manually by directly editing configuration files. The two main configuration files are `/etc/sysconfig/network-scripts/ifcfg-<interface_name>` and `/etc/resolv.conf`.

Configure the network interface with ifcfg-<interface_name>:

The `/etc/sysconfig/network-scripts/ifcfg-<interface_name>` file contains configuration information for a specific network interface. To configure a network interface manually, follow these steps:

1. Open the `/etc/sysconfig/network-scripts/ifcfg-<interface_name>` file with a text editor.
2. Edit the relevant configuration options (e.g., DEVICE, BOOTPROTO, IPADDR, NETMASK, GATEWAY, DNS1, DNS2, etc.).
3. Save the file and reload or reboot the system for the changes to take effect.

Configure the resolv.conf file:

The **`/etc/resolv.conf`** file contains DNS resolver configuration information. To configure DNS resolver settings manually, follow these steps:

1. Open **`/etc/resolv.conf`** file using a text editor.
2. Edit the nameserver lines to include the appropriate DNS server IP addresses.
3. Save the file and reload or reboot the system for the changes to take effect.

Recipe #15: Set up a firewall

It is good practice to set up a basic firewall in RHEL 9 to ensure traffic going in and out of your system is properly secured and only allowed ports can use the network. Let us go through the process of setting up a firewall in RHEL 9.

1. **Check the status of the firewall**

   Before setting up a firewall, let us check the status of the firewall. To do this, run the **`firewall-cmd`** command from the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo firewall-cmd --state
   ```

   If the firewall is active, you will see the running status. If the firewall is inactive, you will see the status not running. To start the firewall service run the **`systemctl`** command and indicate the **`firewalld`** service.

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl start firewalld
   ```

2. **Allow essential services**

   After checking the firewall status, the next step is to allow essential services with the **`firewall-cmd`** command. These services are required for the system to function correctly. Examples of essential services below include SSH, DNS, HTTP, and HTTPS.

   To add SSH:

   ```
   [jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=ssh
   ```

   To add DNS:

   ```
   [jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=dns
   ```

   To add HTTP:

   ```
   [jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=http
   ```

   To add HTTPS:

   ```
   [jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=https
   ```

The **`--permanent`** flag makes the rule applied persistent even after system reboots. To apply the rules immediately, run the same commands without the **`--permanent`** flag.

3. **Block unwanted traffic**

   The next step to configure in your firewall is to block unwanted traffic. This includes blocking incoming traffic from unknown sources, blocking specific IP addresses, etc. To block incoming traffic from unknown sources, do the following command from the console:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-rich-
rule='rule family="ipv4" source address="0.0.0.0/0" reject'
```

   To block incoming traffic from a specific IP address, do the following command from the console (change the IP address as needed):

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-rich-
rule='rule family="ipv4" source address="192.168.1.100" reject'
```

4. **Reload the firewall configuration**

   After making changes to the firewall configuration, reload the configuration to apply the changes. To reload the firewall configuration, run **`firewall-cmd`** with the **`--reload`** flag:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

5. **Check firewall configuration**

   To check the firewall configuration, and display all active firewall rules including the allowed services and blocked traffic, use the **`--list-all`** flag:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --list-all
```

Remember to reload the firewall configuration after making changes and regularly review the firewall rules to ensure they remain updated, and your system secured.

Recipe #16: Enable and disable system services

System services are background processes that run continuously on a RHEL 9 system to support various functions and applications. RHEL 9 runs several essential system services by default that allows the system to function properly. However, there are times that some services loaded by default may not be considered essential in your day-to-day operations and may need to be disabled or unloaded. Let us go through some basics in managing system services in RHEL 9:

1. **List the system services installed in RHEL 9**

   The **systemctl** command allows you to interact with the **systemd** system and service manager. The command allows you to manage the RHEL 9 system and perform tasks needed to start, stop, restart, loading, and unloading system services.

   The first thing you need to do is check the system services that are loaded and active in RHEL 9. Open a console and list the loaded services:

```
[jsg@rhel9_bpb ~]$ sudo systemctl list-units --type service
```

   You can also list all system services loaded in RHEL 9 regardless of state if they are active or disabled:

```
[jsg@rhel9_bpb ~]$ sudo systemctl list-unit-files --type service
```

2. **Enable a system service in RHEL 9**

   The same **systemctl** command is used in enabling a system service in RHEL 9. Once you have listed the system services and identified what to enable, do the following from the console to enable the service:

```
[jsg@rhel9_bpb ~]$ sudo systemctl enable <name>.service
```

   Replace **<name>** with the name of the service you want to enable. Once executed, the service will be enabled and will automatically start every time you reboot the system.

   It is important to note that enabling a service does not start it immediately. If the service is currently running, you will need to start it manually using the **systemctl start** command:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start <name>.service
```

3. **Disable a system service in RHEL 9**

   Disabling a system service in RHEL 9 is also done using the **systemctl** command:

```
[jsg@rhel9_bpb ~]$ sudo systemctl disable <name>.service
```

   Like before, replace **< name>** with the name of the service you want to disable. When completed, the service will be disabled and will not automatically start after every boot or rebooting of your RHEL 9 system.

   Remember, just like when enabling a service where it does not start automatically, disabling a service does not stop it immediately. If the service is currently running, you will need to stop it manually using the **systemctl stop** command:

```
[jsg@rhel9_bpb ~]$ sudo systemctl stop <name>.service
```

Conclusion

In this chapter, we have covered essential post-deployment configuration tasks in RHEL 9 that would ensure the security and integrity of your system. By applying the best practices described here, your RHEL 9 system would now be ready to run critical applications. Make sure to regularly review the configurations and parameters set and set a regular schedule for the exercise to keep the rule sets fresh and updated depending on your need.

In the next chapter, we will go through managing your RHEL 9 subscriptions to ensure you are properly covered by official Red Hat support.

Points to remember

- You need RHEL 8 to do an in-place upgrade to RHEL 9. For installations running RHEL 7 and below, there is no direct upgrade path to RHEL 9. To upgrade, you must do an upgrade to RHEL 8, prior to doing another upgrade to RHEL 9.
- When setting up the firewall, allow only services that you need to use. For example, if you do not need to run a web server, do not enable the HTTP and HTTPS services through the firewall.
- The **`systemctl`** command is quite powerful and makes it easy to enable and disable services on your RHEL 9 system. Always ensure to carefully select which services to enable or disable to avoid any potential issues with your system's overall functionality.

CHAPTER 5

Managing RHEL 9 Subscriptions

Introduction

RHEL 9 is a Linux distribution targeted for the enterprise and Red Hat provides holistic support for enterprise installations with an assigned paid subscription. In this chapter, we will go through the subscription options (both free and paid) for RHEL 9, navigate the Red Hat Customer Portal, learn how to raise support tickets, and gather important system reports from your RHEL 9 system. We will also be discussing the Red Hat Developer Program, that would allow you to run RHEL 9 systems at no-cost for testing and evaluation.

Structure

In this chapter, we will discuss the following topics:

- Obtaining a no-cost developer subscription of RHEL with the Red Hat Developer Program
- Registering a newly installed RHEL 9 setup to receive software updates
- Installing the sos utility to generate a system report and using it to get technical support from Red Hat

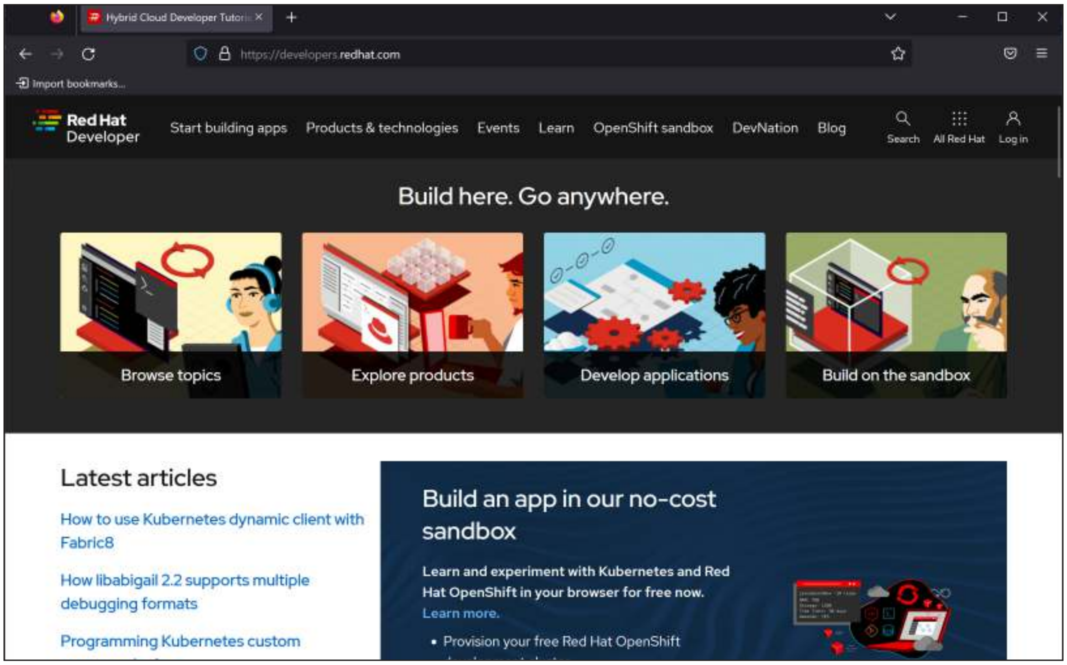
Objectives

In this chapter, our learning objectives are the following: a) understand the Red Hat Developer Program to be able to take advantage of no-cost developer subscriptions of Red Hat products for personal learning and evaluation, and b) use included tools in RHEL in creating system reports that would be useful when obtaining technical support from Red Hat.

The Red Hat Developer Program is an indispensable resource for sysadmins as it allows you to evaluate and test Red Hat products and not just limited to RHEL 9. So, use this to your advantage as you progress you are learning with your RHEL 9 system!

Recipe #17: Get a no-cost RHEL for developers subscription for testing

While RHEL 9 can be freely downloaded from the Red Hat website and released as open source, an enterprise subscription agreement is still necessary for commercial use to activate the system and get software updates and security patches. However, to promote the awareness and development use of RHEL, Red Hat introduced a no-cost RHEL for developers' subscription as part of the Red Hat Developer program intended for individual developers. If you are a sysadmin that is currently upskilling on your own personal time, this subscription is a welcome addition to your learning tools.



Build here. Go anywhere.

Latest articles

How to use Kubernetes dynamic client with Fabric8

How libabigail 2.2 supports multiple debugging formats

Programming Kubernetes custom

Build an app in our no-cost sandbox

Learn and experiment with Kubernetes and Red Hat OpenShift in your browser for free now. Learn more.

• Provision your free Red Hat OpenShift

Figure 5.1

A no-cost RHEL for developers subscription for individual developers provide the following:

- All supported versions of RHEL
- Red Hat Developer tools and compilers
- Red Hat software collection and application streams
- RHEL infrastructure add-ons such as resilient storage, networking, and scalable file systems
- Access to the Red Hat Customer Portal for software updates and knowledge base articles

The no-cost RHEL provided is the same software used in commercial and enterprise deployments. The key difference is that the no-cost RHEL developer subscription is self-supported and access to the Red Hat Customer Portal provides resources needed for self-support such as deployment, configuration, security, and management articles related to RHEL 9.

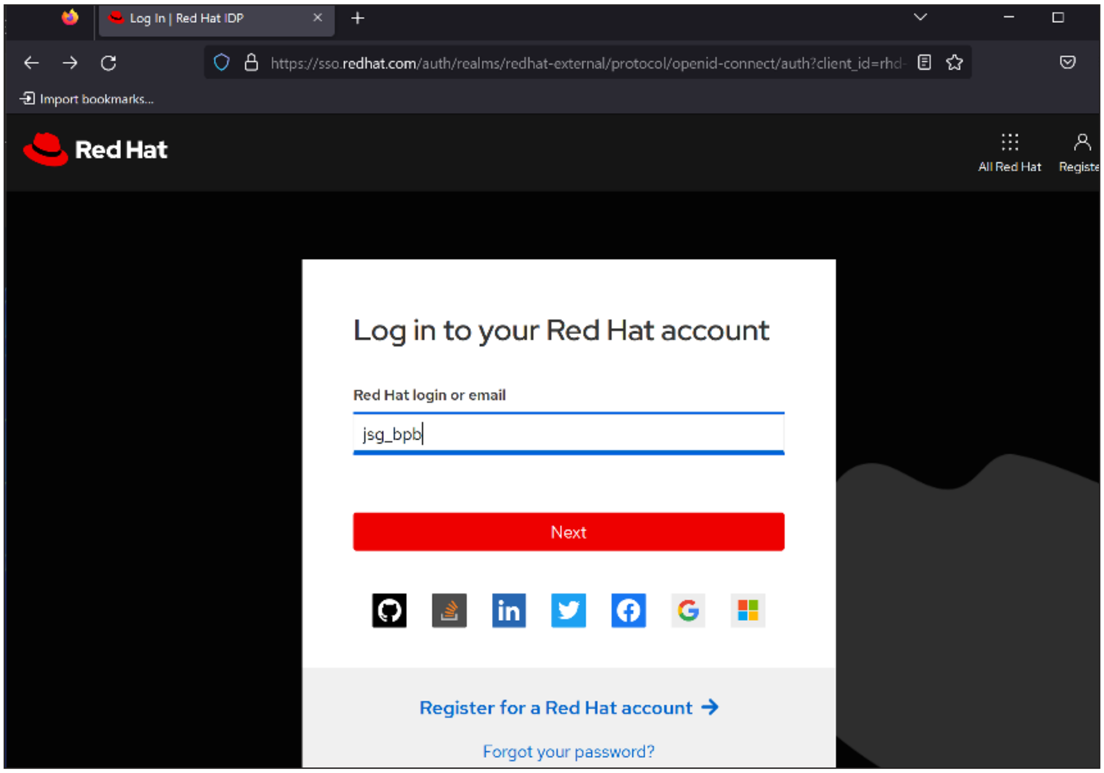
The no-cost RHEL subscription gives you the following entitlements:

- Register up to 16 physical or virtual nodes of RHEL under your individual Red Hat Developer Program account (registering activates system updates for your RHEL 9 setup).
- Access to software and security updates.
- Self-service support through the Red Hat Customer Portal.

Also note that no SLA is provided for RHEL 9 installations that are registered through the no-cost RHEL subscription program. On a positive note, access to the Red Hat Developer Program gives you access to all of Red Hat's product portfolio including Red Hat OpenShift Container Platform and Red Hat Ansible Automation Platform, so you are actually getting a lot of value on the resources and articles you can learn at your own pace.

Now let us get started in getting a Red Hat Developer account created so that you are able to download RHEL 9 at no-cost for individual development and testing.

1. Register for a new account at **https://developers.redhat.com/.** Click on the login icon and select **Register for a Red Hat account**. This new Red Hat account will be used primarily for the individual Red Hat Developer Program.


Log In | Red Hat IDP
https://sso.redhat.com/auth/realms/redhat-external/protocol/openid-connect/auth?client_id=rhd
Import bookmarks...
Red Hat
All Red Hat
Registe

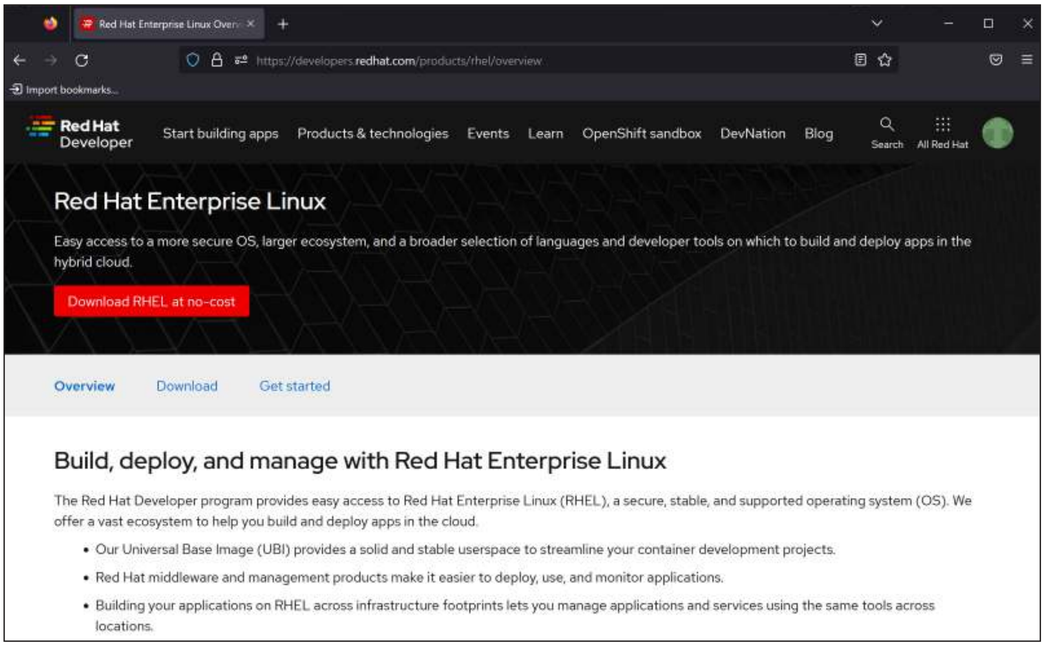
Log in to your Red Hat account

Red Hat login or email: jsg_bpb

Next
Register for a Red Hat account
Forgot your password?

Figure 5.2: Register for a new Red Hat Developer program account

2. Once your Red Hat account is created, login to the Developer Program page and click **Products & technologies** and select **Red Hat Enterprise Linux**.
3. Click the red button that says **Download RHEL at no-cost** and the latest version of the RHEL x86_64 DVD iso will automatically download.


Red Hat Enterprise Linux Overv ×

https://developers.redhat.com/products/rhel/overview

Import bookmarks...

Red Hat Enterprise Linux

Easy access to a more secure OS, larger ecosystem, and a broader selection of languages and developer tools on which to build and deploy apps in the hybrid cloud.

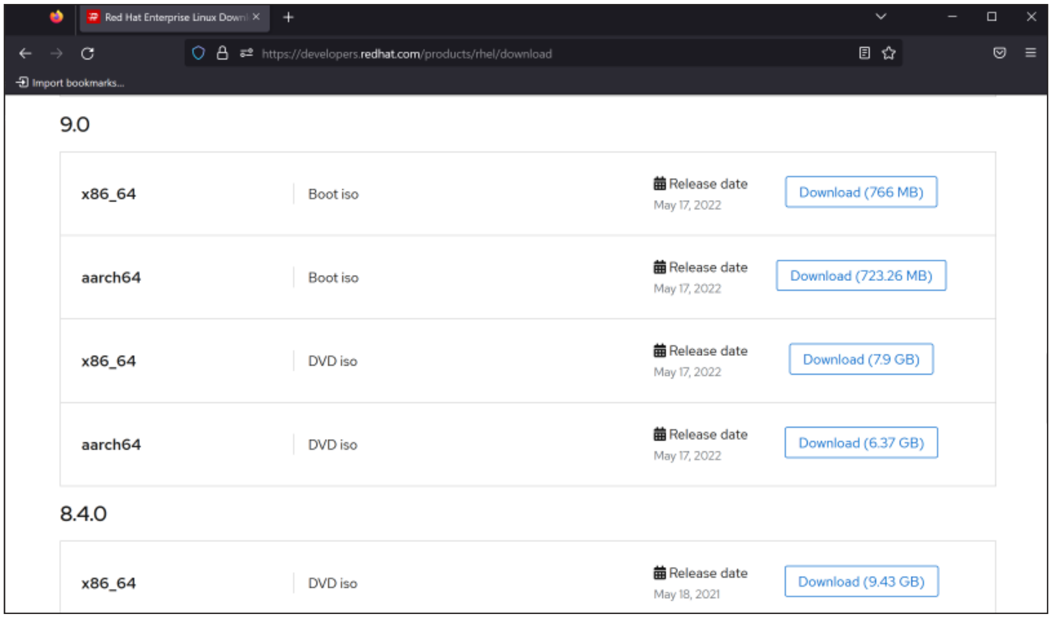
Build, deploy, and manage with Red Hat Enterprise Linux

The Red Hat Developer program provides easy access to Red Hat Enterprise Linux (RHEL), a secure, stable, and supported operating system (OS). We offer a vast ecosystem to help you build and deploy apps in the cloud.

- Our Universal Base Image (UBI) provides a solid and stable userspace to streamline your container development projects.
- Red Hat middleware and management products make it easier to deploy, use, and monitor applications.
- Building your applications on RHEL across infrastructure footprints lets you manage applications and services using the same tools across locations.

***Figure 5.3**: Red Hat Enterprise Linux at Red Hat Developer portal*

4. If you wish to select specific versions and architectures of RHEL click **Download** and select the version to download. Note that only current and supported versions of RHEL are provided (including RHEL 9).


Red Hat Enterprise Linux Downl ×

https://developers.redhat.com/products/rhel/download

Import bookmarks...

9.0

x86_64	Boot iso	Release date May 17, 2022	Download (766 MB)
aarch64	Boot iso	Release date May 17, 2022	Download (723.26 MB)
x86_64	DVD iso	Release date May 17, 2022	Download (7.9 GB)
aarch64	DVD iso	Release date May 17, 2022	Download (6.37 GB)

8.4.0

x86_64	DVD iso	Release date May 18, 2021	Download (9.43 GB)

Figure 5.4

If you are part of a team in a company or organization that seeks to use RHEL 9 for assessment and testing, Red Hat provides a different program called Red Hat Developer Subscription for Teams. The software access and entitlements are similar to the individual program but with added benefits as a corporate user. Red Hat provides a comprehensive FAQ on this program, and you need to get in touch with a local Red Hat representative to get your team started. You can go through the detailed FAQ for more information about the program and the benefits at these links:

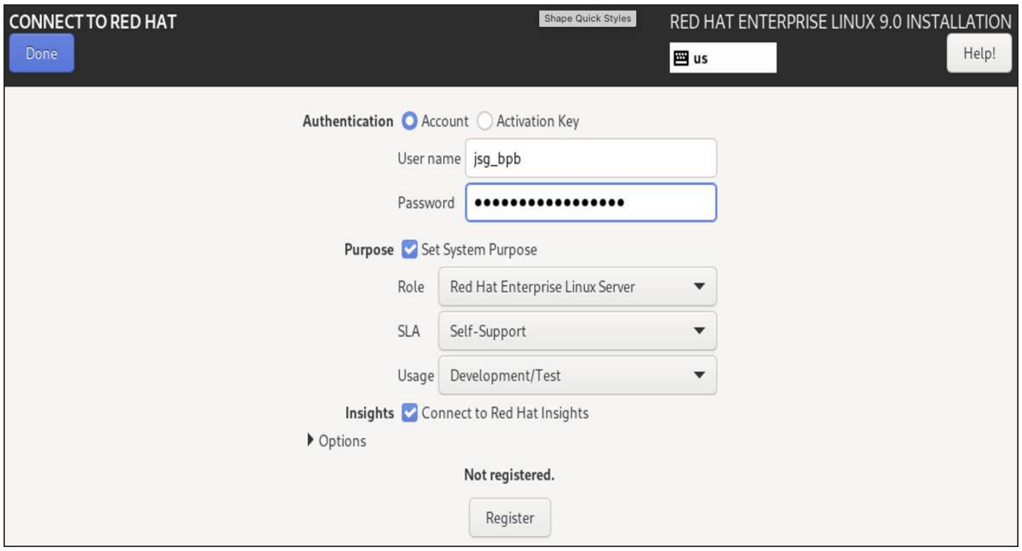
- **https://developers.redhat.com/articles/2022/05/10/access-rhel-developer-teams-subscription**
- **https://developers.redhat.com/articles/2022/07/06/what-qualifies-red-hat-developer-subscription-teams**

Recipe #18: Register your RHEL 9 system to receive software updates

As discussed in Recipe #17, you need to register your RHEL 9 system to Red Hat to receive software and security updates. Registering your RHEL 9 system also lets you connect (as an option) to Red Hat Insights, a managed cloud service by Red Hat that analyzes your RHEL 9 system based on the configuration, installed applications, and provides prescriptive and pro-active information in securing and using your system to its fullest. Red Hat Insights is included in every RHEL subscription (including the no-cost RHEL program).

You can choose between 3 methods to register your RHEL 9 system to Red Hat and you have to option to do it during setup or after completing the installation. Let us go through each option:

1. **Registering through Anaconda**: During the installation setup from Anaconda, select Connect to Red Hat. Use your Red Hat account credentials to register your RHEL 9 system. You can also put information on the purpose of your RHEL 9 system by marking **Set System Purpose** and selecting the following:
    - **Role**: The role of the RHEL 9 system you are registering.
    - **SLA**: Depending on your subscription, it can be Premium (24x7), Standard (8x5), or Self-Support.
    - **Usage**: You can tag your RHEL 9 system as Production, Compute Node, or for Development/Test.



CONNECT TO RED HAT

RED HAT ENTERPRISE LINUX 9.0 INSTALLATION

Authentication [✓ Account] Activation Key

User name jsg_bpb

Password ●●●●●●●●●●●●●●●●●

Purpose [✓] Set System Purpose

Role Red Hat Enterprise Linux Server

SLA Self-Support

Usage Development/Test

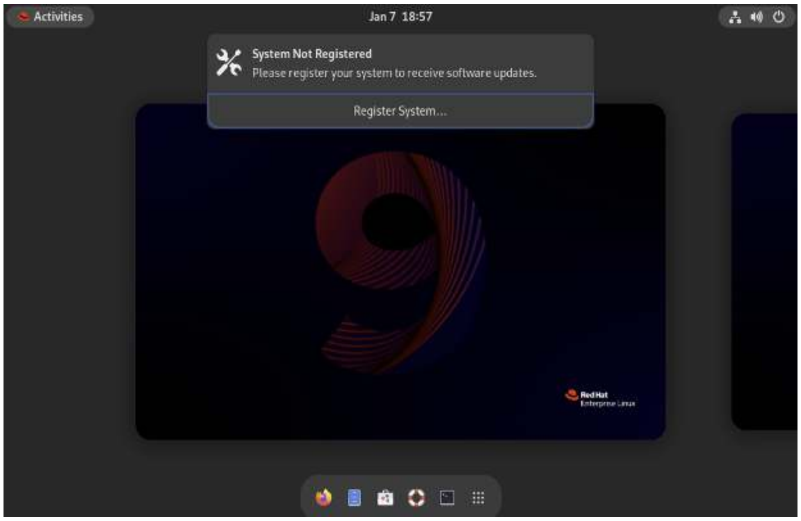
Insights [✓] Connect to Red Hat Insights

Not registered.

***Figure 5.5**: Registering a RHEL 9 system during the installation setup*

If you are installing RHEL 9 on behalf of your organization and have an active subscription from Red Hat, you will be provided an Activation Key by your Red Hat representative to use in registering your RHEL 9 system. You can use this Activation Key in order to register RHEL 9 under your organization.

2. **Registering after setup in the GNU Network Object Model Environment (GNOME)**: After the installation and creation of your local user account, the initial login screen will show that the RHEL 9 system needs to be registered to receive software and security updates as shown in *Figure 5.6:*



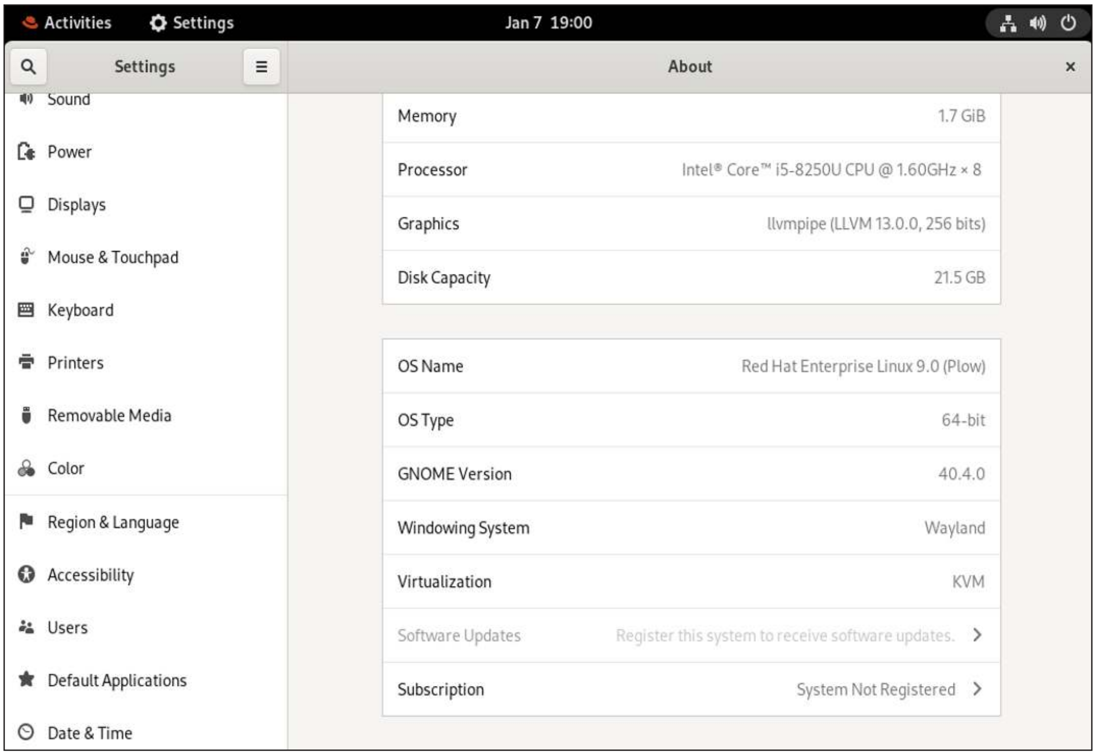
System Not Registered

Please register your system to receive software updates.

Figure 5.6

From GNOME, there are two ways to register this RHEL 9 system with Red Hat using your Red Hat account or Activation Key provided by Red Hat.

- From GNOME Settings, select **About**, and scroll down and you will see that the RHEL 9 system has yet to be registered. Click on **Subscription**, and input your Red Hat account or Activation Key to register this system as shown in *Figure 5.7*:

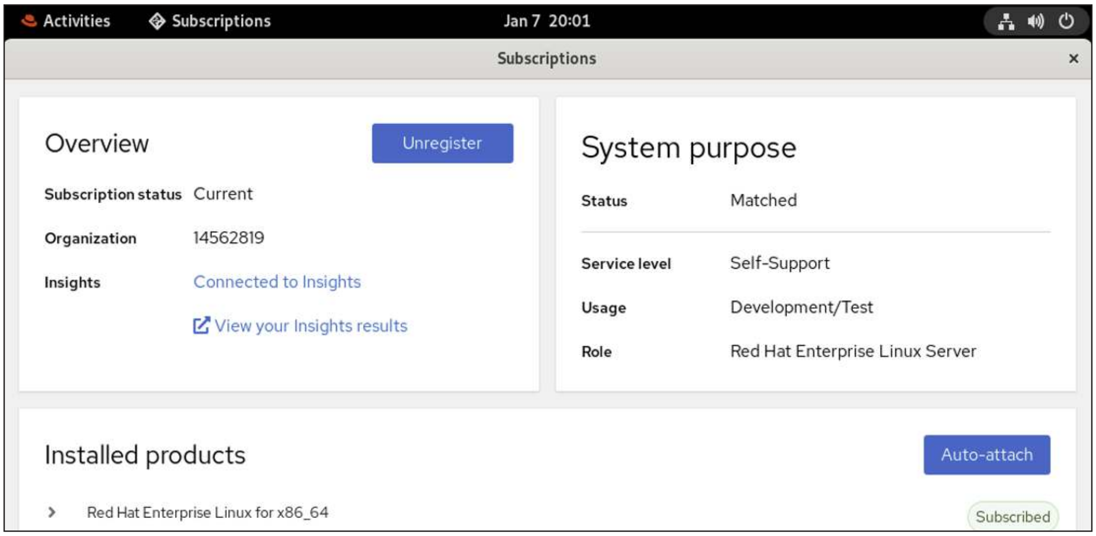


Memory	1.7 GiB
Processor	Intel® Core™ i5-8250U CPU @ 1.60GHz × 8
Graphics	llvmpipe (LLVM 13.0.0, 256 bits)
Disk Capacity	21.5 GB

OS Name	Red Hat Enterprise Linux 9.0 (Plow)
OS Type	64-bit
GNOME Version	40.4.0
Windowing System	Wayland
Virtualization	KVM
Software Updates	Register this system to receive software updates.
Subscription	System Not Registered

***Figure 5.7**: Registering the RHEL 9 system through GNOME Settings.*

- From GNOME, you can also register and activate your RHEL 9 system using the Red Hat Subscription Manager application. Use your Red Hat account or Activation Key to register the RHEL 9 system and once validated, the subscription manager will show that the setup is already registered as shown in *Figure 5.8*. Finally, click on the **Auto-attach** button to attach this RHEL 9 setup to the Red Hat account associated. This will ensure this RHEL 9 system can receive system and security updates when made available by Red Hat.



Overview

Subscription status	Current
Organization	14562819
Insights	Connected to Insights
	View your Insights results

System purpose

Status	Matched
Service level	Self-Support
Usage	Development/Test
Role	Red Hat Enterprise Linux Server

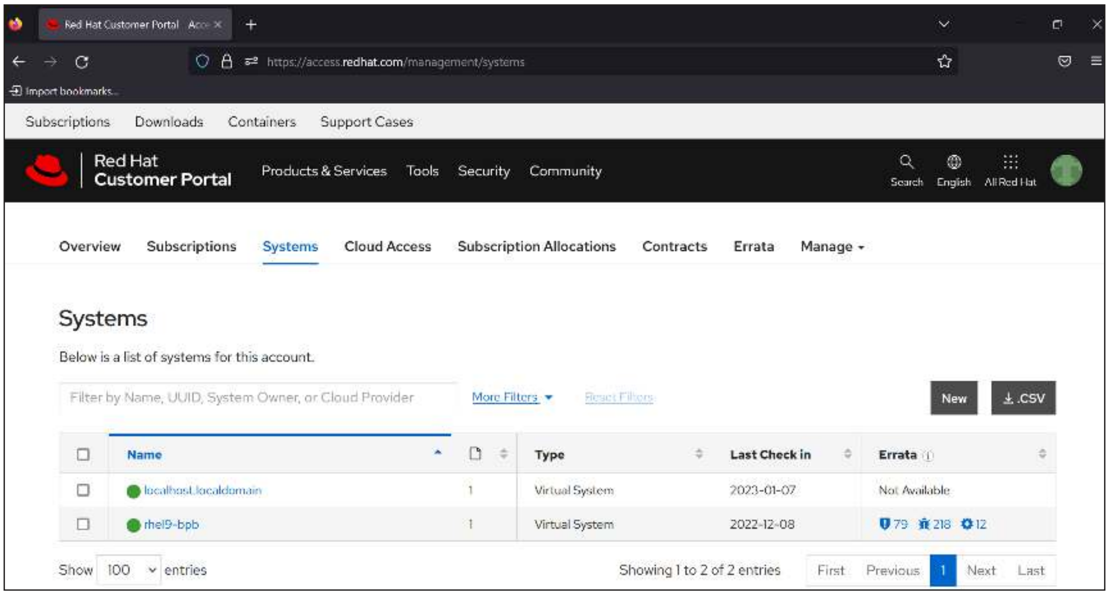
Installed products

Red Hat Enterprise Linux for x86_64	Subscribed

***Figure 5.8**: Information of a registered RHEL 9 system using Red Hat Subscription Manager application in GNOME*

Once your RHEL 9 system is registered and the subscription activated, the last exercise for this recipe is to validate the activation and check the entitlements utilized based on your account. From the Red Hat Developer portal (you can also go to the Red Hat Customer Portal for this), click on your Account icon at the top right side of the page and select **Subscriptions**. You will be presented with a summary of subscriptions usage as well as relevant errata information on your activated RHEL 9 systems such as active security advisories, bug fixes, and product enhancements.

Click **Systems** to show a list of activated systems under your Red Hat account and attached subscription as shown in *Figure 5.9*. From here, you can select the system to check if its properly subscribed to receive updates from Red Hat through the duration of the subscription.



Systems

Below is a list of systems for this account.

Name		Type	Last Check in	Errata
localhost.localdomain	1	Virtual System	2023-01-07	Not Available
rhel9-bpb	1	Virtual System	2022-12-08	79 218 12

Show 100 entries Showing 1 to 2 of 2 entries First Previous 1 Next Last

***Figure 5.9**: Red Hat Customer Portal showing Systems attached to a Red Hat account and subscription*

Recipe #19: Install the sos utility and generate a system report

The first thing to do prior to getting official technical support from Red Hat is to generate a report that contains system and diagnostic information of your RHEL 9 installation and setup. This report is generated by the sos utility and is called a **sos report**. The sos report is the initial data used by Red Hat technical support engineers for analysis of a RHEL 9 setup. The data collected from your RHEL 9 system would include the following:

- The kernel version installed
- System and services configuration
- Installed packages
- The command output from sos itself

To generate the sos report, first you will need to install the sos utility tool.

Open the command line and switch to root user. As root, install the **`sos`** package with the **`dnf`** command:

```
[jsg@rhel9-bpb ~]$ sudo -i
[sudo] password for jsg:
[root@rhel9-bpb ~]# dnf install sos
```

To verify if **`sos`** is installed, do an **`rpm`** query command:

```
[root@rhel9-bpb ~]# rpm -q sos
sos-4.3-5.el9_1.noarch
```

Once the sos utility tool is installed, run **`sos report`** from the command line as a **`sudo`** user:

```
[jsg@rhel9-bpb ~]$ sudo sos report

sosreport (version 4.3)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.lpv3r96z and may be provided to a Red Hat support
representative.

Any information provided to Red Hat will be treated in accordance with
the published support policies at:

 Distribution Website : https://www.redhat.com/
 Commercial Support   : https://www.access.redhat.com/
The generated archive may contain data considered sensitive and its
content should be reviewed by the originating organization before being
passed to any third party.

No changes will be made to system configuration.

Press ENTER to continue, or CTRL-C to quit.
```

Once the sos report is completed, it will be available at the **`/var/tmp`** folder and ready for sending to Red Hat technical support. The output would be like the one below:

```
Your sosreport has been generated and saved in:
  /var/tmp/sosreport-rhel9-bpb-2023-01-06-qppkokn.tar.xz

Size 11.65MiB
Owner root
sha256 3c1cbe975a5269acbb29aa21333c162ea9347f2be7001c7ac51110ab56619449
```

```

Please send this file to your support representative.
```

Recipe #20: Clean the sos report to remove sensitive or private data

If you wish to not include data that is deemed private by your organization, you can use the sos utility to obfuscate data that is potentially sensitive. You need to run the **`sos report`** command first to create the report and start cleaning up the report output using the **`sos clean`** command.

1. Open the command line and run the **`sos clean`** command as a **`sudo`** user and include the location of the sos report generated that you would like to be cleaned:

```
[jsg@rhel9-bpb ~]$ sudo sos clean /var/tmp/sosreport-rhel9-bpb-2023-01-06-qppkokn.tar.xz
[sudo] password for jsg:

sos clean (version 4.3)

This command will attempt to obfuscate information that is generally considered
to be potentially sensitive. Such information includes IP addresses, MAC
addresses, domain names, and any user-provided keywords.

Note that this utility provides a best-effort approach to data obfuscation, but
it does not guarantee that such obfuscation provides complete coverage of all
such data in the archive, or that any obfuscation is provided to data that does
not fit the description above.

Users should review any resulting data and/or archives generated or processed by
this utility for remaining sensitive content before being passed to a third
party.


Press ENTER to continue, or CTRL-C to quit.
```

```

Found 1 total reports to obfuscate,
processing up to 4 concurrently

sosreport-rhel9-bpb-2023-01-06-qppkokn :
Extracting...
sosreport-rhel9-bpb-2023-01-06-qppkokn :
Beginning obfuscation...
sosreport-rhel9-bpb-2023-01-06-qppkokn :
Re-compressing...
sosreport-rhel9-bpb-2023-01-06-qppkokn :
Obfuscation completed [removed 879 unprocessable files]

Successfully obfuscated 1 report(s)

A mapping of obfuscated elements is available at
/var/tmp/sosreport-host0-2023-01-06-qppkokn-private_map

The obfuscated archive is available at
/var/tmp/sosreport-host0-2023-01-06-qppkokn-obfuscated.tar.xz

Size 5.32MiB
Owner root

Please send the obfuscated archive to your support
representative and keep the mapping file private
```

2. The **`sos clean`** command will create two new files inside **`/var/tmp`** along with the original sos report that was previously generated. First, a **`*-obfuscated.tar.xz`** file which is the cleaned version of the original report, an a **`*-private_map`** file which shows what data was obfuscated during the **`sos clean`** command execution:

```
[jsg@rhel9-bpb ~]$ sudo ls -l /var/tmp/sosreport-host0-2023-
01-06-qppkokn-private_map /var/tmp/sosreport-host0-2023-01-06-
qppkokn-obfuscated.tar.xz
-rw-------. 1 root root 5578196 Jan  8 14:28 /var/tmp/
sosreport-host0-2023-01-06-qppkokn-obfuscated.tar.xz
-rw-------. 1 root root    3639 Jan  8 14:28 /var/tmp/
sosreport-host0-2023-01-06-qppkokn-private_map
```

3. To show the mapping of obfuscated elements you can do a cat of the **`*-private_map`** file:

```
[jsg@rhel9-bpb ~]$ sudo cat /
```

```
06-qppkokn-private_map
[sudo] password for jsg:
{
    "hostname_map": {
        "rhel9-bpb": "host0"
    },
    "ip_map": {
        "10.0.2.0/24": "100.0.0.0/24",
        "10.0.2.15/24": "100.0.0.1/24",
        "10.0.2.255": "100.0.0.255",
        "192.168.0.0/16": "101.0.0.1/16",
        "192.168.0.1": "101.0.0.2/16",
        "10.1.2.3": "47.13.37.22",
        "192.168.1.1": "101.0.0.3/16",
        "192.168.2.0/24": "102.0.0.1/24",
        "192.168.3.200": "101.0.0.4/16",
        "192.68.3.100": "27.47.97.70",
        "192.168.0.150": "101.0.0.5/16",
        "192.168.0.50": "101.0.0.6/16",
        "192.168.0.60": "101.0.0.7/16",
        "192.168.0.70": "101.0.0.8/16",
        "192.168.0.61": "101.0.0.9/16",
        "192.168.0.4": "101.0.0.10/16",
        "10.10.0.5": "81.62.35.46",
        "192.168.1.0/24": "103.0.0.1/24",
        "10.0.0.0/8": "104.0.0.1/8",
        "5.6.7.8": "51.18.97.86",
        "192.168.1.100": "103.0.0.2/24",
        "192.168.0.3": "101.0.0.11/16",
        "64.94.110.11": "43.30.96.11",
        "5.6.7.0": "42.73.14.54",
        "192.168.0.10": "101.0.0.12/16",
        "192.168.0.40": "101.0.0.13/16",
        "10.0.0.10": "104.0.0.2/8",
        "10.0.0.40": "104.0.0.3/8",
        "10.0.2.3": "100.0.0.2/24",
        "192.168.1.12": "103.0.0.3/24",
        "192.168.1.0/25": "105.0.0.1/25",
        "192.168.1.13": "105.0.0.2/25",
        "192.168.3.0/24": "106.0.0.1/24",
        "192.168.3.0/25": "107.0.0.1/25",
        "192.168.0.133": "101.0.0.14/16",
        "192.168.100.50": "101.0.0.15/16",
        "192.168.122.0/24": "108.0.0.1/24",
        "192.168.200.1": "101.0.0.16/16",
        "192.168.200.4": "101.0.0.17/16",
        "192.168.200.9": "101.0.0.18/16",
        "192.168.201.0/24": "109.0.0.1/24",
        "192.168.201.0/25": "110.0.0.1/25",
        "34.25.0.29": "37.32.76.81",
        "10.0.2.0/28": "111.0.0.1/28",
        "209.132.178.16": "58.40.52.49",
        "162.159.200.1": "60.40.42.52",
        "162.159.200.123": "36.78.47.44",
        "224.0.0.251": "26.21.97.40",
        "23.46.103.16": "92.38.65.96",
        "4.16.1.3": "83.16.37.11",
        "224.0.0.1": "62.48.31.44",
        "224.0.0.22": "31.56.71.18",
        "10.88.0.0/16": "112.0.0.1/16",
        "10.88.0.1": "112.0.0.2/16",
        "10.89.0.0/16": "113.0.0.1/16",
        "10.90.0.0/15": "114.0.0.1/15",
        "10.92.0.0/14": "115.0.0.1/14",
        "10.96.0.0/11": "116.0.0.1/11",
        "10.128.0.0/9": "117.0.0.1/9",
        "10.0.2.100": "100.0.0.3/24",
        "10.10.1.136": "104.0.0.4/8",
        "18.168.6.1": "22.98.89.24",
        "25.30.13.0": "70.60.19.65",
        "2.7.10.1": "47.87.54.76",
        "10.0.2.15/32": "111.0.0.15",
        "39.31.5.1": "99.99.16.31"
    },
    "mac_map": {
        "52:54:00:ca:01:98": "53:4f:53:77:28:7c",
        "11:22:33:44:55:66": "53:4f:53:51:48:b2",
        "12:34:56:78:90:12": "53:4f:53:51:b9:18",
        "e4:45:74:68:00:00": "53:4f:53:64:e5:e3",
        "52:55:0a:00:02:02": "53:4f:53:f1:e9:cd",
        "52:55:0a:00:02:03": "53:4f:53:b1:b4:93",
        "01:00:5e:00:00:01": "53:4f:53:52:76:75",
        "33:33:00:00:00:01": "53:4f:53:4f:a1:1e",
        "33:33:ff:ca:01:98": "53:4f:53:48:ac:ae",
        "01:00:5e:00:00:fb": "53:4f:53:02:d0:cf",
        "33:33:00:00:00:fb": "53:4f:53:b7:be:d3",
        "52:56:00:00:00:02": "53:4f:53:c9:ed:07",
        "01:00:5e:00:00:16": "53:4f:53:07:f0:06",
        "33:33:00:00:00:02": "53:4f:53:d7:44:c6",
        "33:33:00:00:00:16": "53:4f:53:e1:53:cf",
        "1af4:1050:1af4:1100": "534f:53ff:fe57:70c8",
        "08:05:06:19:40:00": "53:4f:53:63:6e:82"
    },
    "keyword_map": {},
    "username_map": {
        "jsg": "obfuscateduser0"
```

Recipe #21: Sending the sos report to Red Hat Technical support with various tools

After the sos report is generated. You can now send the **`*.tar.xz`** to Red Hat for technical support, provided you have a commercial RHEL subscription agreement with Red Hat (applies to both Premium and Standard subscriptions. Self-support is not entitled for this).

1. Use the **`sos report --upload`** command to transfer the sos report to Red Hat. Additionally, if you create a support case in the Red Hat Customer Portal, you will be assigned a Case ID which is 8 digits long. You will be prompted as well for your Red Hat Customer Portal username to associate the sos report under your account.

```
[jsg@rhel9-bpb ~]$ sudo sos report --upload
[sudo] password for jsg:

sosreport (version 4.3)

This command will collect diagnostic and configuration information from
this Red Hat Enterprise Linux system and installed applications.

An archive containing the collected information will be generated in
/var/tmp/sos.4gw3dmes and may be provided to a Red Hat support
```

```
representative.

Any information provided to Red Hat
will be treated in accordance with
the published support policies at:

        Distribution Website : https://www.redhat.com/
        Commercial Support   : https://www.access.redhat.com/

The generated archive may contain data
considered sensitive and its
content should be reviewed by the originating
organization before being
passed to any third party.

No changes will be made to system configuration.

Press ENTER to continue, or CTRL-C to quit.

Optionally, please enter the case id that
you are generating this report for []:

Setting up archive ...
Setting up plugins ...

[plugin:networking] skipped command 'ip -s macsec show':
required kmods missing: macsec.   Use '--allow-system-changes'
to enable collection.
[plugin:networking] skipped command 'ss -peaonmi':
required kmods missing: xsk_diag.   Use
'--allow-system-changes' to enable collection.
[plugin:sssd] skipped command 'sssctl config-check':
required services missing: sssd.
[plugin:sssd] skipped command 'sssctl domain-list':
required services missing: sssd.
[plugin:systemd] skipped command 'systemd-resolve --status':
required services missing: systemd-resolved.
[plugin:systemd] skipped command 'systemd-resolve
--statistics': required services missing: systemd-resolved.
```

```
[plugin:wireless] skipped command 'iw list': required kmods missing: cfg80211.
[plugin:wireless] skipped command 'iw dev': required kmods missing: cfg80211.
[plugin:wireless] skipped command 'iwconfig': required kmods missing: cfg80211.
[plugin:wireless] skipped command 'iwlist scanning': required kmods missing: cfg80211.
Running plugins. Please wait ...

  Finishing plugins             [Running: subscription_manager]
                                  ]
  Finished running plugins
Creating compressed archive...

Your sosreport has been generated and saved in:
/var/tmp/sosreport-rhel9-bpb-2023-01-08-qaswruh.tar.xz

Size 11.65MiB
Owner root
sha256 4584b3b6abb80f2bfc8c2766e5292988dd744828846442f475b8f8efca0c090c

Please send this file to your support representative.

Attempting upload to Red Hat Secure FTP
User 'xiYKqGKw'used for anonymous upload. Please inform your support engineer so they may retrieve the data.
Uploaded archive successfully
```

If you do not include (or do not have it on-hand) a Case ID or your Red Hat Customer Portal username, the sos report will be uploaded to the public FTP site of Red Hat and anonymized. You will need to provide to Red Hat Technical Support the exact file name of the sos report that was uploaded.

2. The other option is to manually upload the generated sos report file to the Red Hat Customer Portal and attach it when you open a support case. This option is useful if the RHEL 9 server you generated the sos report with has no direct outbound connection to the internet. To open a Support Case, head to **https://access.redhat.com/support/cases/** and login with your Red Hat account. Click on the Open a case button at the upper right side of the support portal to start your technical support request.

Conclusion

To start managing your RHEL 9 subscriptions and entitlements, you first need to create a Red Hat account. Doing so makes it easier for you to manage your Red Hat subscriptions and entitlements which cover both enterprise and individual/developer accounts. A Red Hat account also gives you access to the Red Hat Developer Portal that provides a no-cost RHEL developer subscription for assessment, testing, and learning. The Red Hat account is also needed for creating support requests in the Red Hat Customer Portal. If you are managing a RHEL 9 environment with a commercial subscription from Red Hat, you can use the sos utility tool to generate a system report that you can send to Red Hat as part of the technical support resolution process and this report will allow Red Hat support engineers to diagnose your system based on the generated report.

In the next chapter, we will go through the steps and details in configuring additional software repositories in RHEL 9 so you can expand the software catalog based on your needs.

Points to remember

- Create a Red Hat account to manage your RHEL 9 subscriptions and entitlements using the Red Hat Customer Portal. This can be used for both developer (no-cost) and enterprise accounts.
- For individual testing and evaluation, take advantage of the no-cost RHEL for developers subscription as it gives enough entitlements for your assessment and learning goals.
- To ensure that data is anonymized prior to getting technical support from Red Hat (for commercial subscriptions), always use the clean option and send the obfuscated **`*.obfuscated.tar.xz`** file to Red Hat or upload using the sos utility tool.

Chapter 6
Configuring Software Repositories and RHEL 9 Updates

Introduction

Red Hat provides official support to the software packages it provides out of the box as well as those downloaded and installed from their official software repositories. As a SysAd, you will encounter cases where some packages are not officially supported by Red Hat, but you will still need to install and configure it into your system. To do so, you will need to configure a new software repository that would allow you to install and manage a software package that is not provided, nor officially supported by Red Hat.

In this chapter, we will learn how to manage software packages with the **`dnf`** tool, configure new software repositories so we can install additional software packages that are not provided by Red Hat, update our RHEL 9 system, and configure **`dnf`** to automatically download and update RHEL 9 when desired.

Structure

In this chapter, we will discuss the following topics:

- Using the dnf command for managing software packages
- Adding additional software repositories based on need, and understand how it affects your RHEL 9 system support

- Configure the dnf command to update newly available software updates and security packages

Objectives

In this chapter, we set the following learning objectives: learn and understand the **`dnf`** tool to be able to install and update software packages for your RHEL 9 system. This tool is essential learning for sysadmins; and add and manage new software repositories to your RHEL 9 system so you would be able to install software packages outside those provided by Red Hat (such as proprietary or non-free software packages) and widen the capabilities of your system.

Now, let us start installing packages and managing them in your RHEL 9 system!

Recipe #22: Configure the DNF tool and update RHEL 9

By default, RHEL 9 uses **Dandified YUM** (**DNF**) package manager to install and manage RHEL 9 software packages delivered in the **Red Hat Package Manager** (**RPM**) format. DNF is also used in Fedora as well as the other RHEL-based distributions available. Previously, RHEL used **`yum`** *(*Yellowdog Updater Modified*)* as default but has since transitioned to DNF since RHEL 8. While yum still works, you should use **`dnf`** and the syntax and flags are similar. So, let us start exploring some basic DNF commands that you will be using in managing your RHEL 9 setup.

Before upgrading any package, it would be ideal to check first if there are updates available:

```
[jsg@rhel9-bpb ~]$ sudo dnf check-update
Updating Subscription Management repositories.
Last metadata expiration check: 2:03:03 ago on
Sat 14 Jan 2023 01:41:55 PM PST.
```

If package updates are available, you can upgrade a specific package only:

```
[jsg@rhel9-bpb ~]$ sudo dnf upgrade <package>
```

You can also upgrade all the packages available with the with the **`dnf`** upgrade command:

```
[jsg@rhel9-bpb ~]$ sudo dnf upgrade
```

If you wish to install a new package, you can do a search. If you are not familiar with the package name, you can use specific keywords:

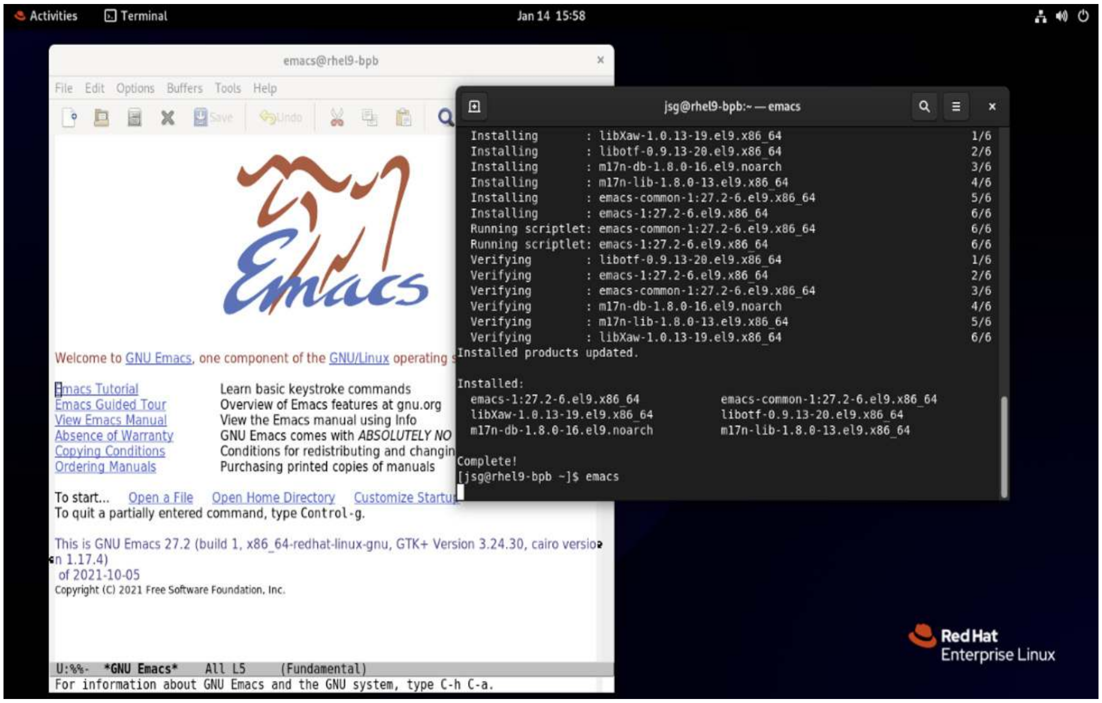
```
[jsg@rhel9-bpb ~]$ sudo dnf search emacs
Updating Subscription Management repositories.
Last metadata expiration check: 1:24:24 ago on
Sat 14 Jan 2023 01:41:55 PM PST.
```

```
======================== Name & Summary Matched: emacs =========================
emacs.x86_64 : GNU Emacs text editor
emacs-auctex.noarch : Enhanced TeX modes for Emacs
emacs-common.x86_64 : Emacs common files
emacs-filesystem.noarch : Emacs filesystem layout
emacs-lucid.x86_64 : GNU Emacs text editor with LUCID toolkit X support
emacs-nox.x86_64 : GNU Emacs text editor without X support
```

As shown in the search above, we can install the **`emacs.x86_64`** package. Let us do what with the install command. All package dependencies will also be installed to ensure **`emacs.x86_64`** package will run:

```
[jsg@rhel9-bpb ~]$ sudo dnf install emacs.x86_64
Updating Subscription Management repositories.
Last metadata expiration check: 2:12:09 ago on Sat 14 Jan 2023 01:41:55 PM PST.
Dependencies resolved.
================================================================================
Package       Arch    Version                     Repository                           Size
================================================================================
Installing:
emacs         x86_64  1:27.2-6.el9 rhel-9-for-x86_64-appstream-rpms  3.3 M
Installing dependencies:
emacs-common  x86_64  1:27.2-6.el9 rhel-9-for-x86_64-appstream-rpms   36 M
libXaw        x86_64  1.0.13-19.el9 rhel-9-for-x86_64-appstream-rpms  200 k
libotf        x86_64  0.9.13-20.el9 rhel-9-for-x86_64-appstream-rpms  106 k
m17n-db       noarch  1.8.0-16.el9 rhel-9-for-x86_64-appstream-rpms  661 k
m17n-lib      x86_64  1.8.0-13.el9 rhel-9-for-x86_64-appstream-rpms  201 k

Transaction Summary
================================================================================
Install  6 Packages

Total download size: 41 M
Installed size: 109 M
Is this ok [y/N]: y
Downloading Packages:
(1/6): libotf-0.9.13-20.el9.x86_64.rpm
 36 kB/s | 106 kB     00:02
(2/6): emacs-27.2-6.el9.x86_64.rpm
 775 kB/s | 3.3 MB     00:04
(3/6): emacs-common-27.2-6.el9.x86_64.rpm
 6.7 MB/s |  36 MB     00:05
(4/6): m17n-db-1.8.0-16.el9.noarch.rpm
 260 kB/s | 661 kB     00:02
(5/6): libXaw-1.0.13-19.el9.x86_64.rpm
 281 kB/s | 200 kB     00:00
(6/6): m17n-lib-1.8.0-13.el9.x86_64.rpm
 113 kB/s | 201 kB     00:01
--------------------------------------------------------------------
------------
Total
 6.6 MB/s |  41 MB     00:06
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
        1/1
  Installing       : libXaw-1.0.13-19.el9.x86_64
         1/6
  Installing       : libotf-0.9.13-20.el9.x86_64
          2/6
  Installing       : m17n-db-1.8.0-16.el9.noarch
           3/6
  Installing       : m17n-lib-1.8.0-13.el9.x86_64
           4/6
  Installing       : emacs-common-1:27.2-6.el9.x86_64
          5/6
  Installing       : emacs-1:27.2-6.el9.x86_64
          6/6
  Running scriptlet: emacs-common-1:27.2-6.el9.x86_64
          6/6
  Running scriptlet: emacs-1:27.2-6.el9.x86_64
            6/6
  Verifying        : libotf-0.9.13-20.el9.
x86_64                          1/6
  Verifying        : emacs-1:27.2-6.el9.
x86_64                            2/6
  Verifying        : emacs-common-1:27.2-6.el9.
x86_64                     3/6
  Verifying        : m17n-db-1.8.0-16.el9.noar
ch                          4/6
  Verifying        : m17n-lib-1.8.0-13.el9.
x86_64                         5/6
  Verifying        : libXaw-1.0.13-19.el9.
x86_64                          6/6
Installed products updated.

Installed:
  emacs-1:27.2-6.el9.x86_64           emacs-common-1:27.2-6.el9.
x86_64
  libXaw-1.0.13-19.el9.x86_64         libotf-0.9.13-20.el9.x86_64
  m17n-db-1.8.0-16.el9.noarch         m17n-lib-1.8.0-13.el9.x86_64

Complete!
```

You can now run the **`emacs`** from the console or launch it from GNOME as shown in *Figure 6.1:*



```
Installing       : libXaw-1.0.13-19.el9.x86_64                    1/6
Installing       : libotf-0.9.13-20.el9.x86_64                    2/6
Installing       : m17n-db-1.8.0-16.el9.noarch                    3/6
Installing       : m17n-lib-1.8.0-13.el9.x86_64                   4/6
Installing       : emacs-common-1:27.2-6.el9.x86_64               5/6
Installing       : emacs-1:27.2-6.el9.x86_64                      6/6
Running scriptlet: emacs-common-1:27.2-6.el9.x86_64               6/6
Running scriptlet: emacs-1:27.2-6.el9.x86_64                      6/6
Verifying        : libotf-0.9.13-20.el9.x86_64                    1/6
Verifying        : emacs-1:27.2-6.el9.x86_64                      2/6
Verifying        : emacs-common-1:27.2-6.el9.x86_64               3/6
Verifying        : m17n-db-1.8.0-16.el9.noarch                    4/6
Verifying        : m17n-lib-1.8.0-13.el9.x86_64                   5/6
Verifying        : libXaw-1.0.13-19.el9.x86_64                    6/6
Installed products updated.

Installed:
  emacs-1:27.2-6.el9.x86_64              emacs-common-1:27.2-6.el9.x86_64
  libXaw-1.0.13-19.el9.x86_64            libotf-0.9.13-20.el9.x86_64
  m17n-db-1.8.0-16.el9.noarch            m17n-lib-1.8.0-13.el9.x86_64

Complete!
[jsg@rhel9-bpb ~]$ emacs
```

```
Welcome to GNU Emacs, one component of the GNU/Linux operating s

Emacs Tutorial        Learn basic keystroke commands
Emacs Guided Tour     Overview of Emacs features at gnu.org
View Emacs Manual     View the Emacs manual using Info
Absence of Warranty   GNU Emacs comes with ABSOLUTELY NO
Copying Conditions    Conditions for redistributing and changin
Ordering Manuals      Purchasing printed copies of manuals

To start...   Open a File   Open Home Directory   Customize Startu
To quit a partially entered command, type Control-g.

This is GNU Emacs 27.2 (build 1, x86_64-redhat-linux-gnu, GTK+ Version 3.24.30, cairo versio
n 1.17.4)
 of 2021-10-05
Copyright (C) 2021 Free Software Foundation, Inc.

U:%%-  *GNU Emacs*    All L5     (Fundamental)
For information about GNU Emacs and the GNU system, type C-h C-a.
```

***Figure 6.1**: Emacs now running after a dnf install*

Recipe #23: Install and configure the EPEL repository

The world of open-source software is all about choice and this is a key consideration in adopting Linux in the enterprise with RHEL 9 as a top choice in enabling that strength. However, there would be cases where you wish to install a particular piece of software in RHEL 9, but it is not included in the officially supported software list and repositories provided by Red Hat. Given that RHEL 9 is based on the excellent development work done in Fedora, there is a wide range of software applications and platforms that are available in the latter but not included as part of the officially supported packages in the former.

This led to the creation of a **Special Interest Group** (**SIG**) within the Fedora Project called **Extra Packages for Enterprise Linux** (**EPEL**), which provides the additional packages that made Fedora great and made it available and compatible for RHEL and other RHEL-based distributions. EPEL is available as a repository that can be added to your RHEL 9 setup.

Now let us install and configure the EPEL repository in your RHEL 9 system. First, you need to enable the CodeReady Linux Builder repository. This is an official repository from Red Hat providing additional packages typically useful for software developers. Open a terminal and enable it:

```
[jsg@rhel9-bpb ~]$ sudo subscription-
manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
[sudo] password for jsg:
Repository 'codeready-builder-for-rhel-9-x86_64-
rpms' is enabled for this system.
```

Next, let us do a remote install of the EPEL RPM package from the Fedora Project repository:

```
[jsg@rhel9-bpb ~]$ sudo dnf install \
https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.
rpm
Updating Subscription Management repositories.
Last metadata expiration check: 0:00:34 ago on
Sat 14 Jan 2023 08:55:53 PM PST.
epel-release-latest-9.noarch.rpm
  18 kB/s |  18 kB     00:01
Dependencies resolved.
====================================================================
==================
Package                Architecture    Version
Repository              Size
====================================================================
==================
Installing:
epel-release           noarch          9-4.el9
 @commandline            18 k

Transaction Summary
====================================================================
==================
Install  1 Package

Total size: 18 k
Installed size: 25 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
```

```
  Preparing        :
                          1/1
  Installing       : epel-release-9-4.el9.noarch
                   1/1
  Running scriptlet: epel-release-9-4.el9.noarch
                   1/1
Many EPEL packages require the CodeReady Builder (CRB) repository.
It is recommended that you run /usr/bin/crb enable to
enable the CRB repository.

  Verifying        : epel-release-9-4.el9.noarch
                    1/1
Installed products updated.

Installed:
  epel-release-9-4.el9.noarch

Complete!
```

Once the EPEL RPM installed, update the repositories again with **`dnf update`**:

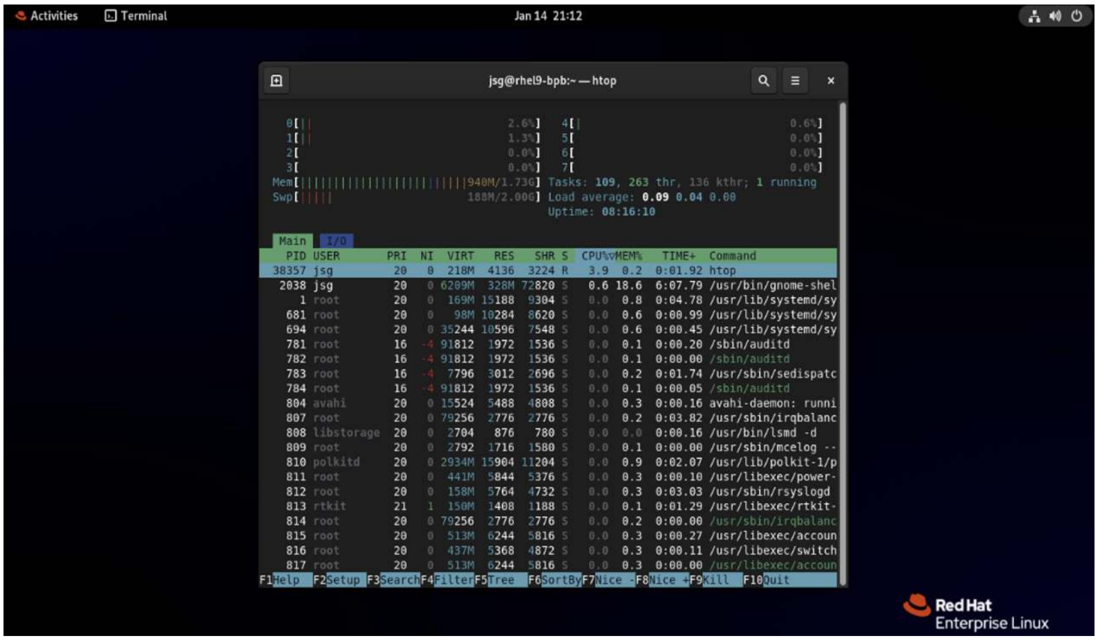
```
[jsg@rhel9-bpb ~]$ sudo dnf update
```

Now let us try and install an application. For this recipe, let us install **`htop`**, a popular interactive system monitor, process viewer, and process manager. This is a great alternative to the default **`top`** tool installed as it provides a better visual interface:

```
[jsg@rhel9-bpb ~]$ sudo dnf install htop.x86_64
Updating Subscription Management repositories.
Last metadata expiration check: 0:18:20 ago on
Sat 14 Jan 2023 08:59:06 PM PST.
Dependencies resolved.
============================================================================
==================
Package          Architecture       Version
Repository        Size
============================================================================
==================
Installing:
htop             x86_64             3.2.1-1.el9
epel             181 k

Transaction Summary
```

```
====================================================================
==================
Install  1 Package

Total download size: 181 k
Installed size: 428 k
Is this ok [y/N]: y
Downloading Packages:
htop-3.2.1-1.el9.x86_64.rpm
3.4 kB/s | 181 kB     00:53
--------------------------------------------------------------------
------------------
Total
3.3 kB/s | 181 kB     00:54
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
                  1/1
  Installing       : htop-3.2.1-1.el9.x86_64
                   1/1
  Running scriptlet: htop-3.2.1-1.el9.x86_64
                  1/1
  Verifying        : htop-3.2.1-1.el9.x86_64
                            1/1
Installed products updated.

Installed:
  htop-3.2.1-1.el9.x86_64

Complete!
```

From the command line, just type in **`htop`** and you will see the application run from the terminal as shown in *Figure 6.2:*



```
    0[||                              2.6%]   4[|                             0.6%]
    1[||                              1.3%]   5[                              0.0%]
    2[                                0.0%]   6[                              0.0%]
    3[                                0.0%]   7[                              0.0%]
  Mem[|||||||||||||||||||||||||940M/1.73G] Tasks: 109, 263 thr, 136 kthr; 1 running
  Swp[||||                   188M/2.00G] Load average: 0.09 0.04 0.00
                                          Uptime: 08:16:10

  Main  I/O
    PID USER       PRI  NI  VIRT   RES   SHR S  CPU%▽MEM%   TIME+  Command
  38357 jsg         20   0  218M  4136  3224 R   3.9  0.2  0:01.92 htop
   2038 jsg         20   0 6209M  328M 72820 S   0.6 18.6  6:07.79 /usr/bin/gnome-shel
      1 root        20   0  169M 15188  9304 S   0.0  0.8  0:04.78 /usr/lib/systemd/sy
    681 root        20   0   98M 10284  8620 S   0.0  0.6  0:00.99 /usr/lib/systemd/sy
    694 root        20   0 35244 10596  7548 S   0.0  0.6  0:00.45 /usr/lib/systemd/sy
    781 root        16  -4 91812  1972  1536 S   0.0  0.1  0:00.20 /sbin/auditd
    782 root        16  -4 91812  1972  1536 S   0.0  0.1  0:00.00 /sbin/auditd
    783 root        16  -4  7796  3012  2696 S   0.0  0.2  0:01.74 /usr/sbin/sedispatc
    784 root        16  -4 91812  1972  1536 S   0.0  0.1  0:00.05 /sbin/auditd
    804 avahi       20   0 15524  5488  4808 S   0.0  0.3  0:00.16 avahi-daemon: runni
    807 root        20   0 79256  2776  2776 S   0.0  0.2  0:03.82 /usr/sbin/irqbalanc
    808 libstorage  20   0  2704   876   780 S   0.0  0.0  0:00.16 /usr/bin/lsmd -d
    809 root        20   0  2792  1716  1580 S   0.0  0.1  0:00.00 /usr/sbin/mcelog --
    810 polkitd     20   0 2934M 15904 11204 S   0.0  0.9  0:02.07 /usr/lib/polkit-1/p
    811 root        20   0  441M  5844  5376 S   0.0  0.3  0:00.10 /usr/libexec/power-
    812 root        20   0  158M  5764  4732 S   0.0  0.3  0:03.03 /usr/sbin/rsyslogd
    813 rtkit       21   1  150M  1408  1188 S   0.0  0.1  0:01.29 /usr/libexec/rtkit-
    814 root        20   0 79256  2776  2776 S   0.0  0.2  0:00.00 /usr/sbin/irqbalanc
    815 root        20   0  513M  6244  5816 S   0.0  0.3  0:00.27 /usr/libexec/accoun
    816 root        20   0  437M  5368  4872 S   0.0  0.3  0:00.11 /usr/libexec/switch
    817 root        20   0  513M  6244  5816 S   0.0  0.3  0:00.00 /usr/libexec/accoun
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

***Figure 6.2**: htop running in RHEL 9*

Since EPEL is part of the Fedora Project, you can browse available packages at **https://packages.fedoraproject.org**.

You can also directly check the repository index at **https://docs.fedoraproject.org/en-US/epel/#what_packages_and_versions_are_available_in_epel**.

While EPEL provides additional software for your RHEL 9 system, do note that it is still a community-driven project, hence the packages that you get installed with EPEL are not part of the supported packages provided by Red Hat. You as the SysAd will have to manage and support the software installed through EPEL if you choose to include it as part of your RHEL 9 software repository. It is highly encouraged to subscribe to the low-volume epel-announce mailing list to receive updates and announcements related to EPEL. You can subscribe to the mailing list at this URL: **https://lists.fedoraproject.org/admin/lists/epel-announce@lists.fedoraproject.org/**.

Recipe #24: Install and configure RPM Fusion repositories

Another software repository to consider adding on RHEL 9 is the RPM Fusion. Like EPEL, RPM Fusion is maintained by a group of volunteers and the primary purpose of the project is to provide access to packages that are not included in Fedora due to various reasons, such as not being accepted in Fedora, software that are licensed as proprietary (also known as non-free), as well as software with strict redistribution policy due to copyright and license applied.

RPM Fusion classifies software packages with the following:

- **Free repository**: Software that is released under a free license but is not included in Fedora. The audio mixing editor *Audacity* is available in this repository.
- **Non-free repository**: Software that is not released under a free license but can be redistributed through RPM Fusion repository. *NVIDIA* drivers are available in this repository.
- **Free tainted**: Software that is distributed in a free license, but use is restricted in some countries. A good example of this would be the **`libdvdcss`** library which was a popular add-on to enable Linux desktops to play DVD content.
- **Non-free tainted**: Software that is non-free, but redistribution policy is not made clear by the creator of the said software. The **`kmod-nvidia-open`** package which provide the NVIDIA driver kernel module is available here.

So, in our RHEL 9 system let us include the RPM Fusion free and non-free repositories by opening up a terminal and do a **`dnf`** install as **`sudo`** user:

```
[jsg@rhel9-bpb ~]$ sudo dnf install --nogpgcheck https://mirrors.rpmfusion.org/free/el/rpmfusion-free-release-$(rpm -E %rhel).noarch.rpm https://mirrors.rpmfusion.org/nonfree/el/rpmfusion-nonfree-release-$(rpm -E %rhel).noarch.rpm
```

Type **`y`** to confirm that you wish to install RPM Fusion free and non-free repositories:

```
Updating Subscription Management repositories.
Last metadata expiration check: 23:35:36 ago on
Sat 14 Jan 2023 09:17:56 PM PST.
rpmfusion-free-release-9.noarch.rpm
3.8 kB/s |  10 kB     00:02
rpmfusion-nonfree-release-9.noarch.rpm
4.9 kB/s |  10 kB     00:02
Dependencies resolved.
====================================================================
============
Package                         Arch        Version
Repository         Size
====================================================================
============
Installing:
rpmfusion-free-release          noarch      9-1
@commandline       10 k
```

```
rpmfusion-nonfree-release      noarch      9-1
@commandline       10 k

Transaction Summary
====================================================================
============
Install  2 Packages

Total size: 21 k
Installed size: 7.7 k
Is this ok [y/N]: y
Downloading Packages:
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
           1/1
  Installing       : rpmfusion-free-release-9-1.noarch
           1/2
  Installing       : rpmfusion-nonfree-release-9-1.noarch
           2/2
  Verifying        : rpmfusion-free-release-9-1.noarch
            1/2
  Verifying        : rpmfusion-nonfree-release-9-1.noarch
           2/2
Installed products updated.

Installed:
  rpmfusion-free-release-9-1.noarch
rpmfusion-nonfree-release-9-1.noarch

Complete!
```

Next, do a **dnf** update to refresh the RPM Fusion repositories that was just added:

```
[jsg@rhel9-bpb ~]$ sudo dnf update
Updating Subscription Management repositories.
RPM Fusion for EL 9 - Free - Updates
79 kB/s | 244 kB     00:03
```

```
RPM Fusion for EL 9 - Nonfree - Updates                    19 kB/s |  63 kB     00:03
Dependencies resolved.
Nothing to do.
Complete!
```

Finally, let us test if RPM Fusion packages are available. Try to do a search of **`kmod-nvidia`**:

```
[jsg@rhel9-bpb ~]$ sudo dnf search kmod-nvidia
Updating Subscription Management repositories.
Last metadata expiration check: 0:03:24 ago on Sun 15 Jan 2023 08:57:34 PM PST.
================================ Name Exactly Matched: kmod-nvidia ================================
kmod-nvidia.x86_64 : Metapackage which tracks in nvidia kernel module for newest kernel
==================================== Name Matched: kmod-nvidia ====================================
akmod-nvidia.x86_64 : Akmod package for nvidia kernel module(s)
akmod-nvidia-340xx.x86_64 : Akmod package for nvidia-340xx kernel module(s)
akmod-nvidia-470xx.x86_64 : Akmod package for nvidia-470xx kernel module(s)
akmod-nvidia-open.x86_64 : Akmod package for nvidia-open kernel module(s)
kmod-nvidia-340xx.x86_64 : Metapackage which tracks in nvidia-340xx kernel module for newest kernel
kmod-nvidia-340xx-5.14.0-162.el9_1.x86_64 : nvidia-340xx kernel module(s) for 5.14.0-162.el9_1
kmod-nvidia-470xx.x86_64 : Metapackage which tracks in nvidia-470xx kernel module for newest kernel
kmod-nvidia-470xx-5.14.0-162.el9_1.x86_64 : nvidia-470xx kernel module(s) for 5.14.0-162.el9_1
kmod-nvidia-5.14.0-162.el9_1.x86_64 : nvidia kernel module(s) for 5.14.0-162.el9_1
kmod-nvidia-open.x86_64 : Metapackage which tracks in nvidia-open kernel module for newest kernel
kmod-nvidia-open-5.14.0-162.el9_1.x86_64 : nvidia-open kernel module(s) for 5.14.0-162.el9_1
```

RPM Fusion aims to make it easier for users to download and install these types of software and the project only includes packages that are legally redistributable. You can check the full list of packages available at **https://rpmfusion.org**.

Recipe #25: Manage RHEL 9 software packages

As shown in *Recipe #22*, **dnf** is a handy tool for installing new packages and updating your RHEL 9 system. And the use of the tool is not just limited to packages provided by Red Hat, but also to the repositories we added in *Recipes #23* (EPEL) and *#24* (RPM Fusion). Now let us go through some more in-depth use of **dnf** in managing the packages in your RHEL 9 system.

To check the list of repositories that are active in our RHEL 9 system, do a **repolist**:

```
[jsg@rhel9-bpb ~]$ sudo dnf repolist
Updating Subscription Management repositories.
repo id                                            repo name
codeready-builder-for-rhel-9-x86_64-rpms
Red Hat CodeReady Linux Builder for RHEL 9 x86_64 (RPMs)
epel
Extra Packages for Enterprise Linux 9 - x86_64
rhel-9-for-x86_64-appstream-rpms
Red Hat Enterprise Linux 9 for x86_64 - AppStream (RPMs)
rhel-9-for-x86_64-baseos-rpms
Red Hat Enterprise Linux 9 for x86_64 - BaseOS (RPMs)
rpmfusion-free-updates
RPM Fusion for EL 9 - Free - Updates
rpmfusion-nonfree-updates
RPM Fusion for EL 9 - Nonfree - Updates
```

You can also list both active and inactive repositories available by invoking **repolist all**:

```
[jsg@rhel9-bpb ~]$ sudo dnf repolist all
```

The command will provide a list of the enabled repositories as well as disabled repositories. This would be a long list, so the output of the command is not shown here.

If you wish to enable a repository and install a particular package, you can invoke it in one line:

```
[jsg@rhel9-bpb ~]$ sudo dnf --enablerepo=repo-name install package
```

Another thing to consider in managing software packages is to be able to list all installed packages. From the terminal you can list all installed packages with the list installed command:

```
[jsg@rhel9-bpb ~]$ sudo dnf list installed
```

This would be a long list, so we did not include the output. But you will be provided with details of the packages along with information on what repository it was installed from.

When you need to check what packages were installed in the past week, do a **`dnf list recent`** command:

```
[jsg@rhel9-bpb ~]$ sudo dnf list recent
```

So, let us say you need to check available packages that can be installed from enabled repositories. Aside from doing a search, you can just invoke a **`dnf list available`** command. Like in previous examples, the list would be long so we are not including the output here.

```
[jsg@rhel9-bpb ~]$ sudo dnf list available
```

In *Recipe #22*, we installed the **`emacs`** package along with the required dependencies. We can remove Emacs by using the **`remove`** command. But that would only remove the Emacs package and not the dependencies that we installed earlier, so those would be left in the system and would be pretty much unuseful and just eating up disk space. So, it is better to remove both Emacs along with the dependencies that would not be needed anymore. To do that, use the **`autoremove`** command:

```
[jsg@rhel9-bpb ~]$ sudo dnf autoremove emacs.x86_64
[sudo] password for jsg:
Updating Subscription Management repositories.
Dependencies resolved.
====================================================================
============
Package       Arch    Version         Repository
          Size
====================================================================
============
Removing:
emacs         x86_64  1:27.2-6.el9
@rhel-9-for-x86_64-appstream-rpms   16 M
Removing unused dependencies:
emacs-common  x86_64  1:27.2-6.el9
@rhel-9-for-x86_64-appstream-rpms   89 M
libXaw        x86_64  1.0.13-19.el9
@rhel-9-for-x86_64-appstream-rpms  498 k
libotf        x86_64  0.9.13-20.el9
@rhel-9-for-x86_64-appstream-rpms  240 k
m17n-db       noarch  1.8.0-16.el9
@rhel-9-for-x86_64-appstream-rpms  1.9 M
m17n-lib      x86_64  1.8.0-13.el9
@rhel-9-for-x86_64-appstream-rpms  458 k

Transaction Summary
```

```
===================================================================
============
Remove  6 Packages

Freed space: 109 M
Is this ok [y/N]: y
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
                   1/1
  Running scriptlet: emacs-1:27.2-6.el9.x86_64
                  1/6
  Erasing          : emacs-1:27.2-6.el9.x86_64
                  1/6
  Erasing          : m17n-lib-1.8.0-13.el9.x86_64
                2/6
  Erasing          : libotf-0.9.13-20.el9.x86_64
              3/6
  Erasing          : m17n-db-1.8.0-16.el9.noarch
                4/6
  Erasing          : libXaw-1.0.13-19.el9.x86_64
                5/6
  Running scriptlet: emacs-common-1:27.2-6.el9.x86_64
                 6/6
  Erasing          : emacs-common-1:27.2-6.el9.x86_64
               6/6
  Running scriptlet: emacs-common-1:27.2-6.el9.x86_64
                6/6
  Verifying        : emacs-1:27.2-6.el9.x86_64
                1/6
  Verifying        : emacs-common-1:27.2-6.el9.x86_64
                2/6
  Verifying        : libXaw-1.0.13-19.el9.x86_64
                 3/6
  Verifying        : libotf-0.9.13-20.el9.x86_64
                    4/6
  Verifying        : m17n-db-1.8.0-16.el9.noarch
                  5/6
  Verifying        : m17n-lib-1.8.0-13.el9.x86_64
                  6/6
Installed products updated.

Removed:
  emacs-1:27.2-6.el9.x86_64            emacs-common-1:27.2-6.el9.
x86_64
  libXaw-1.0.13-19.el9.x86_64          libotf-0.9.13-20.el9.x86_64
  m17n-db-1.8.0-16.el9.noarch          m17n-lib-1.8.0-13.el9.x86_64

Complete!
```

You will notice that **`emacs`**, along with the dependencies, are now removed from your RHEL 9 system.

Lastly, you can remove dependencies in your RHEL 9 system that is not used anymore with the **`autoremove`** command:

```
[jsg@rhel9-bpb ~]$ sudo dnf autoremove
Updating Subscription Management repositories.
Last metadata expiration check: 2:33:33 ago on
Sat 14 Jan 2023 01:41:55 PM PST.
Dependencies resolved.
====================================================================
============
Package           Arch   Version          Repository
        Size
====================================================================
============
Removing:
grub2-tools-efi   x86_64 1:2.06-46.el9
@rhel-9-for-x86_64-baseos-rpms 2.7 M
grub2-tools-extra x86_64 1:2.06-46.el9
@rhel-9-for-x86_64-baseos-rpms 5.3 M
python3-dmidecode x86_64 3.12.2-27.el9
@anaconda                      272 k

Transaction Summary
====================================================================
============
Remove  3 Packages

Freed space: 8.3 M
Is this ok [y/N]: y
```

```
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
           1/1
  Erasing          : python3-dmidecode-3.12.2-27.el9.x86_64
             1/3
  Erasing          : grub2-tools-extra-1:2.06-46.el9.x86_64
           2/3
  Erasing          : grub2-tools-efi-1:2.06-46.el9.x86_64
            3/3
  Running scriptlet: grub2-tools-efi-1:2.06-46.el9.x86_64
            3/3
  Verifying        : grub2-tools-efi-1:2.06-46.el9.x86_64
           1/3
  Verifying        : grub2-tools-extra-1:2.06-46.el9.x86_64
             2/3
  Verifying        : python3-dmidecode-3.12.2-27.el9.x86_64
             3/3
Installed products updated.

Removed:
  grub2-tools-efi-1:2.06-46.el9.x86_64
grub2-tools-extra-1:2.06-46.el9.x86_64
  python3-dmidecode-3.12.2-27.el9.x86_64

Complete!
```

Recipe #26: Automate software updates with DNF Automatic

In cases where automatic software updates and security patches installation is desired, RHEL 9 provides a handy tool for this called DNF Automatic. The tool parameters are set in a configuration file and the command will execute based on the information or data set in the said file. DNF Automatic can be set to execute the following outcomes:

- **Notify and exit**: Sends a notification after running the DNF Automatic tool checked updates.
- **Download the updates**: Downloads the updated packages only.

- **Download and install the updates**: Downloads the packages then installs them, similar to how you run `dnf` on-demand.

To enable DNF Automatic, install the **`dnf-automatic`** package from the console:

```
[jsg@rhel9_bpb ~]$ sudo dnf install dnf-automatic
```

Next, validate with the **`rpm`** command if **`dnf-automatic`** is already installed:

```
[jsg@rhel9-bpb ~]$ sudo rpm -qi dnf-automatic
Name        : dnf-automatic
Version     : 4.12.0
Release     : 4.el9
Architecture: noarch
Install Date: Sat 14 Jan 2023 12:58:27 PM PST
Group       : Unspecified
Size        : 53144
License     : GPLv2+
Signature   : RSA/SHA256, Thu 15 Sep 2022 10:39:24 PM PST,
Key ID 199e2f91fd431d51
Source RPM  : dnf-4.12.0-4.el9.src.rpm
Build Date  : Thu 15 Sep 2022 08:05:47 PM PST
Build Host  : x86-64-02.build.eng.rdu2.redhat.com
Packager    : Red Hat, Inc. <http://bugzilla.redhat.com/bugzilla>
Vendor      : Red Hat, Inc.
URL         : https://github.com/rpm-software-management/dnf
Summary     : Package manager - automated upgrades
Description : Systemd units that can periodically
download package upgrades and apply them.
```

Once validated, check the **`/etc/dnf/automatic.conf`** file by opening it with a text editor and set the actions or behaviors of the tool as well as the time you wish **`dnf-automatic`** to run. The configuration file is well commented on the parameters you must set and has four key sections with the following settings by default:

`[commands]`

`upgrade_type = default` (this default setting executes all available upgrades. This can also be set as **`security`** to do only security upgrades).

`network_online_timeout = 60` (defines the maximum time in seconds that the tool is connecting to Red Hat repositories. The default setting is **`60`** seconds).

`download_updates = yes` (updates will be downloaded when made available. Setting this to **`no`** will just check updates but will not download them).

`apply_updates = no` (updates will not be applied by default. If set to **`yes`**, updates will be applied after download).

`[emitters]`

`emit_via = stdio` (sets how the **`dnf-automatic`** sends messages. By default, it is set to **`stdio`** which is useful when setting up a cron job to send the message. Other options include **`email`** and **`motd`**. If left blank, no messages will be sent).

`[email]`

`email_from = root@example.com` (if **`emit_via`** is set to **`email`**, the messages will be sent from the email address set here).

`email_to = root` (you can set the addresses where you want messages to be received).

`email_host = localhost` (set the host that would send the email message).

`[command_email]`

`email_from = root@example.com` (this is the shell command executed from the background to send the email message to the recipients sent from the [email] section).

`email_to = root`

When the **`automatic.conf`** file input parameters are set, as **`sudo`** user, enable **`systemctl`** to enable DNF Automatic:

```
[jsg@rhel9-bpb ~]$ sudo systemctl enable --now dnf-automatic.timer
```

Once enabled, check if the timers are now set:

```
[jsg@rhel9-bpb ~]$ systemctl list-timers dnf-*
NEXT                         LEFT       LAST
  PASSED      >
Sat 2023-01-14 15:12:13 PST 15min left
Sat 2023-01-14 13:41:51 PST 1h 14min ago>
Sun 2023-01-15 06:02:21 PST 15h left   n/a
        n/a         >

2 timers listed.
Pass --all to see loaded but inactive timers, too.
lines 1-6/6 (END)
```

While automating updates provide benefits on speed, continuous security coverage, and convenience, as a SysAd, automating the installation of software updates and security patches is a decision that needs careful consideration: you may opt to select a few systems that would benefit on this. For critical production systems, holding off on automatic updates to ensure system uptime would be a more prudent path to choose.

Conclusion

In this chapter, we went through the basics, as well as some advanced use cases on managing software in your RHEL 9 system using the **`dnf`** command. The tool syntax and use are comprehensive and can be the only one you need in managing RHEL 9. We have also added some useful community-developed repositories that can add additional software packages. Although Red Hat does not officially support them, they add additional functionalities that would be useful in deploying your platform solutions with RHEL 9.

Much of the tools and commands we have done in this chapter were invoked from the command line interface.

In the next chapter, we will go through some basics on managing RHEL 9 using the GNOME graphical user interface!

Points to remember

- EPEL and RPM Fusion repositories are not officially supported by Red Hat. These repositories are and remain to be community-developed, so please check your organization's policy on using the software packages provided in these repositories.
- The use of DNF Automatic is also dependent on your organization's policy. As explained in *Recipe #26*, mission-critical and general production systems are not the best candidates to use this configuration.
- If you are using RHEL 9 as a workstation and have an NVIDIA graphics card installed in the workstation, RPM Fusion provides access to the NVIDIA drivers and kernel modules needed so you can make the most out of your hardware set up.

CHAPTER 7

Managing RHEL 9 with GNOME Desktop

Introduction

A default RHEL 9 installation is set up with a server + GUI configuration, meaning your RHEL 9 server will have a default desktop environment installed along with some basic server software.

In this chapter, we will go through some details on navigating and managing your RHEL 9 server installation with GNOME, a free and open-source desktop environment for Linux and similar operating systems. **GNU Network Object Model Environment** (**GNOME**) development started in 1997. Development of GNOME is managed by the GNOME Project, composed of volunteers working alongside corporate sponsored contributors to the project. Red Hat is currently one of the biggest corporate sponsors of the project.

We will dive deeper into GNOME, navigating the user interface, and using the tools included to manage our RHEL 9 system.

Structure

In this chapter, we will discuss the following topics:

- The GNOME Desktop environment
- GNOME Software, the application manager within the GNOME Desktop environment
- Configuring and managing printers in GNOME
- Modifying GNOME settings to end user requirements

Objectives

In this chapter, you will understand how to use and navigate the GNOME Desktop environment. You will also go through the steps in installing and managing software with the GNOME Software Application manager, along with storage management, setting up printers, and creating user-specific configurations for GNOME.

Now let us begin by going through the basics of using your GNOME Desktop!

Recipe #27: Running an application using GNOME

There are two (2) methods to run or launch an application in the GNOME Desktop environment:

Running an application in the standard GNOME session:

1. From the GNOME Desktop click the **Activities** shortcut at the top left of your screen. A search bar will appear at the top center, as well as the shortcuts of default application. The default applications include **Firefox**, **Files**, Software, Help, Terminal as well as the Show Applications.
2. From the search bar, type the application you wish to run. In the example below as shown in *Figure 3.1*, start to type the letters `f` and `i` to start the search and run Firefox. The search then shows apps available, as well as other search results. When the `firefox` is completed on the search bar, the icon for Firefox browser appears.

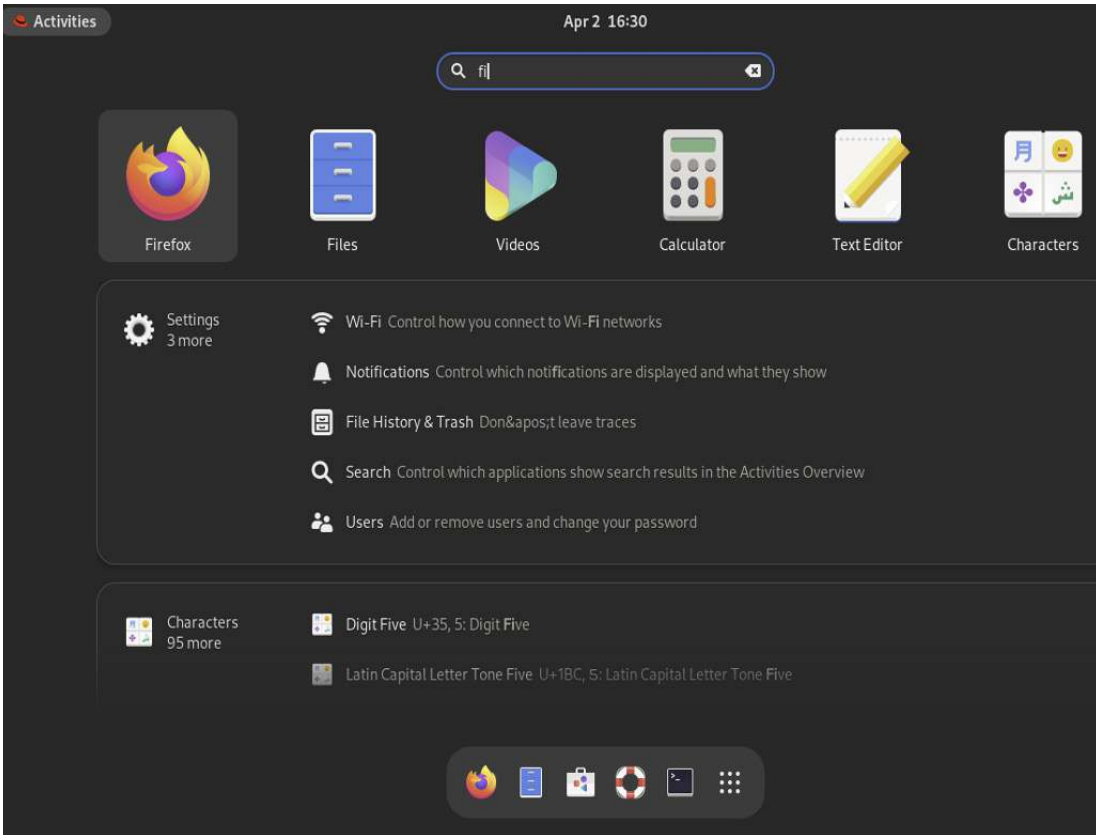

Activities
Apr 2 16:30
fi
Firefox
Files
Videos
Calculator
Text Editor
Characters
Settings
3 more
Wi-Fi Control how you connect to Wi-Fi networks
Notifications Control which notifications are displayed and what they show
File History & Trash Don&apos;t leave traces
Search Control which applications show search results in the Activities Overview
Users Add or remove users and change your password
Characters
95 more
Digit Five U+35, 5: Digit Five
Latin Capital Letter Tone Five U+1BC, Ƽ: Latin Capital Letter Tone Five

***Figure 7.1**: Searching for Firefox web browser*

3. Another way to run an application is to click the shortcut icons at the bottom part of the desktop. If the application you want to run is not available in the default shortcuts, click the Show Applications icon and you will get a list of available applications installed in your GNOME Desktop.

Using a Run command in GNOME

1. From the desktop, open a **Run a Command** window by pressing *Alt* + *F2*.
2. Type the application in the command space provided and press *Enter*. In *Figure 3.2* below, the **`firefox`** browser is launched through the **Run a Command** window:

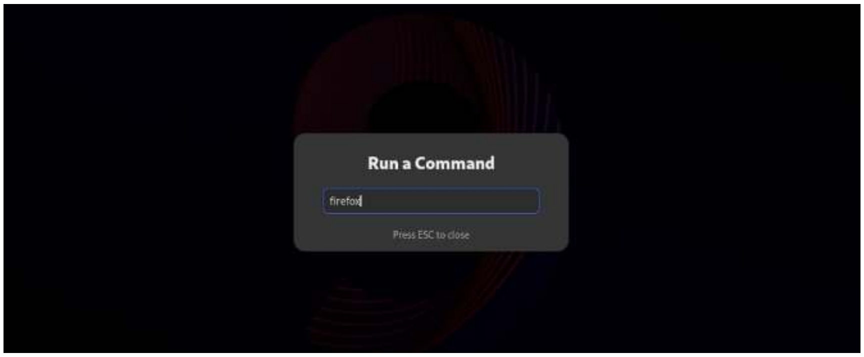


***Figure 7.2**: Running the firefox browser from the Run a Command window*

3. You can also run an application from the terminal by typing the name of the application and pressing *Enter*. Note that the terminal will stay open while you are running an application. You can create a new terminal session if you wish to use the command line for other uses.

Recipe #28: Manage and install applications with GNOME Software application manager

You can install new applications and manage existing installed ones (if they are graphical applications) with GNOME Software application manager (or Software). Available applications are grouped into categories for easy referencing, and you can also do a search using keywords and Software will provide recommendations based on the search results.

Now let us go through the steps in installing a new graphical application with GNOME Software application manager:

1. **Run the GNOME Software application manager tool**: From GNOME Desktop, click on Activities at the top left of your desktop. From here, you can choose between 3 methods to run Software:
   a. From the search bar, just type **`Software`** and the application icon will be shown.
   b. From the bottom panel of GNOME Desktop, click the icon that looks like a white briefcase. This icon launches the GNOME Software application manager.
   c. From GNOME, open a terminal and launch Software with the **`gnome-software`** command.
2. **Understanding the GNOME Software application manager tool**: When GNOME Software application manager is open, you will see an interface like what is show in in *Figure 3.3* below:

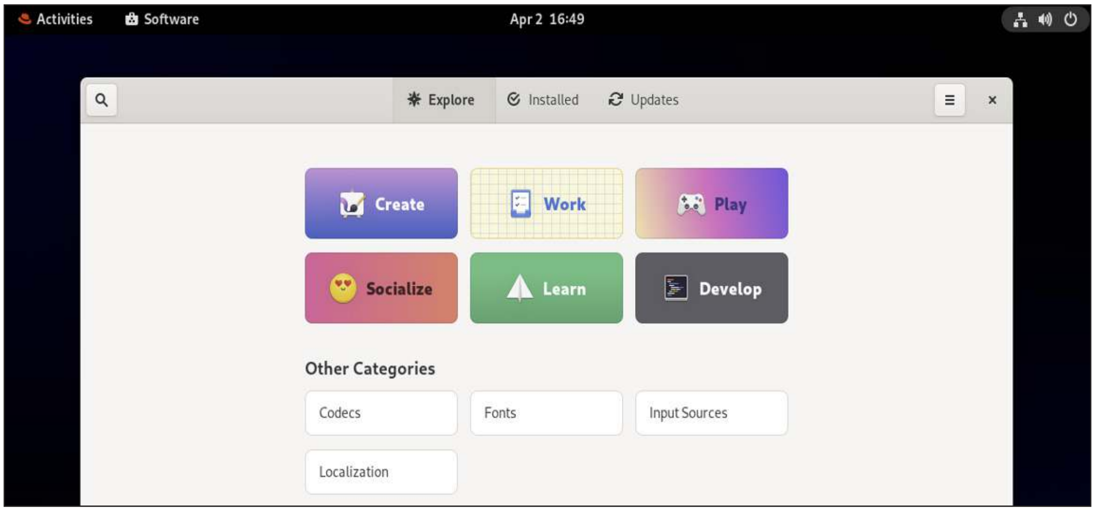


Figure 7.3*: The GNOME Software application interface*

Now let us go through some details on the application interface:

a. **Search icon**: To look for an application, you can click on the magnifying glass icon at the top left of the interface. Clicking this icon will activate a search bar in the middle where you can search for new apps that you can install.

b. **Explore**: This would be shown by default and lets you navigate, and list applications classified in categories.

c. **Installed**: A list of installed applications will be shown here. You can also uninstall applications that are not used anymore by clicking on the Uninstall button.

d. **Updates**: If there are system and application updates, you can install them from this section.

e. **Settings**: By clicking on the icon with 3 lines at the top right of the interface, some application settings can be configured such as activating additional software repositories or deactivating current ones, setting update preferences such as automatic updates and notifications, and reading the application information and credits in the About section of the application.

3. **Installing an application**: To install an application, select through the options presented in GNOME Software application. You can use either search or explore options as explained in the previous section. In the example below in *Figure 3.4*, the **GNU Emacs** editor is being installed and the installation status is shown. All the dependencies required to install **GNU Emacs** are being installed in the background as well.

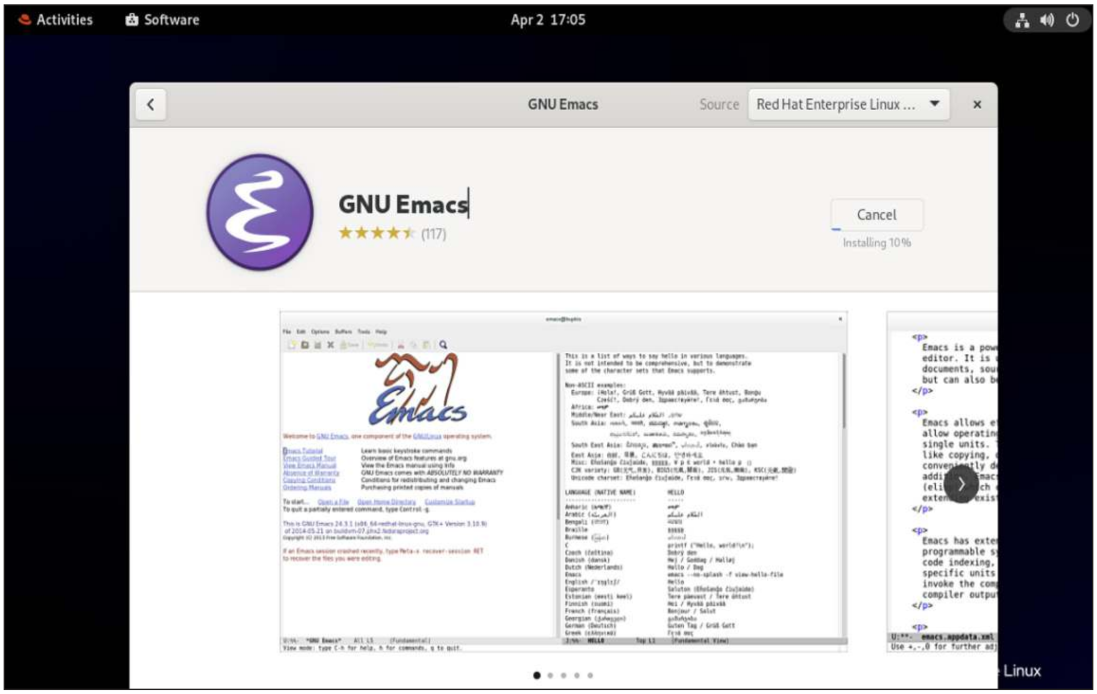
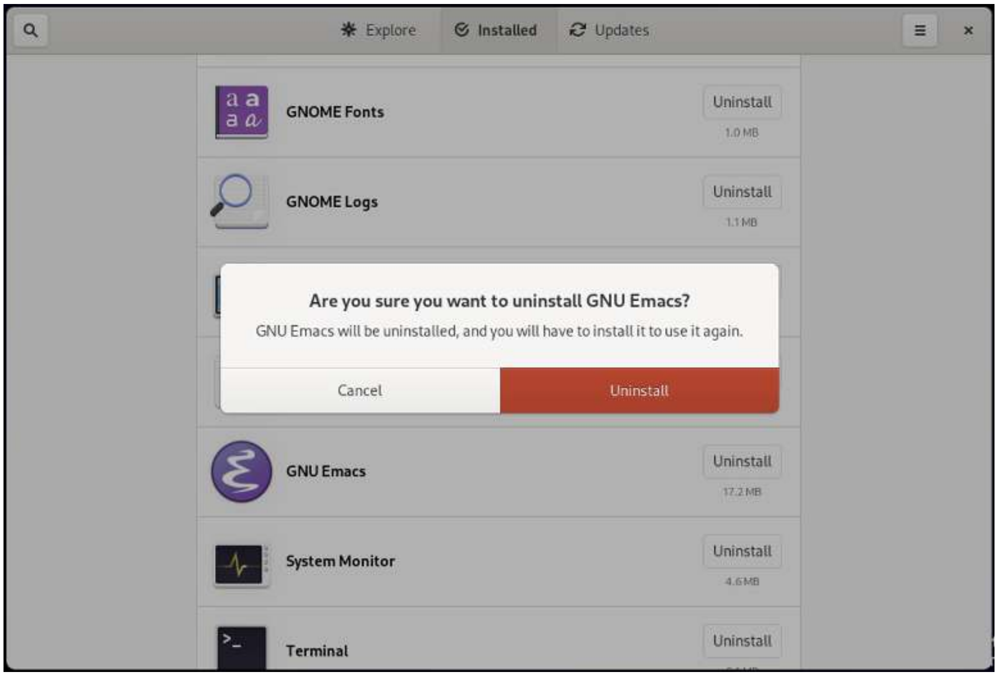


***Figure 7.4**: Installing GNU Emacs editor with GNOME Software application tool*

Once installation is complete, the **Cancel** button will change to Open. The **GNU Emacs** icon will also appear when the Show Applications shortcut is clicked from GNOME Desktop.

4. **Manage installed applications**: You can check the list of applications installed and manage them by selecting Installed in GNOME Software application manager. Installed applications will be shown and you have the option to remove an application (along with its dependencies) by selecting and clicking the **Uninstall** button as shown in *Figure 3.5*:



Are you sure you want to uninstall GNU Emacs?

GNU Emacs will be uninstalled, and you will have to install it to use it again.

***Figure 7.5**: Uninstalling the GNU Emacs application*

5. **Updating applications**: If an application or system update is available, details will appear in the **Updates** section of GNOME Software application manager. You can go through the detailed information presented and apply the update whenever you want to. If no updates are available, **Updates** will show that your system is up to date like in *Figure 3.6:*

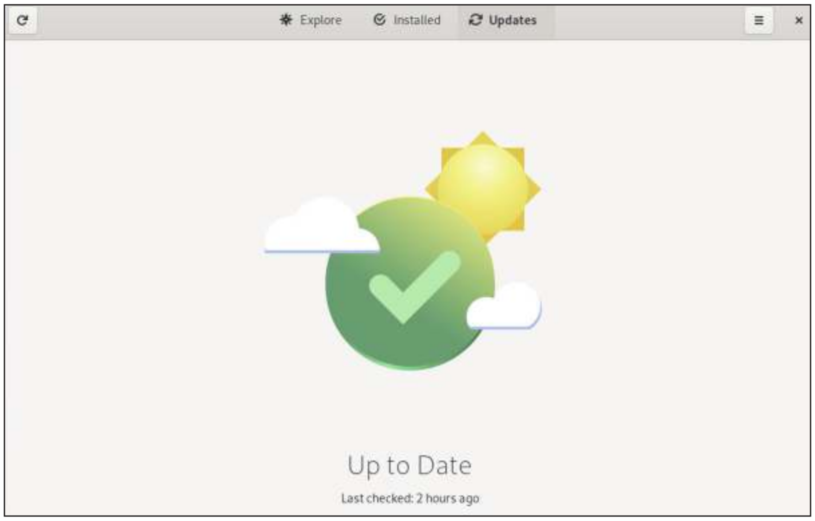


Figure 7.6

Recipe #29: Managing storage in GNOME

The Disks utility tool in GNOME is used primarily for managing storage in your GNOME Desktop. You can run the Disks utility from Show Applications, select Utilities, and select Disks. Let us go through some basic operations in managing storage in GNOME.

Check the disk space and usage

1. **Using the Disk Usage Analyzer tool**: From Utilities, click on Disk Usage Analyzer to open the disk visualization tool. You can select between the Home folder and the localhost disk to show a graphical representation of disk usage based on the partitions set by RHEL 9.
2. **Using System Monitor**: From Utilities, click on System Monitor to run the application. Select the File Systems tab to show the physical and virtual devices attached to your RHEL 9 system. The Used column shows how much disk space in the partition is used.

Manage volumes and partitions

You can check and make changes to the storage volumes with the Disk utility.

1. From Utilities, click on Disks to run the Disk utility.
2. From the left pane, you can select between virtual disks, CD/DVD drives, block devices, and partitions. Select the device to check and make changes.
3. Once a device is selected, the details of the storage, capacity and usage, as well as disk operation options are presented in the right pane. Do take care using this tool as it is possible to wipe out storage using the tools provided. You can deactivate a partition and select additional partition tool options such as format, resize, and test the disk with a benchmark tool provided.
4. Note that the Disk utility can destroy a partition along with stored data if not used properly. Do go through the options in detail before making any changes to your disks and volumes.

Recipe #30 Setting up a printer and configuring printer settings

You can connect your GNOME Desktop environment to a printer available from a network. This would be useful when you need to print documents or reports that are generated in your RHEL 9 system. To set up and configure a printer, go through the following steps:

1. Click the system menu located at the top right corner of GNOME and select **Settings** as shown in *Figure 3.7* You can also open **Settings** from Show Applications at the bottom of your desktop.

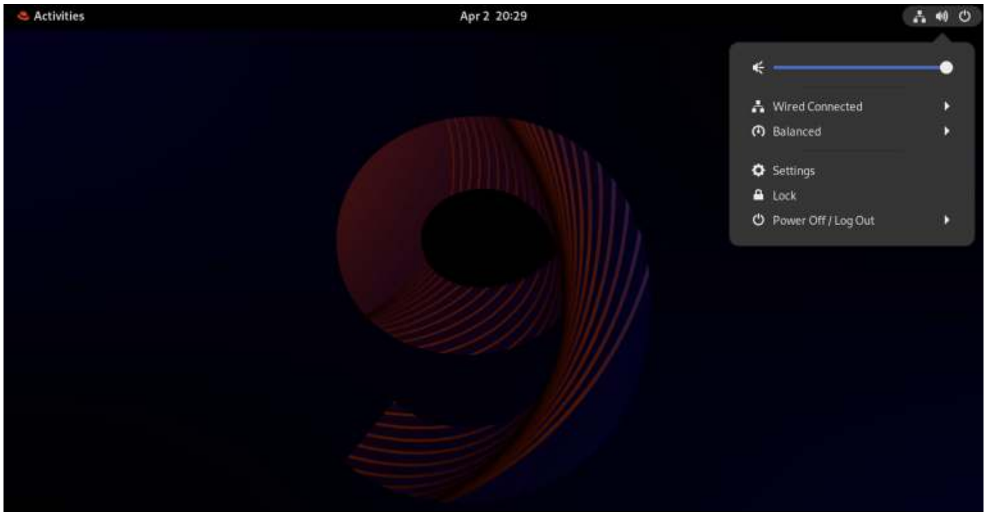


***Figure 7.7**: Click Settings from the System Menu*

2. From **Settings**, select Printers from the left. You need to Unlock the Printers settings as a superuser or provided admin rights as part of sudousers before making any changes. Use your login password with admin rights to unlock and configure a printer.
3. Once unlocked, click the button **Add Printer** and GNOME will do a search of available printers from the network (even active printers connected to Windows machines) as shown in *Figure 3.8:*

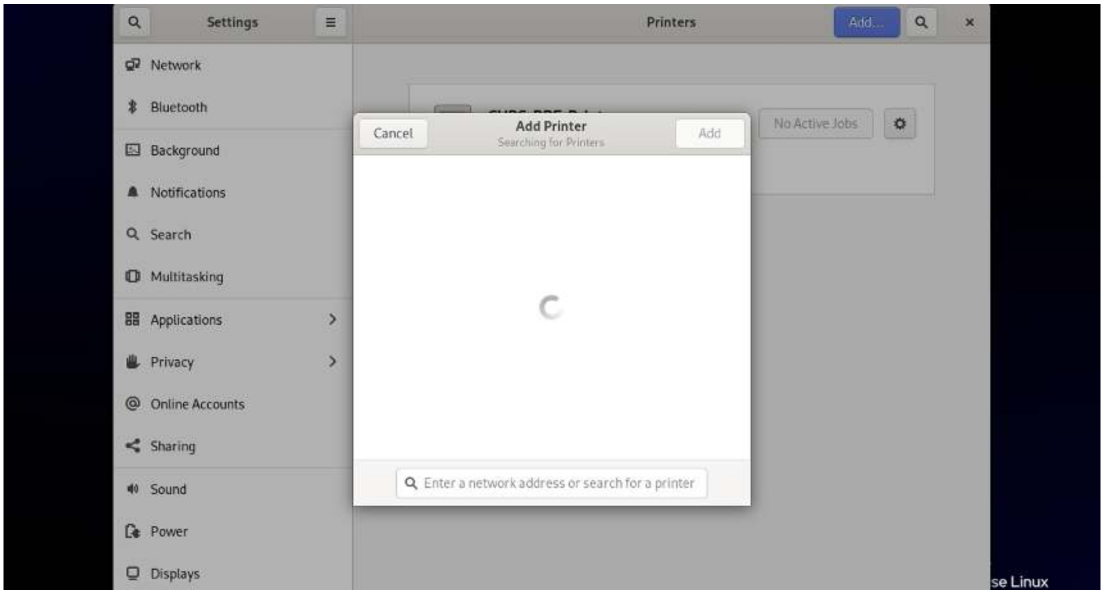


Figure 7.8

4. When you see the list of available printers that you can connect to, select the printer and click **Add** to finalize the set up.
5. To confirm that your printer is working properly, you can choose to print a test page. From the list of printers, select the settings button of the printer you would like to print a test page and select Printing Options.
6. Click the Test Page button at the top left of the window and your printer will print a test page to confirm that the settings are working properly.

Recipe #31: Customizing the GNOME environment

GNOME is a customizable desktop environment that can be configured and modified to your needs. The following are some of the key customizations that you can do out of the box and using extended tools to change the way you work on GNOME and your RHEL 9 system.

Enabling text input for other languages

If you need to write text from a different language, say like Chinese, Korean, or Japanese, you need to configure your RHEL 9 system for additional text input for other languages. You will need to install the required **Input Method Engine** (**IME**) for this. This can be done during the installation setup but if you are not able to add additional language support or IMEs, it can be done by post-installation.

1. To start, you need to install the **`input-methods`** package from the command line:

```
[jsg@rhel9_bpb ~]$ sudo dnf install @input-methods
```

2. Once the **`input-methods`** package is installed, go to **Settings**, select **Keyboard**, and click the + button to add a new text input. A list of input sources will be provided and if the list does not show the language you wish to include, click the icon with the three dots at the bottom to show more languages as shown in *Figure 3.9*:

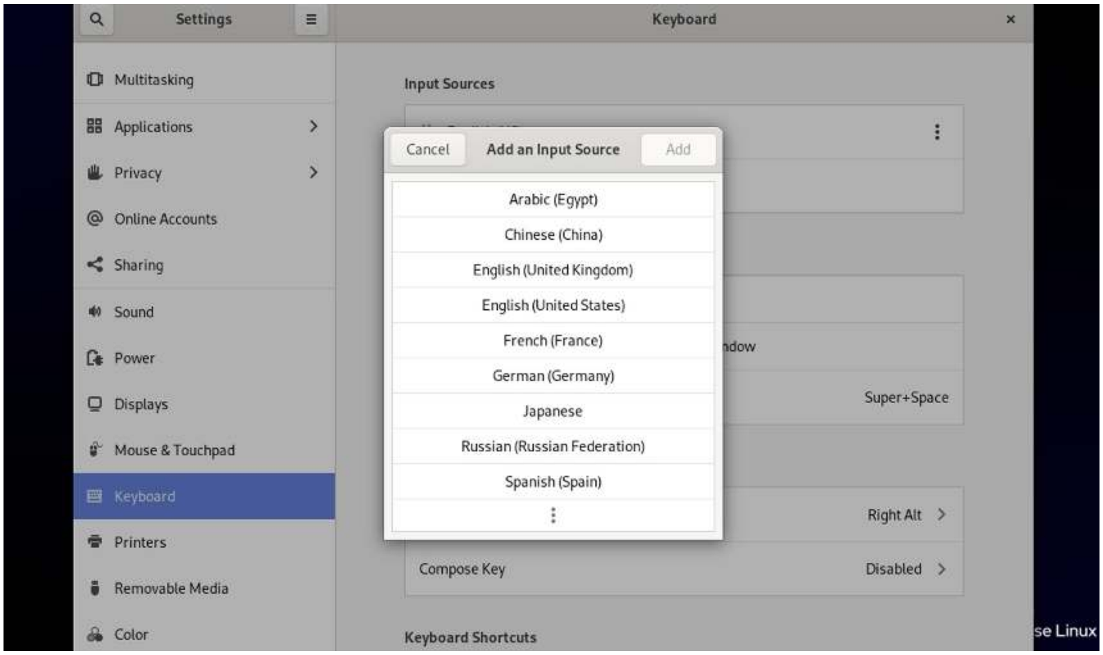


***Figure 7.9**: Adding a new Input Source*

3. Select the language input and click **Add** to include in the **Input Sources** of your GNOME settings.

4. To verify that you have new input sources, a language icon will be added in system settings at the top right of your desktop as shown in *Figure 3.10:*

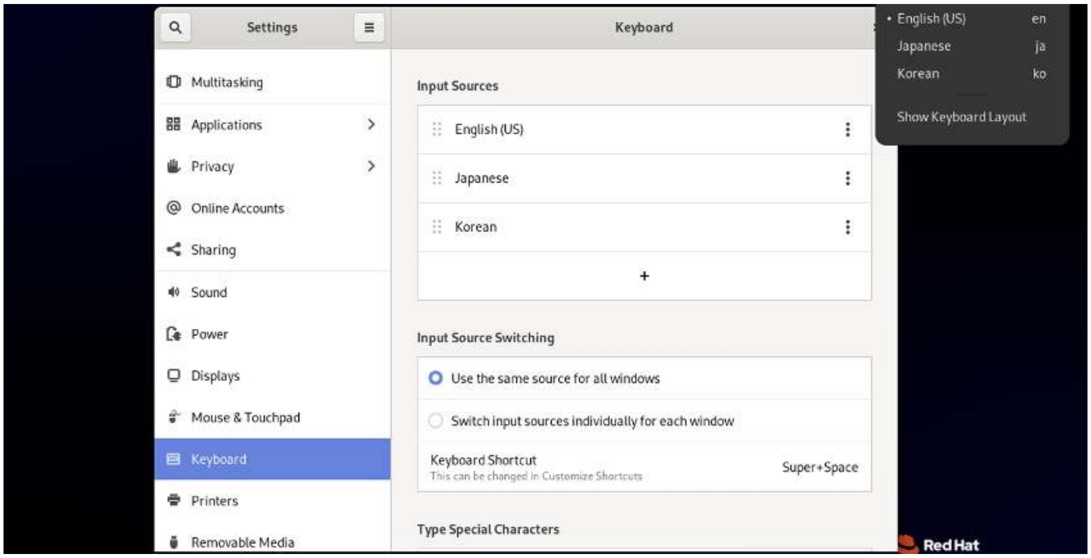


Figure 7.10

5. To switch between language inputs, you can select from the language icon at the top or switch between language inputs using the *Super* + *Spacebar* shortcut. The *Super* key is usually the assigned *Windows* key of your keyboard.

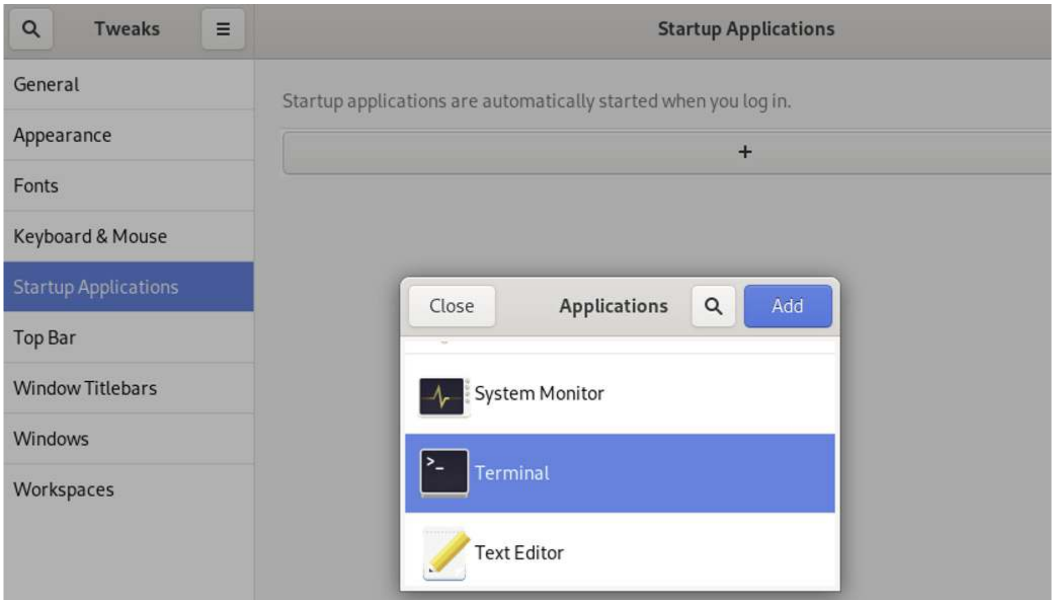
Setting up an application to run automatically upon login in GNOME

You can also set GNOME to launch identified applications upon login. This makes certain applications set to be available whenever. You login to your GNOME session and can be handy at times. To set up automatic run or launch applications, you need to install a tool called GNOME Tweaks:

```
[jsg@rhel9_bpb ~]$ sudo dnf install gnome-tweaks
```

1. Once GNOME Tweaks is installed, open the application from the Utilities group from Show Applications.

2. Select **Startup Applications** on the left navigation pane and click on the + sign button to add an application. In *Figure 3.11* below, the **Terminal** application was added to run at startup:



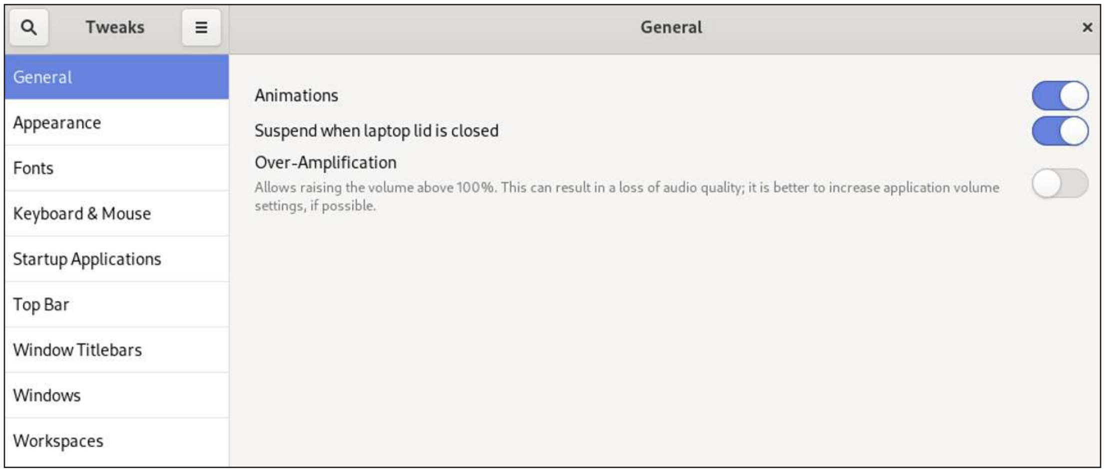
Startup applications are automatically started when you log in.

***Figure 7.11**: Adding the Terminal application to run on startup upon login in GNOME*

3. Close the GNOME Tweaks app. Now every time a login to GNOME Desktop is executed, the terminal application will run on startup and ready to be used.

Using GNOME Tweaks to customize your GNOME Desktop environment experience

You may have noticed already that setting up an application to run on startup is just one of the things you can do with GNOME Tweaks.



Animations

Suspend when laptop lid is closed

Over-Amplification

Allows raising the volume above 100%. This can result in a loss of audio quality; it is better to increase application volume settings, if possible.

***Figure 7.12**: GNOME Tweaks*

You can customize the look and feel of your GNOME Desktop. Let us go through the list of Tweaks that you can configure in your desktop:

- **General**: You can configure to turn off animations on the desktop, turn on suspend when you have configured RHEL 9 on a laptop, and over-amplify the volume of the system. However, this is not recommended, and application based volume control is still the preferred method.
- **Appearance**: You can change the theme details, backgrounds, and lock screen of GNOME.
- **Fonts**: You can make changes to the fonts as well as the hinting, antialiasing, and scale factor of the fonts displayed.
- **Keyboard & Mouse**: You can make changes to keyboard and mouse behavior, as well as enable input sources through this setting.
- **Startup Applications**: You can set certain applications to run on startup as explained in the previous section of this recipe.
- **Top Bar**: You can enable clock options, calendar week numbers, battery percentage and hot corners from the top bar.
- **Window Titlebars**: You can change the behavior of the title bars when you click on

- **Windows**: You can change the behavior of Windows when dragged to screen edges, resize when clicked, assign the behavior of the Windows action key, and set the keyboard actions for the Window focus.
- **Workspaces**: You can switch between Dynamic and Static workspaces, set the maximum number of static workspaces, and display handling of workspaces when you are using multiple displays.

Conclusion

The GNOME Desktop environment is a powerful graphical user interface with a robust set of tools and utilities that allow you to manage and navigate your RHEL 9 system with ease. From managing disks to adding additional software, GNOME provides the tools needed to get your job done as a SysAdmin managing a RHEL 9 system. This chapter is a basic introduction of using GNOME and in the succeeding chapters, we will be using it further in building and managing our RHEL 9 system.

In the next chapter, we will learn how to install and manage software applications related to infrastructure and databases. These applications will make your RHEL 9 system powerful enough to handle enterprise-level workloads.

Points to remember

- Take care in using the Disk utility tool in GNOME. It is quite powerful in functionality and not using it properly may lead to unrecoverable data loss.
- The GNOME Software Application Manager tool provides the same set of functionalities and features compared to that of the `dnf` command line tool. You have the option to select which method fits your style of working.
- Use the GNOME Tweaks tool to customize your desktop experience. However, if you have multiple users sharing the GNOME Desktop, they may have to re-run the tool again to customize their desktop environment.

CHAPTER 8
Managing Infrastructure and Databases

Introduction

In the previous chapters, we have installed and configured RHEL 9 along with the necessary features we need to prepare the system for functions we need to implement solutions. Majority of the recipes in this chapter will revolve around installing and configuring specific solution stacks that are needed to run solutions for your day-to-day operations, like a warehouse inventory system, or a development workstation for writing code. Because of the software available in RHEL 9, along with the additional software repositories we configured, we have more than enough software available that can be used for your solution stacks.

This is a comprehensive chapter that covers the essentials you need to build a resilient solution infrastructure on RHEL 9. Go through the recipes as you need them and reference back from previous chapters and recipes that you may need to do as prerequisites prior to implementing the recipes in this chapter. Do not worry you will be guided along the way to ensure you are on the right track when implementing the recipes!

Structure

In this chapter, we will cover the following topics:

- Installing a basic LAMP stack

- NGNIX
- HAProxy
- Varnish Cache
- Squid proxy cache
- MySQL
- PostgreSQL
- MariaDB
- MongoDB Community Edition
- CockroachDB
- Neo4j
- Cassandra
- Microsoft SQL Server

Objectives

In this chapter, you will learn to install and configure key infrastructure platforms that are available in RHEL 9. You will also understand the basic configuration options for installed web stacks and databases and be able to implement basic security features prior to placing the installed and configured platforms online.

Recipe #32: Install and manage a basic LAMP stack

A **Linux, Apache, MySQL, and PHP** (**LAMP**) stack is the most common and popular platform stack used in building web applications. In this recipe, we will go through the steps in implementing a LAMP stack. In this guide, we will use MariaDB instead of MySQL, as the database component of our stack. MariaDB is community developed, commercially supported, and fully compatible with MySQL.

1. **Update the system and install the Apache Web Server**

   Open up the console and update the system with **`dnf`**. Once the system is updated, install the *Apache Web Server*. After installation, start the **`httpd`** service, and enable it to start every time the RHEL 9 system boots up.

```
[jsg@rhel9_bpb ~]$ sudo dnf update
[jsg@rhel9_bpb ~]$ sudo dnf install httpd
[jsg@rhel9_bpb ~]$ sudo systemctl start httpd
[jsg@rhel9_bpb ~]$ sudo systemctl enable httpd
```

2. **Install and configure MariaDB**

   From the console, install the MariaDB database and enable the service to start every time your RHEL 9 system boots up.

```
[jsg@rhel9_bpb ~]$ sudo dnf install mariadb-server mariadb
[jsg@rhel9_bpb ~]$ sudo systemctl start mariadb
[jsg@rhel9_bpb ~]$ sudo systemctl enable mariadb
```

   After the MariaDB set up, run the **`mysql_secure_installation`** script to improve the security of the MariaDB installation. The script allows you to secure the database with the following:

   - Set a password for the root account
   - Remove root accounts that are accessible remotely
   - Remove anonymous user accounts
   - Remove the test database, which by default is accessible by anonymous users

   From the console, run the **`mysql_secure_installation`** script:

```
[jsg@rhel9_bpb ~]$ sudo mysql_secure_installation
```

   Follow the prompts to set a root password, remove anonymous users, disable remote root login, and remove the test database. When asked if you want to reload the privilege tables, select Yes.

3. **Install and configure PHP**

   PHP is a general-purpose scripting language popular for web development. Released in 1995, much of the growth of the internet is attributed to the popularity of PHP. To complete our LAMP stack, let us install and configure PHP from the console:

```
[jsg@rhel9_bpb ~]$ sudo dnf install php php-mysqlnd
```

   After installing PHP, restart the Apache Web Server to reload the PHP libraries and modules and work alongside Apache:

```
[jsg@rhel9_bpb ~]$ sudo systemctl restart httpd
```

4. **Verify the LAMP installation**

   To verify that the LAMP stack is installed correctly, you can create a quick PHP info page. From a text editor create a new file called **`info.php`**:

```
[jsg@rhel9_bpb ~]$ sudo vi /var/www/html/info.php
```

   In the **`info.php`** file, put the following content:

```
<?php
```

```
phpinfo();
?>
```

Save and exit the **info.php** file. From a desktop, open a web browser and visit the address **http://your_server_IP_address/info.php** in your web browser. Put the server IP address in place of the URL placeholder. If you see a page displaying your PHP information, your LAMP stack is installed correctly. Remember to remove this file after checking to avoid exposing information about your server's PHP configuration:

```
[jsg@rhel9_bpb ~]$  sudo rm /var/www/html/info.php
```

Remember to always follow best security practices when setting up a server on the internet. This guide is a basic setup and does not include steps for securing your server from threats.

Recipe #33: Install and configure NGINX

NGINX is a web server that can also be configured as a load balancer, reverse proxy, mail proxy and HTTP cache. This versatile server has been the primary choice among sysads on building robust, public-facing web infrastructure. Let us go through the steps on installing and configuring NGINX on RHEL 9.

Open a console and update the package repository. If you have not added the EPEL repository which contains NGINX, it would be a good time to do so as well. For more information about the EPEL repository, make sure to check *Recipe #23* from *Chapter 6, Configuring Software Repositories and RHEL 9 Updates.*

```
[jsg@rhel9_bpb ~]$ sudo dnf update
[jsg@rhel9_bpb ~]$ sudo dnf install epel-release
```

1. After the EPEL repository is installed and enabled, install NGINX and start the **nginx** service with **systemctl**:

```
[jsg@rhel9_bpb ~]$ sudo dnf install nginx
[jsg@rhel9_bpb ~]$ sudo systemctl start nginx
```

2. Make sure to check the status of the **nginx** service if it is running and configure it to start automatically every time your system boots up.

```
[jsg@rhel9_bpb ~]$ sudo systemctl status nginx
[jsg@rhel9_bpb ~]$ sudo systemctl enable nginx
```

By default, NGINX listens on port 80. If you want to change the port number or add additional ports, you can do so by editing the NGINX configuration file at **/etc/nginx/nginx.conf** with a text editor. Make sure to restart the nginx service whenever you make changes to the configuration file for the changes to take effect.

To test your NGINX installation, open a web browser in another machine and enter the configured and running, you should see the default NGINX welcome page. If all goes well as described, you have just completed the NGINX installation and setup for RHEL 9.

Recipe #34: Install and manage HAProxy

HAProxy is an open-source proxy and load balancing software that provides high availability for both application and network layers, improving the speed and performance through distribution of the workload and traffic across multiple platforms and servers. Let us go through the steps of installing HAProxy in RHEL 9:

1. **Install HAProxy**

   From the console, install HAproxy as a **`sudo`** user.

   ```
   [jsg@rhel9_bpb ~]$ sudo dnf install haproxy
   ```

   Once HAProxy is installed, open the configuration file located at **`/etc/haproxy/haproxy.cfg`**. The configuration file is well-documented, and you can refer to the official HAProxy documentation and start guide at **https://docs.haproxy.org** to start configuring basic features such as proxying, SSL, monitoring, HA, etc.

2. **Configure HAProxy with systemctl**

   When you have finished configuring, start the HAProxy service with **`systemctl`**:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl start haproxy
   ```

   It would also be a good idea to enable HAProxy to start every time you boot your RHEL 9 system. Like in starting the service, enable this with **`systemctl`**:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl enable haproxy
   ```

   You can also check and monitor the **`status`** of the HAProxy service from the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl status haproxy
   ```

   This will give you information about the current status of HAProxy, including whether it is running, and any errors that may have occurred during startup.

3. **Testing HAProxy and additional configuration**

   To test that HAProxy is working correctly, you can use a web browser to access them by using the IP address or hostname of the server. If the service is configured and working properly, you should see the default HAProxy welcome page.

   If you need to make additional changes to the configuration of HAProxy, you can open the configuration file at **`/etc/haproxy/haproxy.cfg`** with a text editor and make the changes needed. Once done, restart the HAProxy service with **`systemctl`**:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl restart
   ```

Recipe #35: Install and manage Varnish Cache

Varnish Cache is a popular reverse proxy software used to accelerate web applications by caching the contents retrieved via HTTP/HTTPS protocol. In this recipe, we will go through the steps on installing and configuring Varnish Cache on RHEL 9:

1. Update the system and install Varnish Cache from the console, update your software repositories and install the latest package:

```
[jsg@rhel9_bpb ~]$ sudo dnf update
[jsg@rhel9_bpb ~]$ sudo dnf install varnish
```

2. After the installation, start and enable the Varnish Cache service to start whenever your RHEL 9 system boots up:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start varnish
[jsg@rhel9_bpb ~]$ sudo systemctl enable varnish
```

3. Check the status of Varnish Cache after installation:

```
[jsg@rhel9_bpb ~]$ sudo systemctl status varnish
```

4. Now it is time to configure Varnish Cache through the configuration file. The main configuration file is located at **`/etc/varnish/default.vcl`**. This configuration file defines how Varnish will function. From the console, use a text editor like **`nano`** to edit the **`default.vcl`** file:

```
[jsg@rhel9_bpb ~]$ sudo nano /etc/varnish/default.vcl
```

Replace **`127.0.0.1`** and **`8080`** with the IP address and port of your backend server:

```
backend default {
    .host = "127.0.0.1";
    .port = "8080";
}
```

Save and close the file when finished.

5. Varnish listens on port **`6081`** by default. To configure Varnish to listen to a specific port (say port 80,) you need to edit the Varnish service file. To do this, create a new file **`varnish.service`** in the **`/etc/systemd/system/`** folder:

```
[jsg@rhel9_bpb ~]$ sudo nano /etc/systemd/system/varnish.service
```

Paste the following content in the **`varnish.service`** file:

```
.include /lib/systemd/system/varnish.service

[Service]
ExecStart=
ExecStart=/usr/sbin/varnishd -j unix,user=vcache -F -a
:80 -T localhost:6082 -f /etc/varnish/default
.vcl -S /etc/varnish/secret -s malloc,256m
```

In the configuration file above, **`-a :80`** means that Varnish will listen on port **`80`**. If your web server is using a different port, adjust this as necessary for your configuration.

After making the above changes, reload the **`systemctl`** daemon and restart the Varnish service.

```
[jsg@rhel9_bpb ~]$ sudo systemctl daemon-reload
```

```
[jsg@rhel9_bpb ~]$ sudo systemctl restart varnish
```

You can then check that Varnish is now listening on the correct port:

```
[jsg@rhel9_bpb ~]$ sudo netstat -tunlp | grep varnish
```

6. If you are running a firewall, you would need to adjust it to allow traffic on the port where Varnish is listening. Adjust the firewall settings with **`firewalld`**:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --zone=public
--add-port=80/tcp --permanent
```

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

To test that Varnish is working correctly, open a console session and send a request to your server and check the headers in the response:

```
[jsg@rhel9_bpb ~]$ sudo curl -I http://localhost
```

You should see a result like the one below:

```
HTTP/1.1 200 OK
...
X-Varnish: 2
Age: 0
...
```

The **`X-Varnish`** header indicates that the response has passed through Varnish. The **`Age`** header indicates how long the content has been cached; an **`Age`** of **`0`** means the content was just fetched from the backend web server.

7. To monitor the performance, Varnish provides the tool **`varnishstat`** that you can use to monitor its performance in real-time. From the console, simply run the tool:

```
[jsg@rhel9_bpb ~]$ varnishstat
```

Varnish Cache is a very powerful tool that would help your web applications perform faster. Remember to adjust the settings to suit your own system and needs, test, and apply the changes accordingly as provided in this recipe.

Recipe #36: Install and configure Squid proxy cache

Squid is a popular caching proxy that supports HTTP, HTTPS, FTP, and other protocols. It is generally used as an accelerator to speed up access to web pages by optimizing the data flow from the web server to client web browsers. The caching is done on frequently accessed content that allows the web host to save bandwidth. Squid is particularly useful when web sites suddenly experience a surge in traffic and allows to increase the serving capacity of web servers that use the cached content when serving to web clients. In this recipe, we will go through the steps on installing and configuring Squid proxy cache in your RHEL 9 system:

1. Like in previous recipes, we always ensure our software packages from the repository are updated using the **`dnf`** command. Once updated, we install the Squid package:

   ```
   [jsg@rhel9_bpb ~]$ sudo dnf update
   [jsg@rhel9_bpb ~]$ sudo dnf install squid
   ```

2. Once installation is completed, configure Squid with a text editor. The main configuration file for Squid is located at **`/etc/squid/squid.conf`**:

   ```
   [jsg@rhel9_bpb ~]$ sudo nano /etc/squid/squid.conf
   ```

   The **`squid.conf`** file contains a lot of options to configure the caching server. Below is a simple configuration that provides caching services. Go through each line especially the services that would use Squid and make changes as you need:

   ```
   # Squid normally listens to port 3128
   http_port 3128

   # We recommend you to use at least the following line.
   hierarchy_stoplist cgi-bin ?

   # Uncomment and adjust the following to add a disk cache directory.
   #cache_dir ufs /var/spool/squid 100 16 256

   # Leave coredumps in the first cache dir
   coredump_dir /var/spool/squid

   # Add any of your own refresh_pattern entries above these.
   refresh_pattern ^ftp:           1440    20%     10080
   refresh_pattern ^gopher:        1440    0%      1440
   refresh_pattern -i (/cgi-bin/|\?) 0     0%      0
   refresh_pattern .               0       20%
   ```

In the configuration file above, Squid is enabled to listen for HTTP requests on port 3128 and run some basic caching rules.

Now let us enable the Squid service to start on boot and start the service:

```
[jsg@rhel9_bpb ~]$ sudo systemctl enable squid
[jsg@rhel9_bpb ~]$ sudo systemctl start squid
```

Check the status of the Squid proxy cache service:

```
[jsg@rhel9_bpb ~]$ sudo systemctl status squid
```

3. After verifying that Squid is running, let us adjust the **`firewall`** with the firewalld service to allow port **`3128`** to be accessible:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-port=3128/tcp
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

4. Finally, let us test if the Squid proxy cache service is indeed running through the IP address of your RHEL 9 machine. Use the IP address of your proxy server host machine in place of **`http://the-squid-proxy-ip`**:

```
[jsg@rhel9_bpb ~]$ curl -x http://the-squid-proxy-ip:3128 https://
bpbonline.com/
```

The curl command should return some HTML elements of **https://bpbonline.com** which confirms your Squid proxy cache is working.

Recipe #37: Install and manage MySQL

MySQL is a popular open-source relational database server and is one of the most popular database packages installed in Linux systems as part of the basic LAMP stack. Being open-source, it has become a popular choice as an alternative to proprietary database systems. Oracle currently leads development and distribution of MySQL and organizations can obtain official support directly from Oracle when running mission-critical applications. In this recipe, let us go through the basics of setting up a MySQL server on RHEL 9:

1. RHEL 9 ships MySQL 8 as part of the Application Stream repository. Ensure that your repository sources are updated and install the **`mysql-server`** package from the console:

```
[jsg@rhel9_bpb ~]$ sudo dnf install mysql-server
```

2. Once you have installed the **`mysql-server`** package, start the **`mysqld`** service and enable it to start whenever you boot up your RHEL 9 system:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start mysqld.service
[jsg@rhel9_bpb ~]$ sudo systemctl enable mysqld.service
```

3. To improve the security of your MySQL server setup, run the **`mysql_secure_installation`**

prompts to secure your setup which includes setting up the MySQL root password, removing anonymous users, and disabling remote logins by the root user outside the host machine:

```
[jsg@rhel9_bpb ~]$ sudo mysql_secure_installation
```

After completing the prompts, your MySQL database server on RHEL 9 is now ready for use.

Recipe #38: Install and manage PostgreSQL

PostgreSQL, also known as Postgres, is another popular open-source relational database platform available in RHEL 9. The version provided in the Application Stream repository is PostgreSQL 13. Let us go through the steps for installing and configuring Postgres in RHEL 9. Ensure that your software repositories are updated prior to installing.

From the console, install the **`postgresql-server`** package and initialize the database cluster:

```
[jsg@rhel9_bpb ~]$ sudo dnf install postgresql-server
[jsg@rhel9_bpb ~]$ sudo postgresql-setup --initdb
```

1. Start the **`postgresql`** service and enable it to start whenever your RHEL 9 system boots up:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start postgresql.service
[jsg@rhel9_bpb ~]$ sudo systemctl enable postgresql.service
```

2. Postgres allows you to create various database user types depending on permissions and access. Postgres has a default database superuser named Postgres and access to this user can be configured in the **`pg_hba.conf`** file. It is also possible to create new users as well as database superusers with rights or privileges that would allow that user to create and manage a database. The following Postgres commands are standard database management privileges: **`SELECT`**, **`INSERT`**, **`DELETE`**, **`UPDATE`**, **`REFERENCES`**, **`TRIGGER`**, **`TRUNCATE`**, **`CREATE`**, **`EXECUTE`**, **`CONNECT`**, **`USAGE`**, **`TEMPORARY`**. Special privileges include **`SUPERUSER`**, **`LOGIN`**, **`CREATEDB`**, and **`CREATEROLE`**.

3. Now let us create a user, set the password, and assign privileges for **`CREATEROLE`** and **`CREATEDB`**. From the console, do the following command:

```
[jsg@rhel9_bpb ~]$ sudo postgres=# CREATE USER myuser WITH
PASSWORD 'password' CREATEROLE CREATEDB;
```

Make sure to change **`myuser`** with the desired username and **`password`** with the desired password. Once completed, you are ready to use PostgreSQL on your RHEL 9 system. This recipe covered only the basics of installing and configuring Postgres on your RHEL 9 system. To get more information and detailed documentation, head to the official PostgreSQL 13 documentation page at **https://www.postgresql.org/docs/13/index.html**.

Recipe #39: Install and manage MariaDB

MariaDB is another popular relational database server that is maintained by the community but commercially supported by partners of the MariaDB Foundation. It is a fork of the MySQL database and most, if not all aspects of installing, configuring, and managing MySQL can be applied to MariaDB. In RHEL 9, MariaDB 10.5 is provided in the Application Stream repository and can be easily installed from the console. Given that MySQL and MariaDB are generally similar in code base as well as package dependencies, it is not possible to have both databases installed in a RHEL 9 installation as some installation packages are in conflict with each other. As a SysAd, it is important to determine whether to use MySQL or MariaDB in your setup. Now let us go through the steps in installing MariaDB in RHEL 9:

1. From the console, update your software repositories to ensure the latest packages are pulled, and then install the **`mariadb-server`** package:

```
[jsg@rhel9_bpb ~]$ sudo dnf update
[jsg@rhel9_bpb ~]$ sudo dnf install mariadb-server
```

   Start the MariaDB service and enable it to start at boot:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start mariadb.service
[jsg@rhel9_bpb ~]$ sudo systemctl enable mariadb.service
```

2. You can also run the **`mysql_secure_installation`** script as discussed in *Recipe #32* and *#37* to secure your MariaDB installation (like setting the database root password), just like how we configured secure settings for a MySQL database installation.

   MariaDB has extensive documentation on managing the database as well as building high availability and performance tuning for enterprise environments. You can obtain more detailed documentation and how to from the official MariaDB documentation website at **https://mariadb.com/kb/en/documentation/**.

Recipe #40: Install and manage MongoDB Community Edition

MongoDB is a cross platform document-oriented database platform. It is considered as a NoSQL database and stores data in a type of JSON format called BSON and is binary encoded (compared to the former which is text-based). BSON as a format makes it more efficient for transferring over the network. MongoDB is a popular choice for NoSQL to build highly available and scalable online applications. Because of its flexible document schema, it adapts well to development teams that adopt an agile methodology approach in building cloud-native applications. Now let us go through the steps in deploying and configuring MongoDB on RHEL 9:

1. With a text editor, create the MongoDB repo file at **`/etc/yum.repos.d/mongodb-org-6.0.repo`** and add the repository info as shown below then save the file:

   ```
   [mongodb-org-6.0]
   name=MongoDB Repository
   baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-
   org/6.0/x86_64/
   gpgcheck=1
   enabled=1
   gpgkey=https://www.mongodb.org/static/pgp/server-6.0.asc
   ```

2. Update your repositories and install the **`mongodb-org`** package:

   ```
   [jsg@rhel9_bpb ~]$ sudo yum update -y
   [jsg@rhel9_bpb ~]$ sudo yum install -y mongodb-org
   ```

3. Start the **`mongod`** service and enable it to start on system boot:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl start mongod
   [jsg@rhel9_bpb ~]$ sudo systemctl enable mongod
   ```

4. You can also check the status of the **`mongod`** service to make sure everything is working fine:

   ```
   [jsg@rhel9_bpb ~]$ sudo systemctl status mongod
   ```

   After verifying that everything is now working properly and configured, login to MongoDB by accessing the database shell from the console:

   ```
   [jsg@rhel9_bpb ~]$ sudo mongosh
   ```

The MongoDB documentation site provides comprehensive guides, samples, and references to manage the database, manipulate and visualize data, as well as obtain code samples on building applications using the platform. You can visit the documentation page at **https://www.mongodb.com/docs/**.

Recipe #41: Install and manage CockroachDB

CockroachDB is a source-available, cloud-native distributed SQL database. It is designed to survive software and hardware failures and is popularly known to power popular online services like *Netflix* and *DoorDash*. It is also a popular solution adopted in the financial services industry. CockroachDB can be configured to store copies of data in multiple locations for quick access. CockroachDB is fairly new but becoming quite popular. In this

1. Download the latest full binary release of CockroachDB for Linux at this webpage: **https://www.cockroachlabs.com/docs/releases/index.html**. As of this writing, the latest production release is version v24.2 and is available for both Intel and ARM architectures.

2. Extract the downloaded **`.tgz`** archive file and copy the cockroach binary as a **`sudo`** user into your **`PATH`** settings so you can use the **`cockroach`** command from the console:

   ```
   [jsg@rhel-bpb ~]$ sudo cp -i Downloads/cockroach-v24.2.4.linux-
   arm64/cockroach /usr/local/bin/
   ```

   Then verify the installation if successful:

   ```
   [jsg@rhel-bpb ~]$ cockroach version
   Build Tag:        v24.2.4
   ```

3. By default, CockroachDB looks for external libraries at **`/usr/local/lib/cockroach`**. Create this from the console:

   ```
   [jsg@rhel-bpb ~]$ sudo mkdir -p /usr/local/lib/cockroach
   ```

4. CockroachDB also includes a **`GEOS`** library. **`GEOS`** is a library utilized by CockroachDB for calculations for functions and operators. Make sure the libraries you will use are copied and stored in **`/usr/local/lib/cockroach/`**directory:

   ```
   [jsg@rhel-bpb cockroach-v24.2.4.linux-arm64]$ sudo cp -i lib/
   libgeos.so /usr/local/lib/cockroach/
   [jsg@rhel-bpb cockroach-v24.2.4.linux-arm64]$ sudo cp -i lib/
   libgeos_c.so /usr/local/lib/cockroach/
   ```

5. To ensure that CockroachDB runs properly and references to the external library, **`/usr/local/lib/cockroach`**, verify this from the console:

   ```
   [jsg@rhel-bpb ~]$ which cockroach
   /usr/local/bin/cockroach
   ```

6. From the console, create a temporary in-memory cluster with the **`cockroach demo`** command:

   ```
   [jsg@rhel-bpb ~]$ cockroach demo
   ```

   This will create a temporary single node in-memory CockroachDB cluster. From the interactive SQL shell, run the following command to test that the libraries are available:

   ```
   demo@127.0.0.1:26257/movr> SELECT ST_IsValid(ST_MakePoint(1,2));
   ```

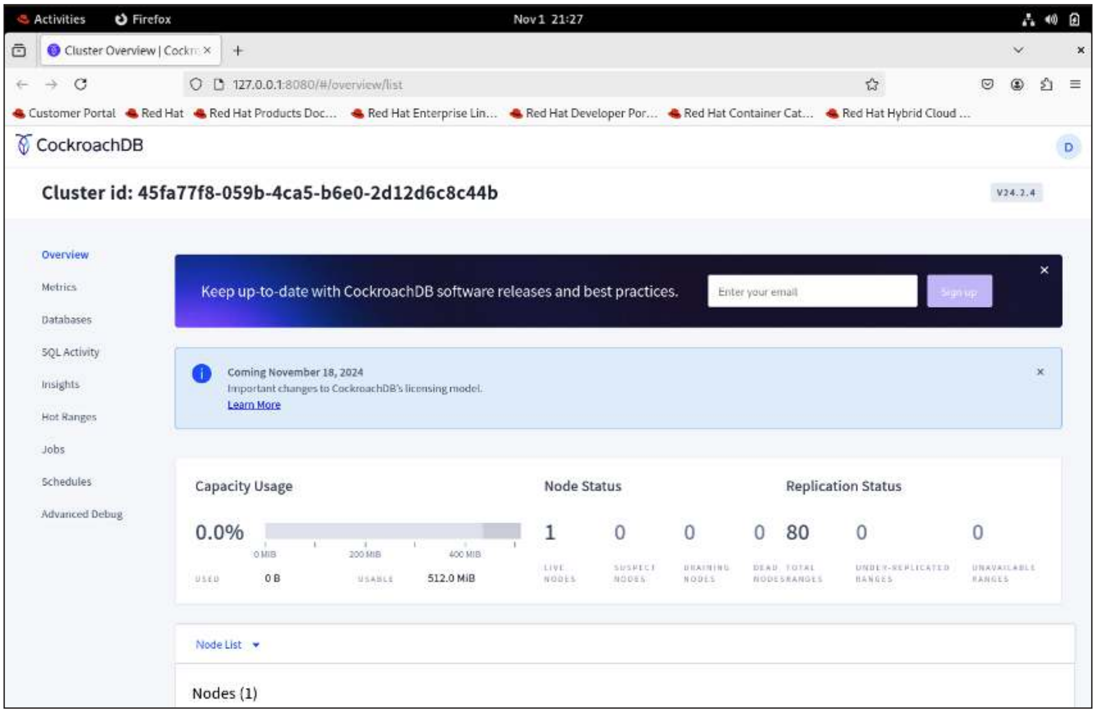
The following output should appear:

```
st_isvalid
--------------
t
```

```
row)
Time: 3ms total (execution 2ms / network 0ms)
```

If the test command does not provide an output and an error message appears, cockroach is not able to find the libraries in **`/usr/local/lib/cockroach`**.

The demo cluster also provides a graphical way to access the user interface of CockroachDB. The link to the web interface would be accessible once the demo cluster starts and is similar to *Figure 8.1* below:



Cluster id: 45fa77f8-059b-4ca5-b6e0-2d12d6c8c44b

Keep up-to-date with CockroachDB software releases and best practices.

Coming November 18, 2024
Important changes to CockroachDB's licensing model.
Learn More

***Figure 8.1**: The CockroachDB web interface*

Do note that the demo cluster is temporary, and that changes made to the database will not be saved. This demo cluster only gives you a sneak peek of what you can do with CockroachDB. This recipe provides the basic installation and configuration of CockroachDB. For more detailed guides installation, deployment, configuration, and management, refer to the official CockroachDB Documentation portal available at **https://www.cockroachlabs.com/docs/stable/**.

Recipe #42: Install and manage Neo4j

Neo4j is a NoSQL, graph database system. A graph database utilizes node edges and properties and used to show and represent relationships of data, instead of having rows nature, it is highly suitable for use in big data and analytics applications, and is popular with data scientists. Neo4j is licensed as open-source, and built on the java platform. Let us go through the steps in installing and configuring Neo4j in RHEL 9:

1. Neo4j requires the Java 17 runtime. RHEL 9 ships OpenJDK and can be installed using yum from the console:

    ```
    [jsg@rhel9_bpb ~]$ sudo yum install java-17-openjdk
    [jsg@rhel9_bpb ~]$ sudo yum install java-17-openjdk-devel
    ```

    Check if you have the right version of Java installed.

    ```
    [jsg@rhel9_bpb ~]$ javac -version
    javac 17.0.2
    ```

    Add the Neo4j repository that we will use to install the Neo4j package.

    ```
    [jsg@rhel9_bpb ~]$ rpm --import https://debian.neo4j.com/
    neotechnology.gpg.key
    [jsg@rhel9_bpb ~]$ cat <  /etc/yum.repos.d/neo4j.repo
    [neo4j]
    name=Neo4j RPM Repository
    baseurl=https://yum.neo4j.com/stable/5
    enabled=1
    gpgcheck=1
    EOF
    ```

2. After adding the repository, install the Community Edition of Neo4j. You also have the option to install the Enterprise Edition of Neo4j using the **`neo4j-enterprise-5.9.0`** package:

    ```
    [jsg@rhel9_bpb ~]$ sudo yum install neo4j-5.9.0
    ```

    After installing, enable Neo4j to start on system boot:

    ```
    [jsg@rhel9_bpb ~]$ sudo systemctl enable neo4j
    ```

    Neo4j has an extensive operations manual covering installation, deployment, and management of the platform for on-premise and cloud. You can get more details about Neo4j and the operations manual at **https://neo4j.com/docs/operations-manual/current/**.

Recipe #43: Install and manage Cassandra

Apache Cassandra is an open-source NoSQL distributed database designed to handle large amounts of data distributed across servers that can be designed with no single point of failure. Cassandra has capabilities for horizontal scaling, distributed architectures and flexible methods for defining schema. Big tech companies known to use Cassandra include *Facebook*, *Instagram*, and *Netflix*. Let us go through the basics of implementing Cassandra on RHEL 9.

Cassandra requires Java. To install Java, see *Recipe #42, Install and Manage Neo4j*, and do *steps 1* and *2*.

1. Once Java is installed, with a text editor, add the Apache repository of Cassandra in **`/etc/yum.repos.d/cassandra.repo`** as a root user. Add the following entries below and save the file:

```
[cassandra]
name=Apache Cassandra
baseurl=https://redhat.cassandra.apache.org/41x/
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://downloads.apache.org/cassandra/KEYS
```

2. Update your repositories and install **`Cassandra`**:

```
[jsg@rhel9_bpb ~]$ sudo yum update
[jsg@rhel9_bpb ~]$ sudo yum install cassandra
```

3. When installation is completed, start the **`cassandra`** service:

```
[jsg@rhel9_bpb ~]$ sudo service cassandra start
```

4. You can monitor the progress of the startup by doing a **`tail`** command in the **`system.log`** file:

```
[jsg@rhel9_bpb ~]$ sudo tail -f logs/system.log
```

Once completed, you can connect to the new Cassandra database with the **`cqlsh`** command. Cassandra has a comprehensive documentation that covers architecture, data modelling, the **Cassandra Query Language** (**CQL**), best practices for operations, and many more. The documentation portal can be accessed at **https://cassandra.apache.org/doc/4.1/index.html**.

Recipe #44: Install and manage Microsoft SQL Server

Microsoft SQL Server (*SQL Server*) is the premier relational database management system developed by *Microsoft*. Previously, it was only available for the Windows platform but now you can install SQL Server on RHEL 9 and create databases that utilize the features and functionalities that were previously only available for the Windows platform. To install SQL Server, you need to have sudo access to the system where you are installing (you can also do this as the root user). Also, ensure that your system meets the minimum hardware and software requirements for SQL Server.

Let us go through the steps in installing, configuring, and running SQL Server on RHEL 9:

1. **Download, install, and configure SQL Server**

   From the console, let us download the SQL Server Red Hat repository configuration file:

```
jsg@rhel9_bpb ~]$ sudo curl -o /etc/yum.repos.d/mssql-server.repo
https://packages.microsoft.com/config/rhel/8/mssql-server-2022.repo
Now install the SQL Server package:
jsg@rhel9_bpb ~]$ sudo yum install -y mssql-server
```

After the package is installed, run the setup script to configure SQL Server. Provide a strong password for the SQL Server **system administrator** (**SA**) account:

```
jsg@rhel9_bpb ~]$ sudo /opt/mssql/bin/mssql-conf setup
```

Once the process is completed, verify that the SQL Server service is running with **`systemctl`**:

```
jsg@rhel9_bpb ~]$ sudo systemctl status mssql-server
```

To allow remote connections, set up your firewall to allow SQL Server transactions through TCP port 1433. Use the **`firewall-cmd`** command and when completed, reload the firewall:

```
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --zone=public --add-port=1433/
tcp --permanent
[jsg@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

2. **Install and set up the SQL Server command line tools**

   Before creating a database, you need to install the SQL Server tools **`sqlcmd`** and **`bcp`**. To start, import the SQL Server public repository GNU Privacy Guard (GnuPG or GPG) keys:

```
[jsg@rhel9_bpb ~]$ sudo rpm --import https://packages.microsoft.com/
keys/microsoft.asc
```

   Download the Red Hat repository configuration file and install the **`mssql-tools`** and the **`unixODBC`** driver packages:

```
[jsg@rhel9_bpb ~]$ sudo curl -o /etc/yum.repos.d/msprod.
repo https://packages.microsoft.com/config/rhel/9/prod.repo
[jsg@rhel9_bpb ~]$ sudo dnf install -y mssql-tools unixODBC-devel
```

   Edit your **`PATH`** environment variable and apply immediately so that you can access **`sqlcmd`** and **`bcp`** from the console:

```
[jsg@rhel9_bpb ~]$ echo 'export PATH="$PATH:/opt/mssql-tools/
bin"' >> ~/.bash_profile
[jsg@rhel9_bpb ~]$ source ~/.bash_profile
```

3. **Connect to SQL Server and create a database**

   To check that everything is working correctly, let us connect to SQL Server with **`sqlcmd`** with the **`sa`** account and password that set earlier:

```
[jsg@rhel9_bpb ~]$ sqlcmd -S localhost -U sa -P '<Password>'
```

   From the **`sqlcmd`**

```
CREATE DATABASE TestDatabase;
GO
```

To confirm the database was created:

```
SELECT Name from sys.databases
GO
```

This will list all databases in SQL Server and **`TestDatabase`** should be among them.

When you are done with your SQL Server session, exit **`sqlcmd`**:

```
QUIT
```

You have just completed the installation of Microsoft SQL Server on RHEL 9.

Conclusion

We have gone through various infrastructure and database platforms that make your RHEL 9 system one of the most robust and feature complete platforms in the market to build applications and have the capacity to serve thousands, if not millions of users to your solution platforms. The recipes in this chapter are but a few of the hundreds more solutions that you can build on top of RHEL 9. The recipes in this chapter provide the most popular and emerging infrastructure and database platforms in the market to date. RHEL is an evolving operating system platform, and along with it, the solution stacks available online. Use this chapter and its recipes whenever you have a project that would require a quick guide in enabling your applications.

In the next chapter, we will discuss about creating and managing virtual machines in in RHEL 9. This can be useful in expanding the capability and capacity of your RHEL 9 system by running multiple guest systems in a single computer host.

Points to remember

- If any of the commands listed in the recipes do not work as expected, make sure you execute them as a sudo user to gain admin privileges and complete the installation and configuration process.
- When choosing a database platform, look at the requirements of your application and with your developers if you require SQL (like MariaDB) or a NoSQL platform (like MongoDB). A quick tip — SQL databases are great for structured data (like putting them in rows and columns), whereas NoSQL are best suited for structured, semi-structured, and unstructured data in one database.
- Make sure you use the latest version of OpenJDK when installing platforms that require Java. RHEL 9 fully supports OpenJDK as shown in some of the recipes of this chapter.

Chapter 9
Administration of Virtualization Workloads

Introduction

Virtualization is a functionality of a RHEL 9 system that allows a sysad to run multiple instances of operating systems as guests without affecting the original system that acts as host. The guests are called virtual machines or VMs and are independent of the host machine yet utilize the underlying hardware and software resources in a shared manner. Since VMs run indecently, they are managed as if they are a complete system, but the hardware is virtualized and shared across other guests. Users may not be aware at all that they are in VM unless they check the details of the system through various tools and utilities. In short, virtualization allows you to run multiple instances of independent systems from a single host, and this gives sysads and application developers a lot of flexibility in building solutions when hardware resources are in short supply. This chapter is all about virtualization in RHEL 9 and is a great reference to bookmark when you need a complete step by step documentation in enabling virtualized workloads in your system.

Structure

In this chapter we will cover the following topics:

- What is virtualization
- Enabling RHEL 9 for virtualization

- Installing a compatible operating system
- Managing the virtual machine lifecycle

Objectives

In this chapter, you will learn and understand virtualization and what is needed in RHEL 9 to support VM guests. We will start by installing the virtualization hypervisor packages, then set up a few Linux and Windows VMs. We will also learn how to manage these VM guests in RHEL 9.

Recipe #45: Install and manage the virtualization hypervisor packages

Virtualization is a great solution to maximize the capacity and capability of existing hardware and software but it is constrained to the physical and logical units of the hardware and software. So, it is important to check the specifications of our system if we are able to properly run a virtualization solution inside your RHEL 9 system.

Let us go through some of the steps in enabling virtualization in RHEL 9:

1. From the console, let us update the system packages prior to installing the software we require to enable virtualization:

```
[jsg@rhel9_bpb ~]$ sudo dnf update
```

2. Now let us install the necessary host and tools packages for virtualization. For this, we will install the following tools: **`qemu-kvm`**, **`libvirt`**, **`virt-install`**, and **`virt-viewer`**, all of which, are available in RHEL 9:

```
[jsg@rhel9_bpb ~]$ sudo dnf install qemu-kvm libvirt virt-
install virt-viewer
```

3. After installation, start the virtualization services. You can also enable the **`libvirtd`** service to start on boot and check the status if its running:

```
[jsg@rhel9_bpb ~]$ sudo systemctl start libvirtd
[jsg@rhel9_bpb ~]$ sudo systemctl enable libvirtd
[jsg@rhel9_bpb ~]$ sudo systemctl status libvirtd
```

4. To verify if the hypervisor is running use the **`virt-host-validate`** command:

```
[jsg@rhel9_bpb ~]$ sudo virt-host-validate
QEMU: Checking for hardware virtualization: PASS
QEMU: Checking for device /dev/kvm: PASS
QEMU: Checking for device /dev/vhost-net: PASS
QEMU: Checking for device /dev/net/tun:
```

The tests should all result to a **`PASS`**. If any of the tests result to a **`FAIL`** output, go through the instructions provided and test again with the **`virt-host-validate`** command. If a test would result to a **`WARN`** result, consider the recommendations provided to improve the set up and run the command again.

When all tests result to a **`PASS`**, we are ready to build a VM guest!

Recipe #46: Install a Linux guest virtual machine

To create a guest VM, we use the **`virt-install`** command and along with it, set the parameters or settings of the VM which includes the name, the set memory, **operating system** (**OS**) disk, the allocated number of vCPUs and the source of the installation, which is usually an OS image.

The following are the required arguments when installing a new VM using the **`virt-install`** command:

- The VM name (**`--name`**)
- The number of vCPUs for the VM (**`--vcpus`**)
- The set memory (**`--memory`**)
- The size of the OS disk (**`--disk`**)
- The location of the OS image (**`--cdrom`**) or network installation (**`--location`**)

Now let us install a basic RHEL 9 guest from the console.

1. Plan first the details of the RHEL 9 VM. For this recipe, we would use the following specifications:

    - 2048 MiB of RAM
    - 4 vCPUs
    - 80 GiB disk
    - We will use a RHEL 9 iso installer located at **`/home/jsg/Downloads/rhel9.iso`**

    Make sure you have enough free resources in your host RHEL 9 system to accommodate this new VM guest.

2. From the console, let us create a new RHEL 9 VM called **`rhel9-guest01`** with the set specifications using the required arguments of the **`virt-install`** command. We will also use the **`--os-variant`** argument to optimize the VM performance. Usually this is detected but its best practice to declare the specific guest OS option:

```
[jsg@rhel9_bpb ~]$ sudo virt-install \
--name rhel9-guest01 --memory 2048 --vcpus 4 \
--disk size=80 --os-variant rhel9.0 \
--cdrom /home/jsg/Downloads/rhel9.iso
```

3. The VM creation will start and if successful, a **`virt-viewer`** window will open and load the **`rhel9.iso`** OS image. You can then proceed with the graphical install of RHEL 9. You can refer to the steps for RHEL 9 installation from Chapter 2, *Setting up RHEL 9*.

4. Once the installation is completed, access the newly installed RHEL 9 VM from the console with the **`virsh console`** command:

```
[jsg@rhel9_bpb ~]$ sudo virsh console rhel9-guest01
```

Recipe #47: Install a Windows guest virtual machine

Installing a Windows VM on a RHEL 9 virtualization host is similar to the steps we have done in the previous recipes (*#45* and *#46*). If you have not set up your RHEL 9 system to run virtualization workloads, refer back to the steps in *Recipe #45 Install and manage the virtualization hypervisor packages*.

You need to have a Windows ISO image readily available. Red Hat provides a list of certified and compatible Windows desktop and server operating systems as VM guests at **https://access.redhat.com/articles/973163#rhelkvm**. Make sure your Windows ISO image is compatible. When ready, open a console and create a new Windows VM.

1. The **`virt-install`** command will also be used and the syntax will be similar to the ones we have done in *Recipe #46, Install a Linux guest virtual machine*. For our Windows VM, we will install and configure it with 2 vCPUs, 4096 MiB of RAM, and 50 GiB disk:

```
[jsg@rhel9_bpb ~]$ sudo virt-install --name=win-guest02 --vcpus=2
--memory=4096 --cdrom=/home/jsg/Downloads/windows.iso --disk size=50
```

2. Proceed with the installation of the Windows VM. Once completed, we need to do one more step to ensure that the performance of your Windows VM — install the virtIO drivers. virtIO is a virtualization standard for network and disk drivers and is widely used to improve Windows performance in a Linux hypervisor environment. However, Microsoft does not include virtIO on Windows, so it needs to be installed separately after the Windows VM installation. You can get more details and installation instructions about virtIO at **https://github.com/virtio-win/virtio-win-pkg-scripts/blob/master/README.md** and download the latest virtIO drivers available at **https://fedorapeople.org/groups/virt/virtio-win/direct-**

downloads/archive-virtio/. Download the ISO file and from the console, attach it to the virtual machine using the **`virsh attach-disk`** command:

```
[jsg@rhel9_bpb ~]$ sudo virsh attach-disk win-
guest02 --source /home/jsg/Downloads/virtio-win.
iso --target hdc --persistent --subdriver raw
```

3. Once the virtIO disk is attached, start the Windows VM with **`virsh start`**:

```
[jsg@rhel9_bpb ~]$ sudo virsh start win-guest02
Domain 'win-guest02' started
```

   In Windows, the **`virtio-win.iso`** will be mounted as a new disk where the virtIO drivers are located. To update the Windows drivers, open Device Manager and check on devices that require updated drivers. Use the drivers provided in the mounted virtIO disk. Reboot your Windows VM to load the new drivers and your Windows VM should now be ready for use.

Recipe #48: Starting the virtual machine from the console

From the previous recipes, we have installed a RHEL 9 and a Windows VM. But these VMs were just installed and are not running yet. We can do this from the console and straightforward. For this recipe, we assume the VMs are already deployed and configured and just require the following:

- The list of VMs that are not running or active
- The name of the VM to start

1. From the console, we just use the **`virsh start`** tool and define the name of the guest VM to start:

```
[jsg@rhel9_bpb ~]$ sudo virsh start rhel9-guest01
Domain 'rhel9-guest01' started
```

2. By default, VM guests are shut down whenever the host machine restarts or boots up. You can set up the VM guest to start automatically, similar to how services are set up to start on boot. From the console use the **`virsh autostart`** tool and specify the guest to start automatically at boot:

```
[jsg@rhel9_bpb ~]$ sudo virsh autostart rhel9-guest01
Domain 'rhel9-guest01' marked as autostarted
```

   Once completed, the **`rhel9-guest01`** VM will start every time the host server boots up.

Recipe #49: View information of deployed virtual machines

Over time, your number of VMs deployed and running on your RHEL 9 host may grow. To do some inventory management and check the status of active VMs, you can view relevant information from the console. The **virsh** command is also used for this and you can even go deeper in each VM to get more information on its state.

1. To view a list of VMs in your host, open a console and run the **virsh list –all** command. An output on the state of your VMs will be shown in a table:

```
[jsg@rhel9_bpb ~]$ sudo virsh list --all
Id   Name              State
----------------------------------
1         rhel9-guest01       running
2      -  win-guest02         shut off
```

   To get the status and information of a particular VM, do a **virsh dominfo VMname** command. Replace the **VMname** with the name of the VM you wish to check.

2. You can also check the status of network interfaces from **virsh**:

```
[jsg@rhel9_bpb ~]$ sudo virsh net-list --all
Name       State    Autostart   Persistent
---------------------------------------------
default    active   yes         yes
labnet     active   yes         yes
```

   By getting a list of deployed VMs, you would be able to check in detail the status of each and troubleshoot based on the output of the **virsh** commands.

Recipe #50: Shut down and delete the virtual machine from the console

Virtual machines and their images can take a lot of disk space in your system and there would be cases where you had need to free up space to accommodate additional files or create a new VM resources. If you do not have much disk capacity, reviewing your use of VM resources would greatly help in managing the disk space of your virtualization host.

Before freeing up space by deleting VM resources and images, you need to shut down the guest VM and ensure that files stored inside it are transferred to a different disk or VM. Now let's go through some commands that would teach you how to manage, shutdown, and delete virtual machine resources and images from the console.

1. From the console, use the **virsh shutdown** command and use the name of the virtual machine to shut down. Make sure all active sessions and applications in the VM are not running anymore:

```
[jsg@rhel9_bpb ~]$ sudo virsh shutdown demo-guest1
Domain 'rhel9-guest01' is being shutdown
```

2. If a VM is not responding, you can do a force shut down with the **virsh destroy** command from the console. Do note that the **virsh destroy** command only terminates the VM session. Ensure that you only use this command if the VM is unresponsive:

```
[jsg@rhel9_bpb ~]$ sudo virsh destroy demo-guest1
Domain 'rhel9-guest01' destroyed
```

3. If you wish to delete the VM, use the **virsh undefine** command. Be careful when using this command as deleting a virtual machine is irreversible, so make sure you have backed up data from the VM along with files stored in attached storage. The VM also needs to be shut down prior to deletion:

```
[jsg@rhel9_bpb ~]$ sudo virsh undefine rhel9-guest01 --remove-all-
storage
Domain 'rhel9-guest01' has been undefined
```

   From the **virsh undefine** command above, we have also removed all associated storage images from the VM. You can also add a **-graceful** option to the command syntax to clean the cache of the disk image.

Conclusion

Virtualization is a key functionality and solution that SysAds can take advantage in RHEL 9. While it has its defined limits set by the hardware, virtualization can help extend your computing capacity requirements when running solutions that can take advantage of shared hardware resources and yet still have independent control of the underlying operating system and solution deployed. RHEL 9 can be used as a virtualization host for workloads that require minimal hardware requirements and the management lifecycle of the VM can be done with a few key commands from the console.

In the next chapter, we will discuss the foundation concepts of container technologies.

Points to remember

- Make sure to check first if your hardware has enough compute and storage capacity to run VM guests as they take resources especially when running at the same time. Your computer's CPU should also support virtualization and RHEL 9 supports only AMD64, Intel 64, and the IBM Z systems (z13 and later).

- Red Hat provides a comprehensive list of compatible and certified operating systems that can be used as guest OS for your VMs. The list includes current and previous RHEL systems as well as Windows desktop and Windows server systems. You can access the compatibility list for all supported architectures at **https://access.redhat.com/articles/973163# rhelkvm**.
- You can only delete a VM when it is turned off or after a graceful shut down. Make sure that all data inside the VM has been backed up prior to the deletion activity as this is irreversible.

Chapter 10
Create, Manage, and Monitor Containers

Introduction

In *Chapter 9, Administration of Virtualization Workloads*, we discussed about using the virtualization functionality of RHEL 9 to run multiple instances of operating system as virtual machine guests. This functionality allowed for greater flexibility for sysads by creating more systems to run on top of an existing hardware host. Much of workloads in the world today run on virtual machines, either running on-premises or in a public cloud provider like AWS or Azure. In this chapter, we will discuss another method of running workloads through containers in Linux, a type of technology that packages and isolates applications with the entire environment (files, execution runtimes, etc.) necessary to run it — but not to the level of completeness of a VM. Linux containers are much more portable than VMs and this portability make it easy to move the application from one environment to another (for example, from on-premises to a public cloud environment). VMs are limited to the hypervisor and the capacity of the machine (usually a physical host), whereas with containers, you can expect consistency across environments because of their independence from the host machine, sharing only the operating system and standard applications but isolate the key runtime and libraries necessary to run the application in the container.

Structure

In this chapter, we will cover the following topics:

- Building containers on RHEL 9 as a single node environment
- Use Podman and Buildah in creating and managing containers
- Create and manage container images such as the Universal Base Image and the RHEL base image

Objectives

Containers are hot technologies currently being adopted by enterprises and the market has seen a shift of preference from virtual machines to containers to run in-house applications as well as third-party software that support it. As the technology is relatively new, set time to learn and understand from this chapter, the foundation concepts of container technologies, and be able to build a single node container environment where you can start using tools like *Podman* to create and manage container workloads.

Recipe #51: Install the container tools

Before we start going through the process of installing the tools you need build and manage containers, you may have used or heard of a technology called **Docker**. This is a very popular tool to build and manage containers and is still widely used today, although its popularity has been overshadowed by another project for managing containers called **Kubernetes**. Docker's approach to containerization is different compared to that of a native Linux container. As of RHEL 9, Red hat has removed Docker tools and support, so we won't be discussing much on Docker in this chapter (though we will reference to some tools that are optional to manage Docker containers).

From the console, update the repositories and install the **`container-tools`** meta package which contains the latest tools for containerization:

```
[jsg@rhel9_bpb ~]$ sudo dnf install container-tools
```

The **`container-tools`** package we will use in this chapter replaces Docker and have the following advantages:

- **Rootless**: Having the container without root means they are secure and can run without privileges that may be taken advantage of.
- **No daemon running**: If you do not run any containers in the system, the tools won't be running in the background, meaning less resource requirements.

The package has three main software tools for containerization:

- Podman is a container engine designed to create and run **Open Container Initiative** (**OCI** and build an open industry standard for container formats and related runtimes. It can also be used to manage container instances. Many of our exercises in this chapter will use Podman.

- Buildah is an image building tool that is used by Podman to perform the creation. The tool is OCI-compatible, so the output is compatible with Docker and Kubernetes. Both tools are complementary with each other.
- Skopeo is used to manipulate, inspect, sign and transfer container images and image repositories. Like the previous tools mentioned, it is open source and lightweight. You can use Skopeo to remotely inspect images on a registry without downloading the whole container image and works with OCI and Docker images.

Tip: This chapter will focus primarily on using Podman with Buildah. The Red Hat blog published a good introduction and overview of the Podman, Buildah, and Skopeo tools. You can read through it at https://www.redhat.com/en/blog/say-hello-buildah-podman-and-skopeo.

If you wish to manage Docker containers, you can do this with Podman by installing the **`podman-docker`** tool. Usually, this tool is included when you install **`container-tools`** but you can also install this separately in case it is not yet part of that installation. This tool replaces the old **`Docker`** command interface and calls the Docker API with equivalent **`podman`** commands:

```
[jsg@rhel9_bpb ~]$ sudo dnf install podman-docker
```

Recipe #52: Build a rootless container environment

By default, running the container tools we installed as sudo or root user gets you full access to the capacity of your system. However, there may be cases where you do not want to grant root or sudo access to some containers. In RHEL 9, you have the option to create a rootless container which means you can run this as a regular user. By creating a rootless container, you can safely run a container limited to the rules and security measures set in a regular user. Note that to allow non-root users to use container tools, you need to set this up as a root user or with root privileges.

From the console, let us pull the latest container image from the remote Red Hat image registry:

```
[jsg@rhel9_bpb ~]$ podman pull registry.access.redhat.com/ubi9/ubi
```

To view the list of images from **`podman`**, use the images command:

```
[jsg@rhel9_bpb ~]$ podman images
```

Note the image ID provided in the 3rd column as this will be the reference ID used when managing images. Once you have pulled the container image, let us run this from **podman** and set a container name called **bpb_ubi** and show the operating system version:

```
[jsg@rhel9_bpb ~]$ podman run --rm --name=bpb_ubi registry.access.redhat.
com/ubi9/ubi cat /etc/os-release
```

Recipe #53: Manage the container registry

A container registry is a collection or a repository of container images used when we build container-based applications with Podman as well as the container tools we have installed in previous recipes (see *Recipe #51*, *Install the container tools*). As part of your RHEL 9 subscription, Red Hat provides access to the following:

- **registry.access.redhat.com**
- **registry.redhat.io**

To get a list of container images from an identified registry, we can do a **pull** command from the terminal:

```
[jsg@rhel9_bpb ~]$ podman pull registry.access.redhat.com/ubi9/ubi
```

Now let us break down the **podman pull** command to understand further:

- **registry.access.redhat.com**: Pertains to the location of the registry server
- **ubi9**: The namespace
- **ubi**: The name of the container image

You can check all the available images in a particular registry by using the **search** command from Podman and explicitly include the registry server location:

```
[jsg@rhel9-bpb ~]$ podman search registry.access.redhat.com
```

You can also search for a particular container image from all available container registries. In the example below, we are searching for images related to **openjdk-11**:

```
[jsg@rhel9-bpb ~]$ podman search openjdk-11
```

You can edit the container registry as a super user by opening the **registries.conf** file located at **/etc/containers/registries.conf** and change the default system-wide settings.

```
[jsg@rhel9-bpb ~]$ sudo nano /etc/containers/registries.conf
```

Finally, you can display the container registries with the **podman** info command. You can format it as well with the **–f** option and show in a **.json** format:

```
[jsg@rhel9-bpb ~]$ podman info -f json
```

Recipe #54: Manage the containers with Podman

With **podman**, you can create and manage the lifecycle of containers. A *pod* is the smallest compute unit in a container environment. A pod is basically a group that contains one or more containers. It includes what is called an infra container that holds information, also known as, namespaces, and this infra container can communicate to other containers in the pod.

Since we have some understanding of pods and how **podman** can be used to manage them, let us go through the steps in creating, managing, and shutting down pods. Make sure the **container-tool** package is installed prior to doing these steps. If you do not have **container-tool** installed yet, check *Recipe #51, Install the container tools.*

To view the Podman documentation and available commands, just open a terminal and use **help**:

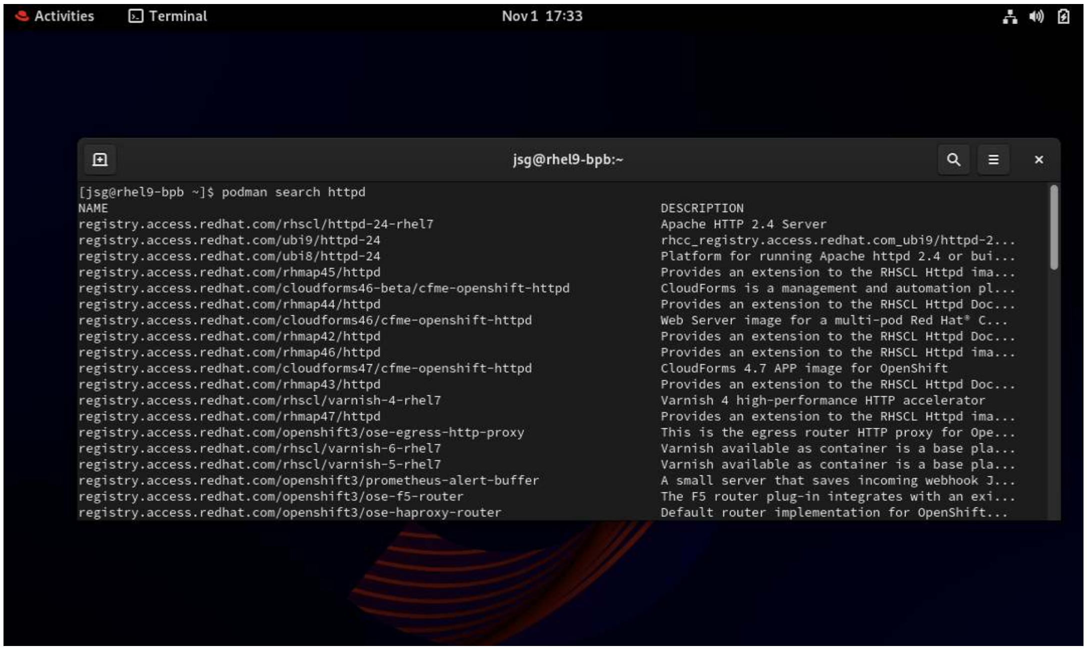
```
[jsg@rhel9-bpb ~]$ podman help
```

To read the full Podman documentation, just do a **man** command:

```
[jsg@rhel9-bpb ~]$ man podman
```

To run a container, we first need to identify an image we would like to use. First do a search for the image you wish to use. In the example below, we are looking for a suitable **httpd** image and the output will show relevant results as shown in *Figure 10.1*:

```
[jsg@rhel9-bpb ~]$ podman search httpd
```



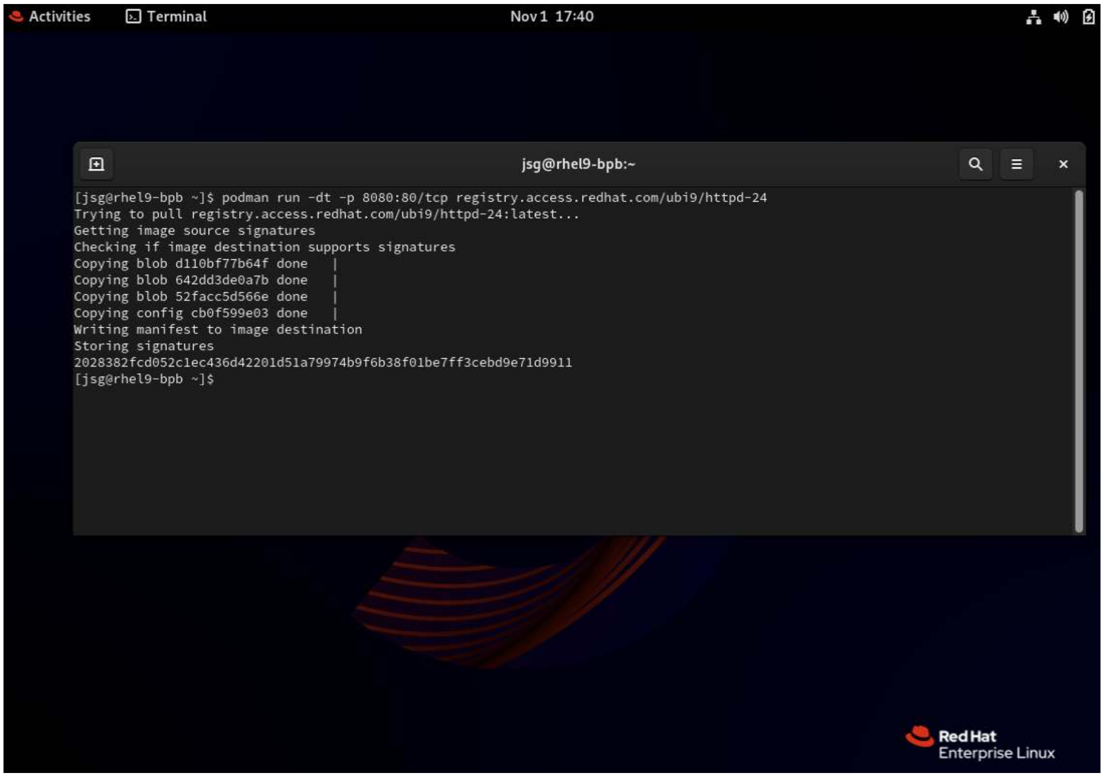
```
[jsg@rhel9-bpb ~]$ podman search httpd
NAME                                                                         DESCRIPTION
registry.access.redhat.com/rhscl/httpd-24-rhel7                              Apache HTTP 2.4 Server
registry.access.redhat.com/ubi9/httpd-24                                     rhcc_registry.access.redhat.com_ubi9/httpd-2...
registry.access.redhat.com/ubi8/httpd-24                                     Platform for running Apache httpd 2.4 or bui...
registry.access.redhat.com/rhmap45/httpd                                     Provides an extension to the RHSCL Httpd ima...
registry.access.redhat.com/cloudforms46-beta/cfme-openshift-httpd            CloudForms is a management and automation pl...
registry.access.redhat.com/rhmap44/httpd                                     Provides an extension to the RHSCL Httpd Doc...
registry.access.redhat.com/cloudforms46/cfme-openshift-httpd                 Web Server image for a multi-pod Red Hat® C...
registry.access.redhat.com/rhmap42/httpd                                     Provides an extension to the RHSCL Httpd Doc...
registry.access.redhat.com/rhmap46/httpd                                     Provides an extension to the RHSCL Httpd ima...
registry.access.redhat.com/cloudforms47/cfme-openshift-httpd                 CloudForms 4.7 APP image for OpenShift
registry.access.redhat.com/rhmap43/httpd                                     Provides an extension to the RHSCL Httpd Doc...
registry.access.redhat.com/rhscl/varnish-4-rhel7                             Varnish 4 high-performance HTTP accelerator
registry.access.redhat.com/rhmap47/httpd                                     Provides an extension to the RHSCL Httpd ima...
registry.access.redhat.com/openshift3/ose-egress-http-proxy                  This is the egress router HTTP proxy for Ope...
registry.access.redhat.com/rhscl/varnish-6-rhel7                             Varnish available as container is a base pla...
registry.access.redhat.com/rhscl/varnish-5-rhel7                             Varnish available as container is a base pla...
registry.access.redhat.com/openshift3/prometheus-alert-buffer                A small server that saves incoming webhook J...
registry.access.redhat.com/openshift3/ose-f5-router                          The F5 router plug-in integrates with an exi...
registry.access.redhat.com/openshift3/ose-haproxy-router                     Default router implementation for OpenShift...
```

Figure 10.1

Next, let us run from the background, the **`httpd-24`** image from **`registry.access.redhat.com`** in the **`ubi9`** namespace by invoking **`podman run`**:

```
[jsg@rhel9-bpb ~]$ podman run -dt -p 8080:80/tcp registry.access.redhat.
com/ubi9/httpd-24
```

Figure 10.2 shows the output from the **`podman run`** command:



```
[jsg@rhel9-bpb ~]$ podman run -dt -p 8080:80/tcp registry.access.redhat.com/ubi9/httpd-24
Trying to pull registry.access.redhat.com/ubi9/httpd-24:latest...
Getting image source signatures
Checking if image destination supports signatures
Copying blob d110bf77b64f done   |
Copying blob 642dd3de0a7b done   |
Copying blob 52facc5d566e done   |
Copying config cb0f599e03 done   |
Writing manifest to image destination
Storing signatures
2028382fcd052c1ec436d42201d51a79974b9f6b38f01be7ff3cebd9e71d9911
[jsg@rhel9-bpb ~]$
```

***Figure 10.2**: Output from the podman run command*

Now list the containers running with the **`podman ps`** command:

```
[jsg@rhel9-bpb ~]$ podman ps
```

From the example above, you can see containers running coming off from different registries and using different ports. You can start and stop these containers through their Container ID (this will be the alphanumeric ID in the first column) with the **`podman start`** and **`podman stop`** command:

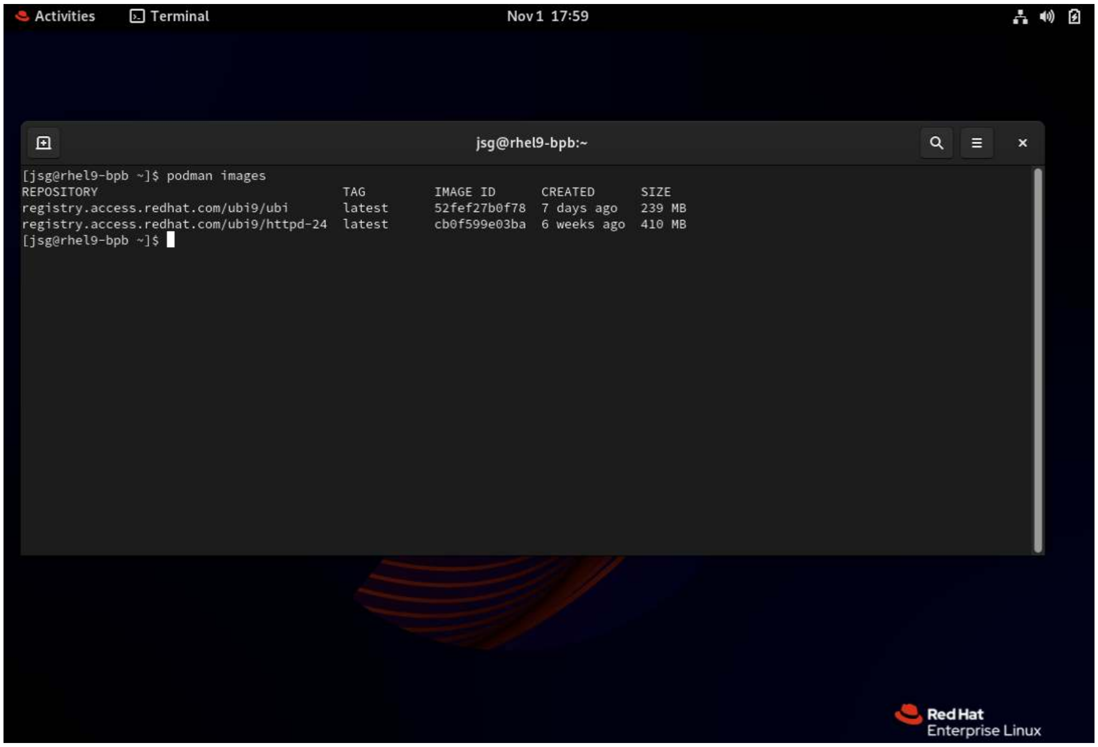
```
[jsg@rhel9-bpb ~]$ podman stop 2028382fcd05
```

Run **`podman ps`** again to check if the container is already stopped. To start it again, just run **`podman start`** command with the Container ID:

```
[jsg@rhel9-bpb ~]$ podman start 2028382fcd05
```

Let us assume that you have stopped all the containers using the **`podman stop`** command. The next thing to do is clean up the images you do not need anymore. First, list down the images using **`podman images`** as shown in *Figure 10.3*:

```
[jsg@rhel9-bpb ~]$ podman images
```



```
[jsg@rhel9-bpb ~]$ podman images
REPOSITORY                            TAG         IMAGE ID      CREATED      SIZE
registry.access.redhat.com/ubi9/ubi       latest      52fef27b0f78  7 days ago   239 MB
registry.access.redhat.com/ubi9/httpd-24  latest      cb0f599e03ba  6 weeks ago  410 MB
[jsg@rhel9-bpb ~]$
```

***Figure 10.3**: Listing of images with podman images command*

Select the image you wish to remove and remember the Image ID. Then from the terminal, run the **`podman rmi`** with the desired Image ID:

```
[jsg@rhel9-bpb ~]$ podman rmi 52fef27b0f78
```

Recipe #55: Monitor the container environment

We can set health check monitors in a container upon deployment by adding some flags. From the example below, we have set a health check monitor using the **`–health-cmd`** flag with **`--health-interval=0`**, meaning we wish to run our health check monitoring manually:

```
[jsg@rhel9_bpb ~]$ podman run -dt --name=mycontainer -p 8282:8282 --health-
cmd='curl http://localhost:8282 || exit 1' --health-interval=0 registry.
access.redhat.com/ubi9/httpd-24
```

```
b6a6a113371140ea2b84de27cb641dca4dd826a58677787c5c44d83d33944bdf
```

Once we have confirmed that this new container is now running using **`podman ps`**, use the **`podman healthcheck run`** command to know the health of the container:

```
[jsg@rhel9_bpb ~]$ podman healthcheck run mycontainer
Healthy
```

Recipe # 56: Manage the container network

In **`podman`**, you can create new rules and manage traffic for both internal and external network access. There are two network types in **`podman`**:

- **Rootless**: The network is automatically set up with no IP address assigned.
- **Rootful**: The container has an IP address assigned

By default, Podman creates a bridged network. From the terminal you can list all the container networks with **`podman network ls`** command:

```
[jsg@rhel9-bpb ~]$ podman network ls
```

You can also inspect the details of the container network and show the range of the IPs, the type of network, and installed plugins by invoking **`podman network inspect`** command and the network name you wish to inspect:

```
[jsg@rhel9-bpb ~]$ podman network inspect podman
```

Let us create a new container network and use a handy container network name that you can remember for later use:

```
[jsg@rhel9-bpb ~]$ podman network create bpbnetwork
```

To check if the **`bpbnetwork`** is created, list all active container networks again with **`podman network ls`** command:

```
[jsg@rhel9-bpb ~]$ podman network ls
```

The next thing to accomplish is connecting a current container to the new container network using the **`podman network connect`** command and include the network name and the container name where the new network will be attached:

```
[jsg@rhel9-bpb ~]$ podman network connect bpbnetwork name_of_container
```

To disconnect, use the same network name and container name.

```
[jsg@rhel9-bpb ~]$ podman network disconnect bpbnetwork name_of_container
```

Finally, remove the unused network with the **`podman network rm`** command and use the network name to identify the network to be removed:

```
[jsg@rhel9-bpb ~]$ podman network rm bpbnetwork
```

Conclusion

Containers are the industry's latest set of platforms and tools used to run high performance, cloud-native applications. RHEL 9 provides a complete set of tools to get started through the **`container-tools`** metapackage which includes essential tools like Podman and Buildah. Much of the work you need to do with containers will revolve around Podman which provides end to end support throughout the lifecycle of containers management and is completely OCI-compliant, making it compatible as well with Docker tools. While Red Hat provides essential tools for containers using Podman, it has an enterprise-ready solution for application development and deployment through its Red Hat OpenShift Container Platform solution. This chapter provided a good starting point in learning and managing basic containers in RHEL 9.

In the next chapter, we will discuss managing networks, files, and storage services in RHEL 9. We will go through the details of setting up and managing these essential services to ensure your RHEL 9 services are robust and secure when you access them.

Points to remember

- The **`container-tools`** metapackage contains all the essential tools necessary in building and managing simple containers through tools like Podman and Buildah. If you have no need for the installed extra tools, you can install individual packages by searching for the software and installing with **`dnf`**.
- Rootless containers are recommended for simple container workloads that do not need to access root-level access of files and libraries. However, before creating rootless containers, you need to create a user that has rights to use the container tools.
- Container network tools allow you open access for your deployed application in the container to other containers or to applications from the internet. Make sure to monitor and manage the created networks and delete unallocated ones when there is no use anymore.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Chapter 11
Working Around Networks, Files, and Storage Services

Introduction

Building on the practical insights of *Chapter 10, Create, Manage, and Monitor Containers,* where we discussed the transformative impact of containerization technology, *Chapter 11, Working around Networks, Files, and Storage Services*, shifts our focus to the foundational network and storage configurations that support robust IT infrastructures. Containers optimize application deployment by abstracting software from the underlying hardware, allowing effortless movement across environments and streamlined development pipelines.

In this chapter, we will explore the physical and logical structures that enable these advanced technologies to function seamlessly. We will explore the configuration and management of network services like **Virtual LANs** (**VLANs**) and **virtual private networks** (**VPNs**), understand network bonding for higher throughput, and set up IP tunnels. Additionally, we will cover fundamental and advanced storage management techniques using tools like **`parted`**, **`fdisk`**, and **Logical Volume Management** (**LVM**). This discussion not only complements the container centric workflows from the previous chapter by providing a secure, efficient, and scalable backend but also sets the stage for complex networking setups essential for today's distributed applications and cloud-based environments.

Structure

In this chapter, we will cover the following topics:

- Configure the ethernet and Wi-Fi connections
- Configure a VLAN to secure network traffic
- Configure network bonding on network interfaces for higher throughput
- Configure a VPN connection
- Set up an IP tunnel
- Do basic disk administration with parted and fdisk
- Create logical storage devices with LVM
- Manage a remote iSCSI storage
- Set up a Samba service for file and print services
- Set up an NFS service

Objectives

In this chapter, we discuss why RHEL 9 is an excellent platform for managing network infrastructure, emphasizing its capabilities in securely configuring and managing network services. We will cover the creation of these network services, focusing on securing them against threats and effectively managing file storage and associated storage devices within the RHEL 9 ecosystem. Additionally, we will emphasize building and managing file storage services tailored for network users, as well as implementing robust measures to secure these storage services and maintain data integrity accessibility.

Recipe #57: Configure the Ethernet and Wi-Fi connection

Before we dive into configuring ethernet and Wi-Fi connections in RHEL 9, you might have used or heard of network configuration tools like **`ifconfig`** or **`ip`** directly. These are low-level tools that have been around for a long time and are still used today, although their popularity has been overshadowed by more user-friendly tools like NetworkManager. The old approach to network configuration often involved editing configuration files in **`/etc/sysconfig/network-scripts/`**, which could be error prone and less intuitive.

As of RHEL 9, Red Hat has made NetworkManager the primary tool for network configuration, providing a more consistent and user friendly approach across different environments. We won't be discussing much about the old methods in this chapter (though we may reference some low-level tools that are optional for advanced troubleshooting).

From the console, you can use the **nmcli** command line tool, which is part of NetworkManager, to configure both ethernet and Wi-Fi connections:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type ethernet con-name "Office-
LAN" ifname eth0 ip4 192.168.1.100/24 gw4 192.168.1.1 [lb@rhel9_bpb ~]$
sudo nmcli device wifi connect "MyHomeWiFi" password "mysecurepassword"
```

NetworkManager, along with its tools like **nmcli** and **nmtui**, were developed to provide a more streamlined network configuration experience and have the following advantages:

- **Cross-platform consistency**: Whether you are on a server, desktop, or even a different Linux distribution, NetworkManager provides a consistent interface.
- **Dynamic configuration**: Easily switch between different networks or connection types without manual reconfiguration.
- **Integration with desktop environments**: For GNOME (GNU Network Object Model Environment) or KDE (K Desktop Environment) users, NetworkManager integrates smoothly with the GUI.
- **Extensibility**: Supports various VPN types, mobile broadband, and more through plugins.

nmcli is the command line interface for NetworkManager, designed to provide full access to NetworkManager capabilities. It is particularly useful in server environments or for scripting. Much of our exercises in this chapter will use **nmcli**.

nmtui is a text-based user interface that provides a more visual, menu-driven approach to network configuration. It is great for those who prefer a more guided experience or when working in environments where a full GUI is not available.

nm-connection-editor is the graphical tool for NetworkManager, typically used in desktop environments. While we focus on command line tools in this chapter, desktop users might find this tool more intuitive.

Tip: This chapter will focus primarily on using nmcli for network configuration. The Red Hat Customer Portal offers comprehensive documentation on NetworkManager in RHEL 9. You can read through it at https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/9/html-single/configuring_and_managing_networking/index.

If you wish to use the old network configuration methods, you can still do this by installing the **network-scripts** package. However, this is not recommended as it is considered a legacy in RHEL 9. The old scripts are kept around mainly for compatibility with older configurations or very specific setups that have not been migrated to NetworkManager yet:

```
[lb@rhel9_bpb ~]$ sudo dnf install network-scripts
```

In the spirit of moving forward with modern, robust tools, we strongly encourage using NetworkManager for all your network configuration needs in RHEL 9.

Recipe #58: Configure a VLAN to secure network traffic

VLANs in RHEL 9 to secure network traffic, you might have used or heard of techniques like physical network segmentation or using separate network switches for different departments. These traditional methods have been around for a long time and are still used today, although their flexibility and cost-effectiveness have been overshadowed by VLAN technology. The old approach often involved purchasing additional hardware, rewiring office spaces, and manually configuring each switch, which could be expensive, time consuming, and less adaptable to changing business needs. As with RHEL 9, Red Hat has seamlessly integrated VLAN configuration with NetworkManager, providing a more dynamic and software defined approach to network segmentation. We won't discuss physical network segmentation much in this chapter (though we may reference some traditional security practices for comprehensive network design). Most enterprise grade switches support VLAN tagging (IEEE 802.1Q), which works seamlessly with the VLAN capabilities of RHEL 9. We can set up a VLAN in RHEL 9 to secure network traffic by using the **`nmcli`** tool provided by NetworkManager. In this example, we will create a VLAN for our finance department to isolate their sensitive traffic from the rest of the network:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type vlan con-name "Finance-
VLAN" ifname vlan10 vlan.parent eth0 vlan.id 10 ip4 192.168.10.50/24 gw4
192.168.10.1
```

```
Connection 'Finance-VLAN' (3ae5f8b0-7e2b-45d5-b5f5-6a9cb5f8e3c0)
successfully added.
```

Let us break down this command:

- **type vlan**: Specifies that we are creating a VLAN connection.
- **con-name "Finance-VLAN"**: A human readable name for our connection.
- **ifname vlan10**: The network interface name for this VLAN.
- **vlan.parent eth0**: The physical interface this VLAN is built on.
- **vlan.id 10**: Our VLAN ID, often coordinated with network admins.
- **ip4 192.168.10.50/24 gw4 192.168.10.1**: Static IP settings for this VLAN.

Once we have confirmed that the VLAN connection is set up correctly, we can activate it:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection up "Finance-VLAN"
```

```
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/3)
```

```
[lb@rhel9_bpb ~]$ ip addr show vlan10
5: vlan10@eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue
state UP group default qlen 1000
link/ether 52:54:00:ee:f5:8b brd ff:ff:ff:ff:ff:ff
inet 192.168.10.50/24 brd 192.168.10.255 scope global vlan10
valid_lft forever preferred_lft forever
```

Next, we will use ping to test connectivity within the VLAN. Let us assume **`192.168.10.10`** is another server in the **`Finance VLAN`**:

```
[lb@rhel9_bpb ~]$ ping -c 4 -I vlan10 192.168.10.10
PING 192.168.10.10 (192.168.10.10) from 192.168.10.50 vlan10: 56(84) bytes
of data.
64 bytes from 192.168.10.10: icmp_seq=1 ttl=64 time=0.352 ms
64 bytes from 192.168.10.10: icmp_seq=2 ttl=64 time=0.426 ms
64 bytes from 192.168.10.10: icmp_seq=3 ttl=64 time=0.373 ms
64 bytes from 192.168.10.10: icmp_seq=4 ttl=64 time=0.389 ms
```

Great! Our RHEL 9 server can communicate with other devices in the **`Finance VLAN`**. Now, let us try to access a server on the general network, say **`192.168.1.100`**, from our **`Finance VLAN`**:

```
[lb@rhel9_bpb ~]$ ping -c 4 -I vlan10 192.168.1.100
PING 192.168.1.100 (192.168.1.100) from 192.168.10.50 vlan10: 56(84) bytes
of data.
^C
--- 192.168.1.100 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3072ms
```

Perfect! The lack of response indicates that our Finance VLAN is properly isolated from the general network. Devices on **`VLAN 10`** cannot directly access resources outside their VLAN, providing a secure enclave for sensitive financial data.

For added security, we can also configure **`firewalld`** to enforce VLAN-specific rules:

```
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --new-zone=finance
success
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --zone=finance --add-
interface=vlan10
success
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --zone=finance --add-
service=mysql --add-service=https
success
[lb@rhel9_bpb ~]$ sudo firewall-cmd --reload
success
```

This setup allows only MySQL and HTTPS traffic in the **`Finance VLAN`**, further securing the environment by limiting network interaction types.

By using RHEL 9's NetworkManager and **`firewalld`**, we have successfully created a secure VLAN for our finance department. This isolates sensitive traffic and demonstrates how software defined networking in RHEL 9 can enhance security without requiring additional physical hardware.

Recipe #59: Configure network bonding on network interfaces for higher throughput

We can set up network bonding in RHEL 9 to aggregate multiple network interfaces into a single logical interface, providing higher throughput and redundancy. In this example, we will bond two Gigabit Ethernet interfaces for increased bandwidth:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type bond con-name bond0 ifname
bond0 mode 802.3ad ip4 192.168.1.100/24 gw4 192.168.1.1
```

```
Connection 'bond0' (8f6e2d1b-3e4a-4c1a-8c9e-6d4f4c0e3b2a) successfully
added.
```

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type ethernet slave-type bond
con-name bond0-port1 ifname eth1 master bond0
```

```
Connection 'bond0-port1' (3ae6f7c0-7e2b-45d5-b5f5-6a9cb5f8e3c1)
successfully added.
```

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type ethernet slave-type bond
con-name bond0-port2 ifname eth2 master bond0
```

```
Connection 'bond0-port2' (5bg8h9d2-9k4c-67e7-d7h7-8b0de7g9f5e2)
successfully added.
```

Let us break down these commands:

Create **`bond0`**:

- **type bond**: Specifies a bonding connection.
- **con-name bond0**: Name for our bond.
- **ifname bond0**: Interface name for the bond.
- **mode 802.3ad**: IEEE 802.3ad dynamic link aggregation (LACP protocol) mode.
- **ip4 192.168.1.100/24**: Static IP for the bond.

Add **`eth1`** to **`bond0`**:

- **type ethernet slave-type bond**: An ethernet interface as a bond slave.
- **con-name bond0-port1**: Name for this slave connection.
- **ifname eth1**: The first interface to add.
- **master bond0**: Associates with our bond.

Similarly, add **eth2** to **bond0**.

Note: The 802.3ad mode requires switch support for Link Aggregation Control Protocol (LACP). Ensure your switch is configured correctly.

Now, let us activate our bonding setup:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection up bond0
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/5)
[lb@rhel9_bpb ~]$ sudo nmcli connection up bond0-port1
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/6)
[lb@rhel9_bpb ~]$ sudo nmcli connection up bond0-port2
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/7)
```

Let us verify our bonding configuration:

```
[lb@rhel9_bpb ~]$ cat /proc/net/bonding/bond0
Ethernet Channel Bonding Driver: v3.7.1
Bonding Mode: IEEE 802.3ad Dynamic link aggregation
Transmit Hash Policy: layer2 (0)
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0
802.3ad info:
LACP rate: slow
Min links: 0
Aggregator selection policy (ad_select): stable
System priority: 65535
System MAC address: 52:54:00:ee:f5:8b
Active Aggregator Info:
Aggregator ID: 1
Number of ports: 2
Actor Key: 17
Partner Key: 1
Partner Mac Address: 00:01:02:03:04:05
Slave Interface: eth1
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
```

```
Permanent HW addr: 52:54:00:aa:bb:cc
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 0
Partner Churned Count: 0
Slave Interface: eth2
MII Status: up
Speed: 1000 Mbps
Duplex: full
Link Failure Count: 0
Permanent HW addr: 52:54:00:dd:ee:ff
Slave queue ID: 0
Aggregator ID: 1
Actor Churn State: none
Partner Churn State: none
Actor Churned Count: 0
Partner Churned Count: 0
```

Great! Our **bond0** interface is properly set up with **eth1** and **eth2** as its slaves, both running at 1000 Mbps full duplex. This configuration theoretically gives us a 2 Gbps link.

Let us test our bonded interface's throughput using **iperf3**. First, we will start an **iperf3** server on another machine (for example: **192.168.1.200**):

```
[user@server ~]$ iperf3 -s
```

Then, on our RHEL 9 machine:

```
[lb@rhel9_bpb ~]$ iperf3 -c 192.168.1.200 -t 30 -P 2 --bind bond0
Connecting to host 192.168.1.200, port 5201
[  5] local 192.168.1.100 port 55862 connected to 192.168.1.200 port 5201
[  7] local 192.168.1.100 port 55864 connected to 192.168.1.200 port 5201
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-30.00  sec  17.8 GBytes  5.11 Gbits/sec
[  7]   0.00-30.00  sec  17.6 GBytes  5.05 Gbits/sec
[SUM]   0.00-30.00  sec  35.4 GBytes  10.2 Gbits/sec
```

Impressive! Our bonded interface is pushing over 10 Gbps. This is because we are using two parallel streams (**-P 2**) to fully utilize our 2 Gbps link. In a real-world scenario, multiple network flows would naturally distribute across both links.

For comparison, let us test a single ethernet interface:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection up "Ethernet 1"
[lb@rhel9_bpb ~]$ iperf3 -c 192.168.1.200 -t 30 --bind eth1
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-30.00  sec  3.44 GBytes  982 Mbits/sec
```

As expected, a single Gigabit Ethernet interface provides around 1 Gbps. Our bonded interface, combining two such links, offers double the throughput.

To ensure our bond also provides redundancy:

```
[lb@rhel9_bpb ~]$ sudo ip link set eth1 down
[lb@rhel9_bpb ~]$ ping -c 4 192.168.1.200
PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data.
64 bytes from 192.168.1.200: icmp_seq=1 ttl=64 time=0.352 ms
64 bytes from 192.168.1.200: icmp_seq=2 ttl=64 time=0.389 ms
64 bytes from 192.168.1.200: icmp_seq=3 ttl=64 time=0.373 ms
64 bytes from 192.168.1.200: icmp_seq=4 ttl=64 time=0.426 ms
```

Perfect! Even with **eth1** down, our server remains connected via **eth2**, demonstrating the high availability provided by network bonding. Using NetworkManager in RHEL 9 to configure network bonding, we have successfully aggregated two Gigabit Ethernet interfaces into a single, more powerful link. This not only doubles our throughput but also provides redundancy, ensuring our server stays connected even if one link fails.

Recipe #60: Configure a VPN connection

We can set up a VPN connection in RHEL 9 using the IPsec protocol provided by *Libreswan*. An IPsec VPN securely connects us to our organization's network resources from a remote location. In this example, we will configure an IPsec VPN connection with IKEv2 certificate based authentication:

- First, install the required packages:

  ```
  lb@rhel9 ~]$ sudo dnf install NetworkManager NetworkManager-
  libreswan-gnome
  ```

  Next, obtain the certificate file (e.g. **company.p12**) from the IT department and import it into the NSS (Network Security Services) database:

  `[lb@rhel9 ~]$ sudo ipsec checknss`

  Initializing NSS database:

  `[lb@rhel9 ~]$ ipsec import company.p12`

  `Enter password for PKCS12 file:`

  Enter the password to import the certificate when prompted.

- Now, open the GNOME control center and navigate to the **Network** settings. Click the + icon, select VPN, and then choose the Identity > IKEv2 (Certificate) option:

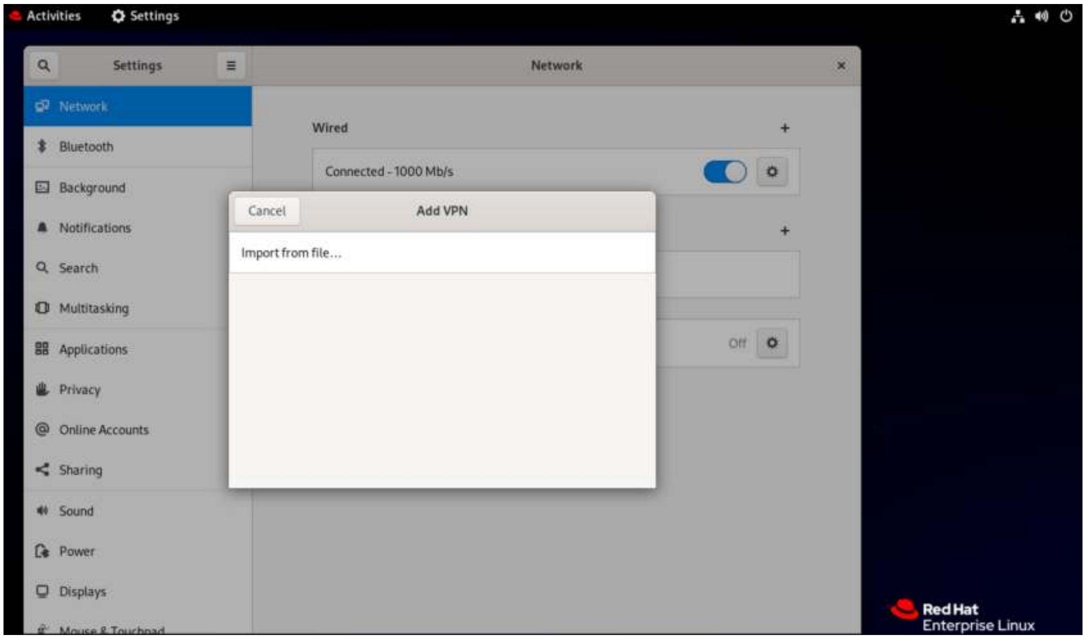


***Figure 11.1**: VPN Configuration*

  Enter the VPN gateway address provided by IT in the Gateway field.

- Under the Authentication section, enter the nickname of the imported certificate.

  Optionally, configure additional advanced settings like phase1/phase2 algorithms, **Perfect Forward Secrecy** (**PFS**), etc. by clicking Advanced.

- Save the VPN connection. It should now appear in the network list.

  To connect, switch on the VPN connection. Verify connectivity by pinging an internal IP or accessing an intranet URL:

```
[lb@rhel9 ~]$ ping 10.8.0.1
[lb@rhel9 ~]$ curl http://intranet.company.com
```

  To restrict routing only company traffic through the VPN, use `nmcli`:

```
Copy code[lb@rhel9 ~]$ sudo nmcli con mod "VPN connection" +vpn.ip4.
routes "192.168.0.0/16 10.0.0.0/8" vpn.ip4.never-default true
```

  This routes only `192.168.0.0/16` and `10.0.0.0/8` (company IP ranges) through the VPN, keeping personal browsing separate.

With RHEL 9's NetworkManager and Libreswan, we have set up a secure IPsec VPN to access internal resources remotely, with additional routing controls for enhanced security and privacy.

Recipe #61: Set up an IP tunnel

We can set up an IP tunnel in RHEL 9 to encapsulate one network protocol inside another to connect two hosts across different networks. In this example, we will create a **Generic Routing Encapsulation** (**GRE**) tunnel to carry IPv6 traffic over an IPv4 network:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection add type ip-tunnel
con-name "IPv6-over-IPv4"
ifname gre1
ip-tunnel.mode gre
ip-tunnel.local 203.0.113.5
ip-tunnel.remote 198.51.100.7
ip6 2001:db8:1234::2/64
Connection 'IPv6-over-IPv4' (7cd9e5f3-6a2b-41c8-a9f6-8e3d4f5c6b7a)
successfully added.
```

Let us break down this command:

- **type ip-tunnel**: Specifies that we are creating a tunneling connection.
- **con-name "IPv6-over-IPv4"**: A descriptive name for our tunnel.
- **ifname gre1**: The interface name for our tunnel.
- **ip-tunnel.mode gre**: We are using GRE mode, which can encapsulate various protocols.
- **ip-tunnel.local 203.0.113.5**: Our server's public IPv4 address.
- **ip-tunnel.remote 198.51.100.7**: The remote server's public IPv4 address.
- **ip6 2001:db8:1234::2/64**: Our IPv6 address inside the tunnel.

Note: We are using reserved IP ranges (`203.0.113.0/24` for server, `198.51.100.0/24` for testing, and `2001:db8::/32` for IPv6 documentation) to avoid conflicts.

Now, let us activate our tunnel:

```
[lb@rhel9_bpb ~]$ sudo nmcli connection up "IPv6-over-IPv4"
Connection successfully activated (D-Bus active path: /org/freedesktop/
NetworkManager/ActiveConnection/11)
```

Let us verify our tunnel setup:

```
[lb@rhel9_bpb ~]$ ip -d link show gre1
12: gre1@NONE: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1476 qdisc noqueue state
UNKNOWN mode DEFAULT group default qlen 1000
link/gre 203.0.113.5 peer 198.51.100.7
inet6 2001:db8:1234::2/64 scope global tentative noprefixroute
valid_lft forever preferred_lft forever
```

Great! Our GRE tunnel is up, using our public IPv4 addresses for encapsulation and providing us with an IPv6 interface. Please note that the lower MTU (**1476** instead of the usual **1500**) is because of the GRE header overhead.

Next, let us make sure our remote server (at **198.51.100.7**) has set up its end of the tunnel correctly. For this example, let us assume it is another RHEL 9 machine with the tunnel configured as:

```
[lb@rhel9_bpb ~]$ nmcli connection add type ip-tunnel \
  con-name "IPv6-over-IPv4" \
  ifname gre1 \
  ip-tunnel.mode gre \
  ip-tunnel.local 198.51.100.7 \
  ip-tunnel.remote 203.0.113.5 \
  ip6 2001:db8:1234::1/64
```

Now, let us try to ping the remote server's IPv6 address through our tunnel:

```
[lb@rhel9_bpb ~]$ ping6 -c 4 2001:db8:1234::1
PING 2001:db8:1234::1(2001:db8:1234::1) 56 data bytes
64 bytes from 2001:db8:1234::1: icmp_seq=1 ttl=64 time=25.3 ms
64 bytes from 2001:db8:1234::1: icmp_seq=2 ttl=64 time=24.9 ms
64 bytes from 2001:db8:1234::1: icmp_seq=3 ttl=64 time=25.6 ms
64 bytes from 2001:db8:1234::1: icmp_seq=4 ttl=64 time=25.1 ms
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 24.932/25.233/25.645/0.283 ms
```

Excellent! We can reach the remote server's IPv6 address through our GRE tunnel. This means our IPv6 traffic is being successfully encapsulated in IPv4 packets, traversing the IPv4 internet, and then decapsulated back into IPv6 at the other end.

To see this in action, let us use **tcpdump** to capture packets on our external interface (assuming it is **eth0**):

```
[lb@rhel9_bpb ~]$ sudo tcpdump -i eth0 -n proto 47
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
15:30:42.123456 IP 203.0.113.5 > 198.51.100.7: GREv0, proto IPv6, length
118
15:30:42.150123 IP 198.51.100.7 > 203.0.113.5: GREv0, proto IPv6, length
118
15:30:43.234567 IP 203.0.113.5 > 198.51.100.7: GREv0, proto IPv6, length
118
```

```
15:30:43.261234 IP 198.51.100.7 > 203.0.113.5: GREv0, proto IPv6, length
118
```

Here, **`proto 47`** filters for GRE packets (**`47`** is the IP protocol number for GRE). As we can see, our IPv6 packets are indeed encapsulated in GRE (IPv4) packets between our two public IPv4 addresses.

This setup allows us to use IPv6 even when our ISP or parts of the internet infrastructure do not support it natively. It is particularly useful in scenarios like:

- **IPv6 migration**: Gradually moving to IPv6 while still relying on IPv4 infrastructure.
- **Cloud connectivity**: Connecting to IPv6 only cloud services over an IPv4 only network.
- **IoT deployments**: Many IoT devices are IPv6 only to conserve IPv4 addresses.

To ensure our tunnel's security, we should also set up encryption. GRE does not provide this natively, so we often pair it with IPsec:

```
[lb@rhel9_bpb ~]$ sudo dnf install libreswan
[lb@rhel9_bpb ~]$ sudo tee /etc/ipsec.d/gre-tunnel.conf << EOF
conn gre-tunnel
left=203.0.113.5
right=198.51.100.7
authby=secret
type=transport
auto=start
phase2=esp
esp=aes256-sha2
EOF
[lb@rhel9_bpb ~]$ sudo tee /etc/ipsec.d/gre-tunnel.secrets << EOF
203.0.113.5 198.51.100.7: PSK "v3ryS3cur3Sh@r3dK3y!"
EOF
[lb@rhel9_bpb ~]$ sudo systemctl enable --now ipsec
```

Now, our GRE tunnel is encrypted with IPsec, providing confidentiality and integrity for our IPv6 traffic as it crosses the IPv4 internet.

By using RHEL 9's NetworkManager and additional tools like **`libreswan`**, we have successfully set up a secure, IPsec encrypted GRE tunnel. This allows us to carry IPv6 traffic over an IPv4 network, showcasing RHEL 9's flexibility in adapting to diverse network environments. Whether you are migrating to new protocols, connecting disparate networks, or just learning about tunneling, RHEL 9 provides the tools to make it happen.

Recipe #62: Do basic disk administration with parted and fdisk

Many system administrators often need to work with storage devices, creating partitions, resizing them, or changing their format. In RHEL 9, we have two powerful tools for this: `parted` for more advanced operations and **`fdisk`** for traditional partitioning. Let us see how to use both:

```
[lb@rhel9_bpb ~]$ lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINTS
sda      8:0    0   40G  0 disk
├─sda1   8:1    0  512M  0 part /boot/efi
├─sda2   8:2    0    1G  0 part /boot
└─sda3   8:3    0 38.5G  0 part
├─rhel-root 253:0    0 34.6G  0 lvm  /
└─rhel-swap 253:1    0  3.9G  0 lvm  [SWAP]
sdb      8:16   0  100G  0 disk
sdc      8:32   0  200G  0 disk
```

As you can see, we have two unpartitioned disks: **`sdb`** (100 GB) and **`sdc`** (200 GB). Let us work on these.

First, let us use **`fdisk`** to create a simple partition layout on **`sdb`**:

```
[lb@rhel9_bpb ~]$ sudo fdisk /dev/sdb
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/sdb: 100 GiB, 107374182400 bytes, 209715200 sectors
...
Command (m for help): n
Partition type
p   primary (0 primary, 0 extended, 4 free)
e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-209715199, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-209715199, default
209715199): +20G

```

```
Command (m for help): n
Select (default p): p
Partition number (2-4, default 2):
First sector (41945088-209715199, default 41945088):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (41945088-209715199, default
209715199):
Created a new partition 2 of type 'Linux' and of size 80 GiB.
Command (m for help): t
Partition number (1,2, default 2): 1
Hex code or alias (type L to list all): 83
Changed type of partition 'Linux' to 'Linux'.
Command (m for help): t
Partition number (1,2, default 2): 2
Hex code or alias (type L to list all): 8e
Changed type of partition 'Linux' to 'Linux LVM'.
Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.
```

Here is what we did with **`fdisk`**:

- Created a 20 GB primary partition (sdb1) for direct use.
- Created an 80 GB primary partition (sdb2) for LVM.
- Set sdb1's type to **`83`** (standard Linux) and sdb2's to **`8e`** (Linux LVM).

Now, let us use parted for more advanced operations on **`sdc`**:

```
[lb@rhel9_bpb ~]$ sudo parted /dev/sdc
GNU Parted 4.1
Using /dev/sdc
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdc: 214GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
(parted) mklabel gpt
(parted) mkpart primary ext4 0% 40%
```

```
(parted) name 1 data
(parted) mkpart primary 40% 70%
(parted) name 2 backup
(parted) set 2 lvm on
(parted) mkpart primary linux-swap 70% 80%
(parted) name 3 swap-space
(parted) mkpart primary 80% 100%
(parted) name 4 future-use
(parted) set 4 msftres on
(parted) print
Model: ATA VBOX HARDDISK (scsi)
Disk /dev/sdc: 214GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:
Number  Start   End     Size    File system  Name        Flags
1      1049kB  85.6GB  85.6GB  ext4        data
2      85.6GB  150GB   64.3GB               backup     lvm
3      150GB   171GB   21.4GB  linux-swap  swap-space
4      171GB   214GB   42.9GB               future-use msftres
(parted) quit
```

Here is what we did with parted:

- Created a **GUID Partition Table** (**GPT**) partition table (more flexible than **Master Boot Record** (**MBR**) used in **`fdisk`**).
- Created a 40% (85.6 GB) **`ext4`** partition named **`data`**.
- Created a 30% (64.3 GB) partition named **`backup`**, set for LVM use.
- Created a 10% (21.4 GB) Linux swap partition.
- Created a 20% (42.9 GB) reserved partition, marked as *Microsoft* reserved (for future Windows dual boot).

Now, let us format and mount our partitions:

- Format and mount the data partition:

```
    sudo mkfs.ext4 /dev/sdc1
    sudo mkdir /mnt/data
    sudo mount /dev/sdc1 /mnt/data
    sudo chown lb:lb /mnt/data
```

- Set up swap space:

```
sudo mkswap /dev/sdc3
sudo swapon /dev/sdc3
```

To make these changes persistent across reboots, add entries to **`/etc/fstab`**:

```
/dev/sdc1 /mnt/data ext4 defaults 0 2
/dev/sdc3 none swap sw 0 0
```

In this recipe, we have used both **`fdisk`** and **`parted`** to partition our disks, showcasing their different strengths:

- **`fdisk`** is the traditional tool that uses the older MBR partitioning scheme. It is simple and widely supported but limited to 2 TB disks and four primary partitions. We used it on **`sdb`** to create a basic layout with standard Linux and LVM partitions.
- **`parted`** is more modern, supporting GPT. GPT allows for disks larger than 2 TB, more than four primary partitions, and partition naming. We used it on **`sdc`** to create a more complex layout, demonstrating features like percentage based sizing and partition flags.

The choice between **`fdisk`** and **`parted`** often comes down to disk size, partition scheme preference, and required features. In modern systems, especially with large disks, **`parted`** with GPT is generally the better choice.

Our partition layout on **`sdc`** shows thoughtful planning:

- A large **`ext4`** partition for immediate data needs.
- An LVM partition for backups, allowing easy size adjustments.
- Dedicated swap space for better system performance.
- A reserved partition, showing foresight for potential future needs like dual booting.

Using LVM on **`sdb2`** and **`sdc2`** provides flexibility. You can easily expand these volumes across multiple disks or resize them without repartitioning. This is invaluable for growing storage needs or shifting allocations between services.

We also demonstrated different filesystem choices:

- ext4 on sdc1: A stable, well-proven filesystem for general use.
- XFS on the LVM volume: Known for good performance with large files, it is a solid backup and server choice.
- We can specify the file system noatime mount option to reduce write operations if we do not need the access time.

Lastly, we configured swap space. While RHEL 9 has better memory management, swapping is still important, especially for memory-intensive tasks or as a safeguard against out-of-memory situations.

This recipe guides disk administration in RHEL 9, covering both traditional and modern tools. System operators can efficiently manage storage resources by understanding **fdisk** and **parted**, along with concepts like LVM and various filesystems, ensuring reliability, performance, and adaptability in their RHEL 9 environments.

Recipe #63: Create logical storage devices with LVM

As system administrators, we often face storage challenges: a database that's outgrowing its disk, a need to add redundancy without downtime, or a desire to snapshot volumes before updates. In RHEL 9, LVM is our *Swiss Army Knife* for these scenarios, offering flexibility, efficiency, and robustness in storage management.

```
[lb@rhel9_bpb ~]$ sudo dnf install lvm2
```

```
Package lvm2-2.03.16-1.el9.x86_64 is already installed.
```

Great! LVM tools are pre-installed in RHEL 9, showcasing their importance in modern storage management. Let's use our unallocated space from **sdb** and **sdc** (*Recipe #62*) to create a robust LVM setup. From *Recipe #62*, we have **/dev/sdb2** (80 GB) and **/dev/sdc2** (64.3 GB) for LVM.

- Create PVs:

```
[lb@rhel9_bpb ~]$ sudo pvcreate /dev/sdb2 /dev/sdc2
```

- Verify PVs:

```
[lb@rhel9_bpb ~]$ sudo pvs
PV         VG        Fmt  Attr PSize   PFree
/dev/sdb2            lvm2 ---   80.00g 80.00g
/dev/sdc2 backup-vg lvm2 a--  <64.30g     0
```

- Create VG:

```
[lb@rhel9_bpb ~]$ sudo vgcreate data-vg /dev/sdb2
sudo vgs
VG        #PV #LV #SN Attr   VSize   VFree
backup-vg   1   1   0 wz--n- <64.30g      0
data-vg     1   0   0 wz--n-  80.00g 80.00g
```

- Create LVs: **20G** for **/var**, **40G** for **/home**, rest for future:

```
[lb@rhel9_bpb ~]$ sudo lvcreate -n var-lv -L 20G data-vg
[lb@rhel9_bpb ~]$ sudo lvcreate -n home-lv -L 40G data-vg
```

- Stripe **home-lv** across both PVs for better performance:

```
[lb@rhel9_bpb ~]$ sudo vgextend data-vg /dev/sdc2
```

```
[lb@rhel9_bpb ~]$ sudo lvconvert --type striped -i 2 /dev/data-vg/
home-lv
```

```
[lb@rhel9_bpb ~]$ sudo lvs
```

```
LV      VG        Attr       LSize   Pool Origin Data%  Meta%  Move
Log Cpy%Sync Convert
```

```
var-lv  data-vg  -wi-a----- 20.00g
```

```
home-lv data-vg  -wi-a----- 40.00g
```

- Format and mount LVs:

```
[lb@rhel9_bpb ~]$ sudo mkfs.xfs /dev/data-vg/var-lv
[lb@rhel9_bpb ~]$ sudo mkfs.ext4 /dev/data-vg/home-lv
[lb@rhel9_bpb ~]$ sudo mkdir /mnt/newvar /mnt/newhome
[lb@rhel9_bpb ~]$ sudo mount /dev/data-vg/var-lv /mnt/newvar
[lb@rhel9_bpb ~]$ sudo mount /dev/data-vg/home-lv /mnt/newhome
```

- Move data and update **/etc/fstab**:

```
[lb@rhel9_bpb ~]$ sudo rsync -avHSX /var/ /mnt/newvar/
[lb@rhel9_bpb ~]$ sudo rsync -avHSX /home/ /mnt/newhome/
```

- Verify before switching:

```
diff -r /var /mnt/newvar
```

```
diff -r /home /mnt/newhome
```

- Update **fstab** and remount:

```
/dev/data-vg/var-lv /var xfs defaults 0 0
/dev/data-vg/home-lv /home ext4 defaults 0 0
```

```
[lb@rhel9_bpb ~]$ sudo umount /var /home /mnt/newvar /mnt/newhome
sudo mount -a
```

- LVM snapshots of the **/var** volume:

```
[lb@rhel9_bpb ~]$ sudo lvcreate -s -n var-snapshot -L 5G /dev/data-
vg/var-lv
```

- Recovery from LVM snapshot:

```
[lb@rhel9_bpb ~]$ sudo lvconvert --merge /dev/data-vg/var-snapshot
```

- Remove the LVM snapshot:

```
[lb@rhel9_bpb ~]$ sudo lvremove /dev/data-vg/var-snapshot
```

- Extend a volume: For example a database in **/var** needs more space:

```
[lb@rhel9_bpb ~]$ sudo lvextend -L +10G /dev/data-vg/var-lv
```

```
[lb@rhel9_bpb ~]$ sudo xfs_growfs /var
```

- Extent a volume to use all remaining space:

```
[lb@rhel9_bpb ~]$ sudo lvextend -l +100%FREE /dev/data-vg/var-lv
[lb@rhel9_bpb ~]$ sudo xfs_growfs /var
```

- Monitor LVM:

```
[lb@rhel9_bpb ~]$ sudo vgdisplay
[lb@rhel9_bpb ~]$ sudo lvdisplay
[lb@rhel9_bpb ~]$ sudo pvdisplay
```

- A nicer view

```
[lb@rhel9_bpb ~]$ sudo lvs -o +devices
  LV         VG        Attr       LSize   Pool Origin Data%  Meta%
Move Log Cpy%Sync Convert Devices
  var-lv     data-vg  -wi-ao----  30.00g
/dev/sdb2(0)
  home-lv    data-vg  -wi-ao----  40.00g
/dev/sdb2(1024),/dev/sdc2(0)
  swap-lv    data-vg  -wi-ao----  10.00g
/dev/sdb2(11264)
  root-lv    rhel-vg  -wi-ao----  34.60g
/dev/sda3(0)
  swap-lv    rhel-vg  -wi-ao----   3.90g
/dev/sda3(8847)
  archive-lv backup-vg -wi-ao---- <64.30g
/dev/sdc2(1024)
```

In this recipe, we have harnessed the power of LVM in RHEL 9 to create a flexible, efficient, and robust storage setup. LVM introduces abstraction layers between physical disks and the file systems:

- **PVs**: The raw disks or partitions.
- **VGs**: Pool of PVs, the storage buckets.
- **LVs**: The virtual partitions carved out from VGs.

This abstraction provides numerous benefits:

- **Flexibility**: We easily created **`/var`** and **`/home`** volumes without worrying about physical partition boundaries. When **`/var`** needed more space, we extended it on-the-fly without affecting **`/home`**.
- **Performance**: By stripping **`home-lv`** across both PVs (in **`data-vg`** and **`backup-vg`**), we improve I/O performance. Each read/write operation can utilize both disks simultaneously.

- **Safety**: LVM snapshots are a game changer. Before updating **`/var`**, we took a snapshot. If something went wrong, a quick rollback with **`lvconvert --merge`** would save the day. This feature is invaluable for system updates, application upgrades, or any risky operation.
- **Efficiency**: With thin provisioning (not shown here), you can overprovision space. For example, you could create a 100 GB volume on a 50 GB VG, and it would grow only as data is added. This is great for environments where you want to allocate generously but conserve actual disk space.
- **Migration**: Need to replace an aging disk? Add the new disk to the VG, mirror the data over, and then remove the old disk — all without downtime.

LVM in RHEL 9 is not just a tool; it is a storage philosophy. It encourages thinking about storage in abstract, dynamic terms. Instead of being constrained by physical disks, you manage pools of storage that adapt to your needs. This shift is crucial in modern, fast-paced IT environments where demands change rapidly.

However, LVM is not without considerations:

- **Complexity**: LVM adds layers between disks and data. While powerful, this can make disaster recovery more complex.
- **Boot issues**: If **`/boot`** is on an LVM, some boot loaders might struggle. RHEL 9 handles this well, but it is why we often keep **`/boot`** on a standard partition.
- **Performance overhead**: LVM's flexibility comes with a tiny I/O overhead. In most cases, the benefits far outweigh this, but it is worth noting for I/O intensive workloads.

This recipe showcases how RHEL 9's robust LVM implementation transforms storage management. By abstracting physical constraints, offering features like snapshots and dynamic resizing, and integrating smoothly with the OS, LVM empowers system administrators to create storage solutions that are not just functional but agile, efficient, and resilient. In today's data-driven world, where storage needs are ever growing and ever changing, LVM in RHEL 9 is an indispensable tool.

Recipe #64: Manage a remote iSCSI storage

In today's distributed environments, storage often resides in centralized locations, away from the servers that use it. **Internet Small Computer Systems Interface** (**iSCI**) is a powerful protocol that allows servers to access this remote storage as if it were directly attached. In this recipe, we will configure RHEL 9 to manage remote iSCSI storage, showcasing its robust support for network attached storage solutions.

First, let us install the necessary packages:

```
[lb@rhel9_bpb ~]$ sudo dnf install iscsi-initiator-utils
```

```
Updating Subscription Management repositories.
Last metadata expiration check: 0:15:29 ago on Tue Jun 04 11:30:15 2024.
Dependencies resolved.
Package                Arch   Version        Repository          Size
Installing:
iscsi-initiator-utils x86_64 6.2.1.4-4.el9 rhel-9-appstream  491 k
Transaction Summary
Install  1 Package
Total download size: 491 k
Is this ok [y/N]: y
Complete!
```

This package provides the iSCSI initiator tools, allowing our RHEL 9 system to connect to iSCSI targets.

Now, let us assume we have an iSCSI target server (we will call it storage.example.com) offering several **Logical Unit Numbers** (**LUNs**). First, we will discover available targets:

```
[lb@rhel9_bpb ~]$ sudo iscsiadm --mode discovery --type sendtargets
--portal storage.example.com
192.168.50.100:3260,1 iqn.2024-06.com.example.storage:array1
192.168.50.100:3260,1 iqn.2024-06.com.example.storage:array1:lun1
192.168.50.100:3260,1 iqn.2024-06.com.example.storage:array1:lun2
192.168.50.100:3260,1 iqn.2024-06.com.example.storage:array1:lun3
Great! The server has three LUNs available. Let's authenticate and connect
to them:
[lb@rhel9_bpb ~]$ sudo iscsiadm --mode node --targetname iqn.2024-06.com.
example.storage:array1 --portal storage.example.com --login
Logging in to [iface: default, target: iqn.2024-06.com.example.
storage:array1, portal: storage.example.com,3260]
Login to [iface: default, target: iqn.2024-06.com.example.storage:array1,
portal: storage.example.com,3260] successful.
```

Now, let us see what block devices have appeared:

```
[lb@rhel9_bpb ~]$ sudo lsblk
NAME   MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
...
sdd      8:48   0  500G  0 disk
sde      8:64   0  1.5T  0 disk
sdf      8:80   0    1T  0 disk
```

Great! Our three iSCSI LUNs have appeared as **`sdd`**, **`sde`**, and **`sdf`**. Let us set up each for different purposes:

- LUN1 (**sdd**, 500 G) for database:

```
[lb@rhel9_bpb ~]$ sudo parted /dev/sdd mklabel gpt
[lb@rhel9_bpb ~]$ sudo parted /dev/sdd mkpart primary 0% 100%
[lb@rhel9_bpb ~]$ sudo mkfs.xfs -K /dev/sdd1
[lb@rhel9_bpb ~]$ sudo mkdir -p /mnt/iscsi/database
[lb@rhel9_bpb ~]$ echo "/dev/sdd1 /mnt/iscsi/database xfs _netdev,x-
[lb@rhel9_bpb ~]$ systemd.requires=iscsi.service 0 0" | sudo tee -a
/etc/fstab
[lb@rhel9_bpb ~]$ sudo mount /mnt/iscsi/database
```

  The **_netdev** option tells the system this is a network based filesystem, and **x-systemd.requires=iscsi.service** ensures iSCSI is available before mounting.

- LUN2 (**sde**, 1.5 TB) for LVM based file shares:

```
[lb@rhel9_bpb ~]$ sudo pvcreate /dev/sde
[lb@rhel9_bpb ~]$ sudo vgcreate shares-vg /dev/sde
[lb@rhel9_bpb ~]$ sudo lvcreate -n engineering-lv -L 500G shares-vg
[lb@rhel9_bpb ~]$ sudo mkfs.xfs /dev/shares-vg/engineering-lv
[lb@rhel9_bpb ~]$ sudo mkdir -p /mnt/iscsi/shares/engineering
[lb@rhel9_bpb ~]$ echo "/dev/shares-vg/engineering -lv /mnt/iscsi/
shares/engineering xfs _netdev,x-systemd.requires=iscsi.service 0 0"
| sudo tee -a /etc/fstab
[lb@rhel9_bpb ~]$ sudo mount /mnt/iscsi/shares/engineering
```

Let us check our mounted iSCSI volumes:

```
[lb@rhel9_bpb ~]$ df -h | grep iscsi
/dev/sdd1              500G   33M  500G   1% /mnt/iscsi/database
/dev/mapper/shares--vg-engineering--lv  500G   26M  500G   1% /mnt/iscsi/
shares/engineering
```

Perfect! All our iSCSI volumes are mounted and ready for use. Now, let us set up automatic iSCSI login at boot:

```
[lb@rhel9_bpb ~]$ sudo systemctl enable --now iscsid
[lb@rhel9_bpb ~]$ sudo iscsiadm --mode node --targetname iqn.2024-06.com.
example.storage:array1 --portal storage.example.com --op update -n node.
startup -v automatic
```

This ensures that our iSCSI sessions are re-established after a reboot.

Lastly, let us configure multipath I/O for our database LUN to enhance performance and redundancy, assuming our storage array provides multiple paths:

```
[lb@rhel9_bpb ~]$ sudo dnf install device-mapper-multipath
```

```
[lb@rhel9_bpb ~]$ echo 'WWID_OF_SDD_FROM_MULTIPATH_-L_OUTPUT' | sudo tee -a
/etc/multipath/wwids
```

```
[lb@rhel9_bpb ~]$ sudo systemctl restart multipathd
```

After rebooting or rescanning, **`/dev/mapper/mpatha`** (or similar) will represent our multi-pathed database LUN.

This recipe demonstrates RHEL 9's robust capabilities in managing remote iSCSI storage. Here is what we have accomplished:

- **Seamless iSCSI integration**: Using **`iscsi-initiator-utils`**, we easily discovered, authenticated, and connected to remote iSCSI targets. RHEL 9's support for iSCSI is mature and seamless.
- **Flexible storage allocation**: We treated iSCSI LUNs just like local disks. One became a database volume, another hosted an LVM setup for departmental shares, and the third served as a backup space. This flexibility is a hallmark of RHEL 9's storage management.
- **LVM on iSCSI**: Using LVM on our second LUN showcases RHEL 9's layered storage approach. We can create, resize, or snapshot volumes on network storage without affecting the underlying iSCSI configuration.
- **High availability with multipath I/O**: By setting up multipath for our database LUN, we are leveraging RHEL 9's enterprise-grade features. This provides higher throughput by using multiple paths simultaneously and fault tolerance, as the failure of one path does not disrupt access.
- **Integration with systemd**: Using **`x-systemd.requires=iscsi.service`** in **`/etc/fstab`** ensures that our iSCSI mounts wait for the service to be ready. This systemd integration is part of RHEL 9's cohesive system design.
- **Network optimization**: The performance we're seeing also reflects RHEL 9's network stack optimizations. Features like TCP BBR (Transmission Control Protocol Bottleneck Bandwidth and Round-trip propagation time) congestion control and optimized NIC (Network Interface Card) drivers help maximize iSCSI throughput.

iSCSI traffic is not encrypted by default. In production, you would want to:

- Use a dedicated, firewalled network for iSCSI.
- Configure **Challenge-Handshake Authentication Protocol** (**CHAP**) for authentication.
- Consider IPsec for encrypting iSCSI traffic.

RHEL 9's iSCSI capabilities shine in this recipe. We are not just attaching storage; we are integrating it deeply into RHEL 9's ecosystem. LVM, multipath I/O, systemd integration—

all these components work harmoniously to provide a storage solution that's flexible, high-performance, and enterprise-ready.

In modern data centers, where storage is often centralized and network-attached, RHEL 9's robust iSCSI support is invaluable. It allows servers to access vast, shared storage pools as if they were local, enabling everything from high-performance databases to scalable file shares, all while maintaining the reliability and efficiency that Red Hat is known for.

Recipe # 65: Set up a Samba service for file and print services

Setting up a Samba service on a RHEL 9 servers can vastly improve the interoperability between Linux and Windows machines within a network. Samba allows file and printer sharing across various operating systems, making it an essential service for mixed-environment setups. This recipe guides you through the steps to install, configure, and secure a Samba server for file and print services. Samba is an open-source software suite that provides seamless file and print services to **SMB/CIFS** (**Server Message Block/ Common Internet File System**) clients. It allows a Linux server to function as a file and print server for Windows, Linux, and macOS clients, facilitating data sharing and centralizing document management in a mixed OS environment.

Step 1: Installing Samba on RHEL 9. First, you need to install the Samba packages. You can do this using the **`dnf`** package manager.

```
[lb@rhel9_bpb ~]$ sudo dnf install samba samba-client samba-common
```

After installation, ensure that the Samba services are enabled and started.

```
[lb@rhel9_bpb ~]$ sudo systemctl enable smb nmb
[lb@rhel9_bpb ~]$ sudo systemctl start smb nmb
```

Step 2: Configuring Samba. The main configuration file for Samba is **`/etc/samba/smb.conf`**. You will need to edit this file to set up your shared directories and printers.

Here is an example configuration:

```
[global]
   workgroup = WORKGROUP
   security = user
   passdb backend = tdbsam
   printing = cups
   printcap name = cups
   load printers = yes
   cups options = raw

[homes]
```

```
   comment = Home Directories
   browseable = no
   writable = yes

[printers]
   comment = All Printers
   path = /var/spool/samba
   browseable = no
   printable = yes
   guest ok = no
   read only = yes
   create mask = 0700
```

In this configuration, **`[homes]`** provides a private share for each user, and **`[printers]`** configures Samba to share printers installed on the server.

Step 3: Adding users. Samba uses its own set of passwords separate from the Linux system passwords. You need to add users to the Samba password database to allow them access to the shares:

```
[lb@rhel9_bpb ~]$ sudo smbpasswd -a username
```

Replace **`username`** with the actual user account name. The user must also exist on the Linux system.

Step 4: Managing Samba services. After configuring your file shares and adding users, restart the Samba services to apply the changes:

```
[lb@rhel9_bpb ~]$ sudo systemctl restart smb nmb
```

Step 5: Configuring firewalls. Ensure that your firewall allows Samba traffic. You can configure **`firewalld`** to allow Samba services through the **`firewall`** as follows:

```
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=samba
[lb@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

Step 6: Accessing shares from a client. From a Windows client, you can access the Samba shares by entering **`\\samba_server_ip\`** in the file explorer address bar, where **`samba_server_ip`** is the IP address of your Samba server. You will be prompted to enter the username and password you configured earlier.

Setting up a Samba server on RHEL 9 provides a powerful, flexible solution for file and print services in a mixed OS environment. By following the steps outlined above, you can successfully deploy Samba to meet your network-sharing needs. Whether you are managing a home network or a corporate infrastructure, Samba offers a robust platform for seamless cross-platform integration.

Recipe # 66: Set up an NFS service

Network File System (**NFS**) is a protocol that allows you to share directories and files with other systems over a network. By setting up NFS, you can allow multiple users or systems to access the same files from their own systems as if the files were present locally. These receipts will take you through the steps to install, configure, and secure an NFS server on RHEL 9. NFS is a distributed file system protocol that allows a user on a client computer to access files over a network in a manner similar to how local storage is accessed. NFS has been widely used in enterprise environments, making it an essential service for systems administrators to manage.

Step 1: Installing NFS. Before you start setting up NFS, you need to install the necessary packages on your RHEL 9 system. You can do this using the **`dnf`** package manager.

```
[lb@rhel9_bpb ~]$ sudo dnf install nfs-utils
```

Once installed, enable and start the NFS service and related services:

```
[lb@rhel9_bpb ~]$ sudo systemctl enable nfs-server rpcbind
```

```
[lb@rhel9_bpb ~]$ sudo systemctl start nfs-server rpcbind
```

Step 2: Configuring NFS Exports. You need to specify which directories you want to share with clients in the **`/etc/exports`** file. Here's how you can configure a shared directory.

Create a directory that you wish to share:

```
[lb@rhel9_bpb ~]$ sudo mkdir /sharedfiles
```

Open the **`/etc/exports`** file and add the directory with its access settings:

```
[lb@rhel9_bpb ~]$ sudo nano /etc/exports
```

Add the following line to allow access from a specific subnet:

```
/sharedfiles 192.168.1.0/24(rw,sync,no_root_squash)
```

This configuration allows machines in the **`192.168.1.0/24`** subnet to read and write to the **`/sharedfiles`** directory.

Step 3: Exporting the shares. After configuring the directories, you want to share, export them and restart the NFS service to apply the changes:

```
[lb@rhel9_bpb ~]$ sudo exportfs -ra
```

```
[lb@rhel9_bpb ~]$ sudo systemctl restart nfs-server
```

Step 4: Configuring firewalls. If you are running a firewall, you need to allow NFS through it. Configure **`firewalld`** to allow NFS:

```
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=nfs
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=mountd
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --add-service=rpc-bind
[lb@rhel9_bpb ~]$ sudo firewall-cmd --reload
```

Step 5: Accessing NFS Shares from a client. On the client side, make sure that the NFS utilities are installed:

```
[lb@rhel9_bpb ~]$ sudo dnf install nfs-utils
```

Create a mount point and mount the NFS share:

```
[lb@rhel9_bpb ~]$ sudo mkdir /mnt/nfs
[lb@rhel9_bpb ~]$ sudo mount -t nfs nfs_server_ip:/sharedfiles /mnt/nfs
```

Replace **`nfs_server_ip`** with the IP address of your NFS server.

Step 6: Automating mounts with **`/etc/fstab`**. To ensure the NFS share is mounted on boot, you can add an entry to the **`/etc/fstab`** file on the client:

```
nfs_server_ip:/sharedfiles /mnt/nfs nfs defaults 0 0
```

Setting up an NFS server on RHEL 9 can significantly enhance your network's data accessibility and efficiency. By following these steps, you have configured an NFS server to share data across your network. NFS remains a critical tool for many IT environments, offering simple yet powerful file sharing capabilities suitable for various applications.

Conclusion

This chapter provides an in-depth exploration of network configuration, storage management, and service setups using RHEL 9. It adeptly covers the transition from traditional tools to modern, robust systems like NetworkManager and LVM, illustrating the enhancement of network and storage capabilities. VLANs, VPNs, network bonding, and IP tunnels are discussed with clarity, showcasing their practical implementations for securing and optimizing network traffic. Additionally, the chapter explores advanced storage configurations using tools such as **`parted`** and **`fdisk`** and highlights the flexibility and power of LVM for managing complex storage needs. The setup of network file services like Samba and NFS integrates these systems into a cohesive network environment, enhancing file sharing and storage accessibility across different platforms.

In the next chapter, we will discuss source codes, DevOps pipelines, and application development. This includes key tools and technologies essential for modern application development and DevOps on RHEL, covering the installation and configuration of tools like OpenJDK JRE, .NET, Git, Jenkins, Docker, Ansible, and Kubernetes to streamline development and deployment pipelines.

Points to remember

- **NetworkManager as the tool of choice**: Transition from traditional networking tools to NetworkManager for a more intuitive and error resistant setup in modern network environments.
- **LVM for flexible storage management**: Utilize LVM to create adaptable and efficient storage solutions that can be easily modified to meet changing data demands without downtime.
- **Secure network practices**: Implement VLANs and VPNs to secure network traffic, ensuring sensitive data remains protected within specified network segments.
- **Storage configuration techniques**: Employ **`parted`** for GPT partition schemes on larger disks, and **`fdisk`** for MBR schemes on smaller or legacy systems, understanding their respective benefits and limitations.
- **Integrating NFS**: Set up and configure Samba and NFS to facilitate cross-platform file sharing, crucial for environments with diverse operating systems.
- **Enhancing connectivity and redundancy**: Use network bonding and IP tunnels to increase network throughput and redundancy, ensuring reliable and continuous network service.
- **Practical commands and configurations**: Familiarity with command line tools like **`nmcli`**, and configuration files such as **`/etc/fstab`**, which are essential for managing network settings and automating tasks efficiently.

This chapter equips readers with the knowledge and skills to effectively manage and troubleshoot networks and storage systems in a RHEL 9 environment, promoting a secure, efficient, and highly available network infrastructure.

CHAPTER 12
Source Codes, DevOps Pipelines, and Application Development

Introduction

This chapter teaches you the critical tools and technologies essential for modern application development and DevOps on RHEL. We will guide you through installing and configuring key software components, including OpenJDK JRE, .NET, and Git, which form the backbone of efficient coding and version control practices. Additionally, we will explore essential DevOps tools such as Jenkins, Docker, Ansible, and minikube Kubernetes, enabling you to automate and streamline your development and deployment pipelines. By the end of this chapter, you will have a robust setup for developing, deploying, and managing applications in a RHEL environment, ensuring your workflows are optimized for performance and reliability.

Structure

In this chapter, we will cover the following topics:

- Install OpenJDK JRE
- Install .NET and publish .NET 6.0 applications
- Install and configure a GIT repository
- Install and configure essential tools for DevOps
-

Objectives

This chapter discusses the installation and configuration of essential development and IT tools on RHEL. It starts with setting up OpenJDK **Java Runtime Environment** (**JRE**) and configuring the `JAVA_HOME` environment variable. Next, it covers the installation of the .NET 6.0 SDK, along with the creation, execution, and publishing of .NET applications, as well as various deployment methods, including single-file, framework-dependent, and self-contained deployments. Additionally, topics include installing and managing Git for local and remote version control and configuring Jenkins for CI/CD (Continuous Integration / Continuous Delivery) pipelines. The session will also address containerization using Docker, automation with Ansible, and deploying Kubernetes environments with minikube and kubectl for local and cluster management.

Recipe #67: Install OpenJDK JRE

To install the OpenJDK JRE on Red Hat Enterprise Linux, you have several options depending on your required Java version. The JRE provides the necessary libraries and components to run Java applications, while the **Java Development Kit** (**JDK**) includes the JRE along with development tools such as compilers and debuggers for creating Java applications. There are multiple versions of OpenJDK available for installation. The installation process for several popular versions: OpenJDK 8, OpenJDK 11, and OpenJDK 17. You can choose the one that fits your requirements.

Installing OpenJDK 8 JRE

1. **Install OpenJDK 8 Runtime Environment**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y java-1.8.0-openjdk
```

2. **Optional: Install the headless version if you don't need GUI support**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y java-1.8.0-openjdk-headless
```

Installing OpenJDK 11 JRE

1. **Install OpenJDK 11 Runtime Environment**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y java-11-openjdk
```

2. **Optional: Install the headless version if you do not need GUI support**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y java-11-openjdk-headless
```

Installing OpenJDK 17 JRE

1. **Install OpenJDK 17 Runtime Environment**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y java-17-openjdk
```

2. **Optional: Install the headless version if you do not need GUI support**:

Post-installation steps

1. **Verify the installation**:

   After installing the JRE, you can verify the installation and check the version with the following command:

```
[lb@rhel9_bpb ~]$ java -version
```

   This command should output the version of the JRE you installed, confirming the successful installation.

2. **Set JAVA_HOME environment variable (optional but recommended)**:

   Setting the **`JAVA_HOME`** environment variable is useful for many applications and tools that require a Java runtime. To set this variable, add the following lines to your **`~/.bashrc`** (for a specific user) or **`/etc/profile`** (for system-wide settings):

```
[lb@rhel9_bpb ~]$ export JAVA_HOME=$(dirname $(dirname $(readlink -f
$(which java))))
```

```
[lb@rhel9_bpb ~]$ export PATH=$JAVA_HOME/bin:$PATH
```

   After adding these lines, apply the changes by sourcing the file:

```
[lb@rhel9_bpb ~]$ source ~/.bashrc  # For user-specific changes
```

```
[lb@rhel9_bpb ~]$ source /etc/profile  # For system-wide changes
```

Choose the version of OpenJDK you need (8, 11, 17, or 21) and follow the appropriate installation steps. Verifying the installation and setting the **`JAVA_HOME`** environment variable will ensure your Java setup is complete and ready for use.

Recipe #68: Install .NET and publish .NET 6.0 applications

.NET is a general-purpose development platform that features automatic memory management and modern programming languages, allowing for the efficient creation of high quality applications. It is available on .NET version 6.0 on RHEL and OpenShift Container Platform, which supports microservices-based architectures. Plus, the application can run on both RHEL and Windows Servers.

Installing .NET

1. **Install .NET 6.0 SDK**:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y dotnet-sdk-6.0
```

2. **Verify the installation**:

```
[lb@rhel9_bpb ~]$ dotnet --info
```

Creating an application using .NET 6.0

1. **Create a new console application**:

```
[lb@rhel9_bpb ~]$ dotnet new console --output my-app
```

2. **Run the application**:

```
[lb@rhel9_bpb ~]$ dotnet run --project my-app
```

The output should be **`Hello World!`**.

Publishing applications with .NET 6.0

Publishing methods are as follows:

- **Single-file application**: Deploy as a single executable. This method packages your application and all its dependencies into a single executable file. It simplifies deployment by reducing the number of files you need to distribute and manage. However, it can result in a larger file size because all necessary dependencies are bundled together.
- **Framework-dependent deployment (FDD)**: Uses a shared system-wide version of .NET. In this approach, your application relies on a shared, system-wide version of the .NET runtime. This means your application package does not include the .NET runtime, which makes the application size smaller. The downside is that the target system must install the appropriate version of the .NET runtime.
- **Self-contained deployment (SCD)**: Includes .NET in the deployment. This method packages the .NET runtime with your application, allowing it to run independently of the .NET version installed on the target system. This ensures compatibility and simplifies deployment at the cost of a larger package size since it includes the .NET runtime.

1. **Publish the application**:

```
[lb@rhel9_bpb ~]$ dotnet publish my-app -f net6.0 -c Release
```

2. **Optional: Trim dependencies for RHEL**:

```
[lb@rhel9_bpb ~]$ dotnet publish my-app -f net6.0 -c Release -r
rhel.9-x64 --self-contained false
```

Running .NET applications in containers

To deploy .NET applications in containers on RHEL, follow the steps below. This method uses .NET runtime and container tools like Podman for efficient deployment pipeline development.

1. **Create and publish a new MVC project**:

```
[lb@rhel9_bpb ~]$ dotnet new mvc --output mvc_runtime_example
[lb@rhel9_bpb ~]$ dotnet publish mvc_runtime_example -f net6.0 -c
Release
```

2. **Create Dockerfile**:

```
[lb@rhel9_bpb ~]$ cat > Dockerfile <<EOF
FROM registry.access.redhat.com/ubi8/dotnet-60-runtime
ADD bin/Release/net6.0/publish/ .
CMD ["dotnet", "mvc_runtime_example.dll"]
EOF
```

3. **Build and run the container**:

```
[lb@rhel9_bpb ~]$ podman build -t dotnet-60-runtime-example .
[lb@rhel9_bpb ~]$ podman run -d -p 8080:8080 dotnet-60-runtime-
example
```

4. **Verify the application**:

```
[lb@rhel9_bpb ~]$ xdg-open http://127.0.0.1:8080
```

At the time of writing this book .NET on RHEL does not provide support for the following technologies and APIs:

- Desktop applications (Windows Forms, WPF)
- WCF servers (clients supported)
- .NET remoting
- Windows-specific APIs (**`Microsoft.Win32.Registry`**, **`System.AppDomains`**, **`System.Drawing`**, **`System.Security.Principal.Windows`**)

For further information, refer to the official Red Hat and Microsoft documentation.

Recipe #69: Install and configure a GIT repository

To install and configure a Git repository on a RHEL system, follow these detailed steps. This guide assumes you have administrative privileges on your RHEL system.

Install Git

1. **Install Git**:

   Install Git using the **`yum`** package manager:

```
[lb@rhel9_bpb ~]$ sudo yum install -y git
```

2. **Verify the installation**:

   Check the installed version of Git to ensure it is correctly installed:

```
[lb@rhel9_bpb ~]$ git –version
git version 2.39.3
```

Configure Git

1. **Set up your Git user information**:

   Configure your name and email address. This information will be used in your commits.

```
[lb@rhel9_bpb ~]$ git config --global user.name "Your Name"
[lb@rhel9_bpb ~]$ git config --global user.email your.email@example.
com
```

2. **Verify the configuration**:

   Check your Git configuration to ensure that your user information is correctly set:

```
[lb@rhel9_bpb ~]$ git config –list
```

Initialize a Git Repository

1. **Create a new directory for your Git repository**:

   Create a directory for your project or navigate to an existing project directory:

```
[lb@rhel9_bpb ~]$ mkdir ~/my-git-repo
[lb@rhel9_bpb ~]$ cd ~/my-git-repo
```

2. **Initialize the Git repository**:

   Initialize a new Git repository in the directory:

```
[lb@rhel9_bpb ~]$ git init
```

3. **Verify the repository initialization:**

   Check the contents of the directory to ensure a **`.git`** directory has been created:

```
[lb@rhel9_bpb ~]$ ls -a
```

Create and commit to the repository

1. **Create a new file in your repository**:

   Create a sample file to add to the repository:

```
[lb@rhel9_bpb ~]$ echo "# My Git Repository" > README.md
```

2. **Add the file to the staging area**:

   Use the **`git add`** command to add the file to the staging area:

```
[lb@rhel9_bpb ~]$ git add README.md
```

3. **Commit the file to the repository**:

   Commit the file to the repository with a descriptive commit message:

```
[lb@rhel9_bpb ~]$ git commit -m "Initial commit"
```

4. **Verify the commit**:

   Check the commit history to ensure your file has been committed:

```
[lb@rhel9_bpb ~]$ git log
```

Set up a remote repository

If you want to set up a remote repository, you can use platforms like GitHub, GitLab, or Bitbucket. Below is an example using GitHub.

1. **Create a new repository on GitHub**:

   Go to GitHub and create a new repository. Do not initialize it with a **`README`**, **`.gitignore`**, or **`license`** file.

2. **Add the remote repository**:

   Add the URL of your GitHub repository as a remote repository:

```
[lb@rhel9_bpb ~]$ git remote add origin https://github.com/
yourusername/your-repo.git
```

3. **Push your local repository to the remote repository**:

   Push the commits in your local repository to GitHub:

```
[lb@rhel9_bpb ~]$ git push -u origin master
```

Recipe #70: Install and configure essential tools for DevOps

To install and configure essential DevOps tools on a RHEL 9 system, follow these steps. This guide covers the installation of Git, Jenkins, Docker, Ansible, and Kubernetes (via minikube) along with necessary configurations. It assumes you have administrative privileges on your RHEL system.

Jenkins

1. **Add the Jenkins repository**:

```
[lb@rhel9_bpb ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo
https://pkg.jenkins.io/redhat-stable/jenkins.repo
```

```
sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.key
```

2. **Install Jenkins**:

3. **Start and enable Jenkins service**:

```
[lb@rhel9_bpb ~]$ sudo systemctl start jenkins
[lb@rhel9_bpb ~]$ sudo systemctl enable Jenkins
```

4. **Adjust the firewall**:

```
[lb@rhel9_bpb ~]$ sudo firewall-cmd --permanent --zone=public --add-
port=8080/tcp
[lb@rhel9_bpb ~]$ sudo firewall-cmd –reload
```

5. **Access Jenkins**:

   Open your web browser and go to **http://your_server_ip:8080** to complete the Jenkins setup.

Docker

1. Add the Docker repository:

```
[lb@rhel9_bpb ~]$ sudo yum-config-manager --add-repo https://
download.docker.com/linux/centos/docker-ce.repo
```

2. Install Docker:

```
[lb@rhel9_bpb ~]$ sudo yum install -y docker-ce docker-ce-cli
containerd.io
```

3. Start and enable Docker service:

```
[lb@rhel9_bpb ~]$ sudo systemctl start docker
[lb@rhel9_bpb ~]$ sudo systemctl enable docker
```

4. Add your user to the docker group:

```
[lb@rhel9_bpb ~]$ sudo usermod -aG docker $USER
newgrp docker
```

5. Verify Docker installation:

```
[lb@rhel9_bpb ~]$ docker –version
Docker version 25.0.2, build 29cf629
```

Ansible

1. Install Ansible:

```
[lb@rhel9_bpb ~]$ sudo yum install -y ansible-core
```

2. Verify Ansible installation:

```
[lb@rhel9_bpb ~]$ ansible --version
ansible [core 2.17.1]
```

Kubernetes (minikube)

1. Install Minikube dependencies:

2. Download and install **minikube**:

```
[lb@rhel9_bpb ~]$ curl -LO https://storage.googleapis.com/minikube/
releases/latest/minikube-linux-amd64
```

```
sudo install minikube-linux-amd64 /usr/local/bin/minikube
```

3. Start **minikube**:

```
[lb@rhel9_bpb ~]$ minikube start --driver=none
```

4. Verify **minikube** installation:

```
[lb@rhel9_bpb ~]$ minikube status
```

Kubernetes CLI (kubectl)

1. Download and install kubectl:

```
[lb@rhel9_bpb ~]$ curl -LO “https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl”
```

```
[lb@rhel9_bpb ~]$ sudo install kubectl /usr/local/bin/kubectl
```

2. Verify kubectl installation:

```
[lb@rhel9_bpb ~]$ kubectl version --client
```

Recipe #71: Install and configure Ansible for DevOps pipeline automation

This step-by-step instruction covers the installation of Ansible, basic configuration, and a simple example of how to use Ansible for automation tasks.

Install Ansible

1. **Install Ansible**:

   Install Ansible from the default RHEL repositories:

```
[lb@rhel9_bpb ~]$ sudo dnf install -y ansible-core
```

2. **Verify the installation**:

   Check the installed version of Ansible to ensure it is correctly installed:

```
[lb@rhel9_bpb ~]$ ansible --version
```

Configure Ansible

1. Create a directory for your Ansible configuration:

```
[lb@rhel9_bpb ~]$ mkdir ~/ansible
```

```
[lb@rhel9_bpb ~]$ cd ~/ansible
```

2. Create an inventory file:

   The inventory file lists the hosts that Ansible will manage. Create a file named **`hosts`**:

```
[lb@rhel9_bpb ~]$ vi hosts
```

Add the following content as an example:

```
[webservers]
webserver1 ansible_host=192.168.1.10 ansible_user=ansible
webserver2 ansible_host=192.168.1.11 ansible_user=ansible

[dbservers]
dbserver1 ansible_host=192.168.1.20 ansible_user=devops
```

Save and close the file.

3. Create a basic Ansible configuration file:

   Create a file named **ansible.cfg** in the same directory:

```
[lb@rhel9_bpb ~]$ vi ansible.cfg
```

Add the following content:

```
[defaults]
inventory = ./hosts
remote_user = root
host_key_checking = False
```

Save and close the file.

Create an Ansible Playbook

1. Create a playbook file:

   A playbook defines the tasks to be executed on the managed hosts. Create a file named **site.yml**:

```
[lb@rhel9_bpb ~]$ vi site.yml
```

2. Add tasks to the playbook:

   Add the following content as a **site.yml** playbook that installs Apache on **webservers** hosts:

```
---
- name: Install and configure Apache
  hosts: webservers
  tasks:
    - name: Install Apache
      ansible.builtin.yum:
        name: httpd
        state: present

    - name: Start and enable Apache
      ansible.builtin.systemd:
        name: httpd
        state: started
        enabled: yes

    - name: Copy index.html to webservers
      ansible.builtin.copy:
        src: ./files/index.html
        dest: /var/www/html/index.html
```

Save and close the file.

3. Create the **index.html** file:

   Create a simple HTML file that will be deployed by Ansible:

```
[lb@rhel9_bpb ~]$ vi index.html
```

   Add the following content:

```
<html>
<head>
  <title>Welcome to Ansible Managed Server</title>
</head>
<body>
  <h1>It works!</h1>
</body>
</html>
```

   Save and close the file.

Run the Ansible Playbook

1. Run the playbook:

   Execute the playbook using the following command:

```
[lb@rhel9_bpb ~]$ ansible-playbook site.yml
```

2. Verify the deployment:

   After the playbook runs successfully, verify that Apache is installed and running on the web servers and that the custom **index.html** is deployed. Open a web browser and navigate to the IP addresses of your webservers to see the changes.

Conclusion

This chapter covered installing and configuring several essential tools and technologies for developing, deploying, and managing applications in an RHEL environment. We began with detailed steps for installing OpenJDK JRE versions 8, 11, 17, and 21, including

.NET 6.0 SDK, creating and running .NET applications, and exploring various publishing methods. Next, we covered the installation and configuration of Git, setting up and initializing repositories, committing files, and configuring remote repositories on GitHub. We also discussed the installation and configuration of essential DevOps tools such as Jenkins for CI/CD, Docker for containerization, Ansible for automation, Kubernetes (minikube) for container orchestration, and Kubernetes CLI (kubectl). Finally, we provided a detailed guide on setting up Ansible for DevOps pipeline automation, including creating inventory files, configuration files, and writing playbooks to automate tasks like installing and configuring Apache on web servers.

In the next chapter, we will discuss the administration of clusters and servers, focusing on setting up and managing high-availability environments. Topics include configuring active/passive clusters for Apache and NFS servers, managing cluster resources and nodes, and ensuring resilient service continuity with Pacemaker on RHEL.

Points to remember

- **JRE and JDK**: JRE provides the necessary libraries to run Java applications, while JDK includes tools for developing Java applications.
- **.NET Framework**: .NET is a versatile development platform that supports various deployment methods, including single-file, framework-dependent, and self-contained deployments.
- **Git**: Git is a distributed version control system essential for tracking changes in code and collaborating with other developers. It involves initial setup, repository creation, and configuring remote repositories for collaborative development.
- **Jenkins**: Jenkins is an open-source automation server that facilitates CI/CD. It requires setting up repositories, configuring firewall rules, and accessing the web interface for further configuration.
- **Docker**: Docker allows for creating, deploying, and running applications in containers. Installing Docker involves setting up repositories, configuring user permissions, and verifying the installation.
- **Ansible**: Ansible is used for automation in IT infrastructure. Setting up Ansible involves creating inventory files, configuration files, and playbooks to automate tasks on managed hosts.
- **Kubernetes (minikube)**: minikube provides a local Kubernetes cluster, useful for development and testing. Installation involves setting up dependencies, downloading minikube, and verifying its status.
- **Kubernetes CLI (kubectl)**: kubectl is the command-line tool for interacting with Kubernetes clusters. Installing kubectl ensures you can manage Kubernetes deployments effectively.

CHAPTER 13
Administration of Clusters and Servers

Introduction

This chapter guides setting up a **high availability** (**HA**) environment by creating a HA cluster to ensure service continuity and minimize downtime. We cover the configuration of an active/passive Apache server, a crucial component for maintaining a resilient web service, and the setup of an active/passive NFS server, enabling reliable file sharing across a network, even in the face of server failures. The important part is managing cluster resources and nodes, offering detailed guidance on maintaining and optimizing your HA cluster for robust performance. This chapter equips you with the necessary skills to implement and manage HA solutions on RHEL, ensuring your services remain operational and resilient.

Structure

In this chapter, we will cover the following topics:

- Install and configure Pacemaker
- Create a high availability cluster
- Configure an active/passive Apache server
- Configure an active/passive NFS server
- Manage cluster resources and cluster nodes

Objectives

By the end of this chapter, you will be able to install and configure Pacemaker to ensure HA on an RHEL system, set up HA clusters for Apache and NFS servers, and effectively manage and maintain cluster resources and nodes to guarantee robust performance and service continuity.

Recipe # 72: Install and configure Pacemaker

Prerequisites:

- A single node running RHEL 9 (or multiple nodes for an entire cluster setup)
- A floating IP address on the same network as the node's static IP address
- The node's hostname must be listed in **`/etc/hosts`**

Step 1: Install required packages

1. Install the HA add-on packages, which include **`pcs`**, **`pacemaker`**, and **`fence-agents`**:

   ```
   [lb@rhel9-bpb ~]$ sudo dnf install -y pcs pacemaker fence-agents-all
   ```

2. Start and enable the **`pcsd`** service:

   ```
   [lb@rhel9-bpb ~]$ sudo systemctl start pcsd.service
   [lb@rhel9-bpb ~]$ sudo systemctl enable pcsd.service
   ```

Step 2: Configure firewall

1. If **`firewalld`** is running, open the necessary ports for HA:

   ```
   [lb@rhel9-bpb ~]$ sudo firewall-cmd –permanent –add-service=high-availability
   [lb@rhel9-bpb ~]$ sudo firewall-cmd –reload
   ```

Step 3: Set up authentication for hacluster **user**

1. Set a password for the **`hacluster`** user:

   ```
   [lb@rhel9-bpb ~]$ sudo passwd hacluster
   ```

2. Authenticate the **`hacluster`** user on the node:

   ```
   [lb@rhel9-bpb ~]$ sudo pcs host auth <node_hostname>
   ```

Step 4: Create and start the cluster

1. Set up and start the cluster:

   ```
   [lb@rhel9-bpb ~]$ sudo pcs cluster setup my_cluster –start <node_hostname>
   ```

2. Check the cluster status:

```
[lb@rhel9-bpb ~]$ sudo pcs cluster status
```

Step 5: Disable fencing (not recommended for production)

1. Disable STONITH (fencing) for testing purposes only:

```
[lb@rhel9-bpb ~]$ sudo pcs property set stonith-enabled=false
```

Warning: Disabling fencing is not recommended in production environments.

Step 6: Install and configure Apache HTTP server

1. Install the Apache web server:

```
[lb@rhel9-bpb ~]$ sudo dnf install -y httpd wget
```

2. Open the necessary firewall ports for HTTP:

```
[lb@rhel9-bpb ~]$ sudo firewall-cmd –permanent –add-service=http
[lb@rhel9-bpb ~]$ sudo firewall-cmd –reload
```

3. Create a simple webpage:

```
[lb@rhel9-bpb ~]$ echo "<html><body>My Test Site - $(hostname)</
body></html>" | sudo tee /var/www/html/index.html
```

4. Configure Apache to enable status monitoring:

```
[lb@rhel9-bpb ~]$ cat <<EOF | sudo tee /etc/httpd/conf.d/status.conf
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from 127.0.0.1
    Allow from ::1
</Location>
EOF
```

Step 7: Configure Pacemaker resources

1. Create an IP address resource and an Apache resource, grouping them to ensure they run on the same node:

```
[lb@rhel9-bpb ~]$ sudo pcs resource create ClusterIP ocf:heartbeat:I-
paddr2 ip=<floating_ip> --group apachegroup
sudo pcs resource create WebSite ocf:heartbeat:apache configfile=/etc/
httpd/conf/httpd.conf  statusurl="http://localhost/server-status"  –
group apachegroup
```

2. Check the status of the configured resources:

```
[lb@rhel9-bpb ~]$ sudo pcs status
```

Step 8: Test the setup

1. Access the test site using the floating IP address in a web browser.
2. Simulate an Apache service failure:

```
[lb@rhel9-bpb ~]$ sudo killall -9 httpd
```

3. Verify that Pacemaker restarts the Apache service:

```
[lb@rhel9-bpb ~]$ sudo pcs status
```

4. If necessary, clear the failure status:

```
[lb@rhel9-bpb ~]$ sudo pcs resource cleanup WebSite
```

Step 9: Stop the cluster

1. When testing is complete, stop the cluster services:

```
[lb@rhel9-bpb ~]$ sudo pcs cluster stop –all
```

Notes:

- This setup is for demonstration and testing purposes only. For a production setup, ensure fencing is properly configured and that you use at least two nodes for HA.
- Always refer to the official Red Hat documentation for production configurations and best practices.

Recipe #73: Create a high availability cluster

HA clusters provide reliable and resilient services by eliminating single points of failure. This cookbook article guides you through setting up an active/passive Apache web server in a two-node Red Hat HA cluster using Pacemaker.

Prerequisites:

- Two RHEL 9 servers with network connectivity. We will call them **node1.example.com** and **node2.example.com**.
- Shared storage accessible by both nodes (iSCSI, Fibre Channel, etc.).
- A floating IP address (e.g., 192.168.1.100) on the same network as the nodes' public interfaces.
- A fencing device for each node (e.g., an APC power switch with IP 192.168.1.200).

Preparation:

1. **Install cluster software**: On both nodes, install the necessary packages:
   ```
   dnf install pcs pacemaker fence-agents-all
   ```
2. **Configure firewall**: Allow cluster traffic through the firewall:
   ```
   firewall-cmd --permanent --add-service=high-availability
   firewall-cmd --reload
   ```
3. **Start and enable pcsd**: Start and enable the `pcsd` service on both nodes:
   ```
   systemctl start pcsd.service
   systemctl enable pcsd.service
   ```
4. **Set hacluster password**: Set the same password for user `hacluster` on both nodes:
   ```
   passwd hacluster
   ```

Steps:

1. **Authenticate nodes**: On `node1`, authenticate `hacluster` for both nodes:
   ```
   pcs host auth node1.example.com node2.example.com
   ```
2. **Create cluster**: Create and start the `cluster`:
   ```
   pcs cluster setup my_cluster --start node1.example.com node2.example.
   com
   ```
3. **Enable cluster services**: Enable cluster services to start at boot on both nodes:
   ```
   pcs cluster enable –all
   ```
4. **Configure fencing**: Create a `stonith` resource for the APC power switch:
   ```
   pcs stonith create myapc fence_apc_snmp ipaddr=192.168.1.200 \
       pcmk_host_map="node1.example.com:1;node2.example.com:2" \
       login=apc passwd=apc
   ```

Important: Replace placeholder values with your actual credentials and port mapping.

5. **Prepare shared storage**:
   a) Create a physical volume on the shared storage (e.g., `/dev/sdb1`):
      ```
      pvcreate /dev/sdb1
      ```
   b) Create a volume group named `my_vg`:
      ```
      vgcreate --setautoactivation n my_vg /dev/sdb1
      ```
   c) Create a logical volume named `my_lv`:
      ```
      lvcreate -L 1G -n my_lv my_vg
      ```
   d) Create an XFS filesystem on the logical volume:
      ```
      mkfs.xfs /dev/my_vg/my_lv
      ```

6. **Configure Apache:**

   a) Install Apache on both nodes:

   ```
   dnf install httpd
   ```

   b) Enable the status module in `/etc/httpd/conf.d/status.conf` on both nodes:

   ```
   <Location /server-status>
       SetHandler server-status
       Require local
   </Location>
   ```

   c) Create a simple web page in `/var/www/html/index.html`:

   ```
   <html>
   <body><h1>Welcome to my HA website!</h1></body>
   </html>
   ```

7. **Create cluster resources**: Create resources for LVM, filesystem, IP address, and Apache:

   ```
   pcs resource create my_lvm ocf:heartbeat:LVM-activate vgname=my_vg vg_access_mode=system_id --group apachegroup
   pcs resource create my_fs Filesystem device="/dev/my_vg/my_lv" directory="/var/www" fstype="xfs" --group apachegroup
   pcs resource create VirtualIP ocf:heartbeat:IPaddr2 ip=192.168.1.100 cidr_netmask=24 --group apachegroup
   pcs resource create Website apache configfile="/etc/httpd/conf/httpd.conf" statusurl="http://localhost/server-status" --group apachegroup
   ```

Serving your HA website

Access your website using the floating IP address (e.g., `http://192.168.1.100`). The cluster will ensure that the website stays accessible even if one node fails.

Troubleshooting

- Check cluster status with pcs status.
- Review system logs on both nodes for errors.
- Use `pcs` resource `debug-start <resource_id>` to test individual resource configuration.

You have successfully set up an active/passive Apache web server in a Red Hat HA cluster. This configuration provides a basic level of high availability for your website. You can explore more advanced configurations and options as needed.

Recipe #74: Configure an active/passive Apache server

Tired of website downtime? This recipe guides you through configuring an active/passive Apache cluster on RHEL, ensuring your web services stay up even when one server takes a nap.

Ingredients:

- **Two RHEL servers**: We will call them node1 and node2, prepped with network connectivity and matching RHEL versions.
- **Shared storage**: Choose your favourite: iSCSI, Fibre Channel, or any storage accessible by both nodes.
- **Floating IP address**: One unused IP address within your network.
- **Fencing device**: Something to reliably power cycle a misbehaving node, like an APC power switch.
- **Spices (Software)**:
  - **pcs**: Our cluster command line tool.
  - **pacemaker**: The brain of the cluster, managing resource failover.
  - **fence-agents**: Tools to control your fencing device.
  - **httpd**: The Apache web server itself.

Preparation:

1. **Spice up your servers**: Install the required packages on both nodes:
   ```
   dnf install pcs pacemaker fence-agents-all httpd
   ```
2. **Firewall seasoning**: Open up the ports needed for cluster communication:
   ```
   firewall-cmd --permanent --add-service=high availability
   firewall-cmd --reload
   ```
3. **pcsd prep**: Start and enable the **`pcsd`** daemon on both nodes:
   ```
   systemctl start pcsd.service
   systemctl enable pcsd.service
   ```
4. **Secret ingredient**: Set the same password for the **`hacluster`** user on both servers.

Recipe:

1. **Node authentication**: On **`node1`**, tell **`pcs`** to trust both nodes using the **`hacluster`** account:
   ```
   pcs host auth node1 node2
   ```

2. **Cluster creation**: Let us bring our nodes together:

   ```
   pcs cluster setup my_cluster --start node1 node2
   ```

3. **Cluster startup**: Enable cluster services on both nodes to start automatically on boot:

   ```
   pcs cluster enable --all
   ```

4. **Fencing setup**: Time to configure our safety net. Replace the placeholders with your fencing device details:

   ```
   pcs stonith create myfence fence_apc_snmp ipaddr=FENCING_IP \
   pcmk_host_map="node1:1;node2:2" \
   login=FENCING_USERNAME passwd=FENCING_PASSWORD
   ```

5. **Shared storage setup**: This depends on your chosen storage. Here is a simple LVM setup using `/dev/sdb1` as an example:

   a) **Create a physical volume**: `pvcreate /dev/sdb1`

   b) **Create a volume group**: `vgcreate --setautoactivation n vg_web /dev/sdb1`

   c) **Create a logical volume**: `lvcreate -L 1G -n lv_web vg_web`

   d) **Format with a filesystem (XFS in this case)**: `mkfs.xfs /dev/vg_web/lv_web`

6. **Apache configuration**:

   a) **Status check**: Enable the Apache status module by uncommenting/adding these lines in `/etc/httpd/conf.d/status.conf` on both nodes:

   ```
   <Location /server-status>
       SetHandler server-status
       Require local
   </Location>
   ```

   b) **Welcome page**: Create a basic `index.html` in `/var/www/html` on one node.

7. **Cluster resources**: Time to tell Pacemaker what to manage:

   ```
   pcs resource create web_lvm ocf:heartbeat:LVM-activate vgname=vg_web vg_access_mode=system_id --group webserver

   pcs resource create web_fs Filesystem device="/dev/vg_web/lv_web" directory="/var/www" fstype="xfs" --group webserver

   pcs resource create web_ip ocf:heartbeat:IPaddr2 ip=FLOATING_IP cidr_netmask=24 --group webserver

   pcs resource create web_apache apache configfile="/etc/httpd/conf/httpd.conf" statusurl="http://localhost/server-status" --group webserver
   ```

Tasting the results:

Access your website using the floating IP. Try stopping Apache on the active node (**`systemctl stop httpd`**). You should see a brief interruption, and then the other node will serve the site. The **`pcs status`** is your friend to see where resources are running and for any errors. Test your fence device using **`pcs stonith fence node1`** to power cycle **`node1`** to ensure it works.

Your web server is more resilient than ever. By clustering Apache in an active/passive setup, you have minimized downtime and ensured your website keeps serving content, even with server hiccups. Happy clustering!

Recipe #75: Configure an active/passive NFS server

You need a reliable way to share files across your network, even in the face of server failures? This recipe guides you through setting up an active/passive NFS cluster on RHEL using Pacemaker. Your files are always available, even if one server takes a break.

- **Two RHEL servers**: Our trusty duo, **`node1`**, and **`node2`**, prepped with network connectivity and matching RHEL versions.
- **Shared storage**: The central repository for your shared files, accessible by both nodes. iSCSI, Fibre Channel, or similar will do.
- **Floating IP address**: A dedicated IP for clients to access the NFS share.
- **Fencing device**: A reliable way to isolate a problematic node, like an APC power switch.
- **Software**:
    - **pcs**: Our cluster command line maestro.
    - **pacemaker**: The cluster's conductor, orchestrating resource failover.
    - **fence-agents**: Tools to interact with your fencing device.
    - **nfs-utils**: The NFS server components.
    - **lvm2-lockd**: For managing LVM in a cluster environment (if using LVM).

Preparation:

1. Install the required packages on both nodes:
```
dnf install pcs pacemaker fence-agents-all nfs-utils lvm2-lockd
```
2. Open the necessary ports for cluster and NFS traffic:

```
firewall-cmd --permanent --add-service=high availability
firewall-cmd --permanent --add-service=nfs
firewall-cmd --reload
```

3. Start and enable the `pcsd` daemon on both nodes:
   ```
   systemctl start pcsd.service
   systemctl enable pcsd.service
   ```
4. Set a common password for the `hacluster` user on both servers.

Recipe:

1. **Node authentication**: On `node1`, authorize `pcs` to manage both nodes with the hacluster account:
   ```
   pcs host auth node1 node2
   ```
2. **Cluster assembly**: Combine your nodes into a delicious cluster:
   ```
   pcs cluster setup my_cluster --start node1 node2
   ```
3. **Cluster startup**: Enable cluster services on both nodes for automatic startup:
   ```
   pcs cluster enable --all
   ```
4. **Fencing setup**: Let us ensure node misbehaviour is handled swiftly. Replace placeholders with your device details:
   ```
   pcs stonith create myfence fence_apc_snmp ipaddr=FENCING_IP \
   pcmk_host_map="node1:1;node2:2" \
   login=FENCING_USERNAME passwd=FENCING_PASSWORD
   ```
5. **Shared storage prep**: Configure your shared storage. Here is a basic LVM setup using `/dev/sdb1` as an example:
   - **Physical volume**: `pvcreate /dev/sdb1`
   - **Volume group**: `vgcreate --shared vg_nfs /dev/sdb1` (use `--shared` for cluster LVM)
   - **Logical volume**: `lvcreate -L 10G -n lv_nfs vg_nfs`
   - **Format**: `mkfs xfs /dev/vg_nfs/lv_nfs`
6. **NFS configuration**:
   - **Exports**: On one node, create the NFS export directory (e.g., `/nfsshare`) and mount the LVM volume there.
   - **/etc/exports setup**: Add the following line to `/etc/exports` on both nodes, replacing placeholders:
     ```
     /nfsshare *(rw,sync,no_root_squash,no_subtree_check)
     ```

7. **Cluster resource creation**: Time to let Pacemaker handle the magic:

   ```
   pcs resource create nfs_lvm ocf:heartbeat:LVM-activate vgname=vg_nfs
   vg_access_mode=lvmlockd --group nfs_server
   ```

   ```
   pcs resource create nfs_fs Filesystem device="/dev/vg_nfs/lv_nfs" di-
   rectory="/nfsshare" fstype="xfs" --group nfs_server
   ```

   ```
   pcs resource create nfs_ip ocf:heartbeat:IPaddr2 ip=FLOATING_IP cidr_
   netmask=24 --group nfs_server
   ```

   ```
   pcs resource create nfs_daemon nfsserver nfs_shared_infodir=/nfsshare/
   nfsinfo nfs_no_notify=true --group nfs_server
   ```

Enjoying your fail-safe NFS:

Mount the NFS share from a client machine using the floating IP: `mount -t nfs FLOAT-ING_IP:/nfsshare /mnt/nfs`. Now, try gracefully stopping the NFS service on the active node (`systemctl stop nfs-server`). After a brief pause, the share will be served from the other node!

- `pcs status` shows resource locations and health.
- Do not forget to test your fencing setup with `pcs stonith fence node1`.
- Fine-tune resource behavior with colocation constraints, resource stickiness, and more.
- For production, dedicated network interfaces for cluster and NFS traffic should be considered.

You have built a robust and reliable NFS server that can gracefully handle server failures. Your files are now accessible even when one node is down, ensuring smooth operations and happy users. Happy sharing!

Recipe #76: Manage cluster resources and cluster nodes

Managing a high availability cluster is important but complex to ensure that it does not stop ensuring smooth operation and graceful handling of maintenance tasks. You are using the `pcs` tool as a trusty cluster command line companion.

Resource relocation

Move a resource to its preferred node based on current cluster conditions. The `resource_name` is the name of the resource you want to relocate.

1. **Relocate the resource**: This command calculates the optimal node and moves the resource:

   ```
   pcs resource relocate run resource_name
   ```

2. **Optional**: **clear relocate constraints**: After relocation, you can remove the temporary constraints created by the command:

   ```
   pcs resource relocate clear resource_name
   ```

Live migration

Move a resource to a different node without interrupting service (if the resource agent supports it). The following command uses the `resource_name` for the resource to migrate and `target_node` for the destination node.

1. **Check for live migration support**: Ensure the resource agent supports live migration. Consult its documentation.
2. **Migrate the resource**: Use the `move` command with the `--wait` flag:

   ```
   pcs resource move resource_name target_node --wait
   ```

3. **Resource disable/enable**:

   Temporarily stop and prevent a resource from running. The `resource_name` specify the resource to manage.

   a) **Disable the resource**:

      ```
      pcs resource disable resource_name
      ```

   b) **Re-enable the resource**:

      ```
      pcs resource enable resource_name
      ```

Resource ban

Prevent a resource from running on a specific node. The resource to restrict is `resource_name`, and `node_name` is the node from which to ban the resource.

1. **Ban the resource**:

   ```
   pcs resource ban resource_name node_name
   ```

2. **Optional: remove the ban**:

   ```
   pcs resource clear resource_name node_name
   ```

Node standby mode

Temporarily remove a node from service, migrating its resources elsewhere. The `node_name` specifies the node to put in standby mode.

1. **Enter standby mode**:

   ```
   pcs node standby node_name
   ```

2. **Return to active mode**:

   ```
   pcs node unstandby node_name
   ```

Node addition

Expand your cluster by adding a new node. The hostname of the new node is **`new_node_name`**.

1. **Prepare the new node**: Install cluster software and configure network settings.
2. **Add the node**:

   ```
   pcs cluster node add new_node_name
   ```

3. **Configure fencing**: Critically important! Set up a fencing device for the new node.

Node removal

Safely remove a node from the cluster. The **`node_name`** specifies the node to be removed.

1. **Remove the node**:

   ```
   pcs cluster node remove node_name
   ```

Additional tips:

- Use **`pcs`** status liberally to monitor cluster health and resource locations.
- Consult resource agent documentation for specific management options.
- Plan maintenance tasks carefully to minimize service disruption.
- Always test fencing configurations after node additions or modifications.

Mastering these resource and node management recipes empowers you to keep your RHEL cluster running smoothly, scale as needed and perform maintenance without breaking a sweat (or your website). Happy clustering!

Conclusion

This chapter provides a comprehensive guide to setting up and managing high availability clusters on RHEL. It covers the installation and configuration of Pacemaker, the creation of high availability clusters, and the management of cluster resources and nodes. The chapter also includes detailed steps for configuring active/passive Apache and NFS servers to ensure service continuity.

In the next chapter, we will discuss the security hardening of RHEL, focusing on implementing best practices and configurations to enhance system security. This includes monitoring and applying security updates, configuring firewalls, enabling FIPS mode, managing SELinux policies, implementing secure communications, and securing network services such as NFS.

Points to remember

- Pacemaker is a crucial component for managing HA clusters on RHEL.
- Always configure fencing in production environments to avoid split-brain scenarios.
- Regularly monitor cluster status and perform maintenance tasks without disrupting services.
- Testing your setup in a non-production environment is recommended before deploying in production.
- Refer to official Red Hat documentation for best practices and advanced configurations.

Chapter 14 Security Hardening of RHEL

Introduction

In this chapter, you will learn how to implement security hardening in RHEL. Security hardening involves applying best practices and configurations to reduce vulnerabilities and enhance the overall security posture of your system. By the end of this chapter, you will be equipped with the knowledge to secure various services and components within RHEL, ensuring that your systems are protected against potential threats.

Structure

In this chapter, we will cover the following topics:

- Check available security advisories from the console
- Identify available security updates not yet installed
- Install a specific security update from an advisory
- Set up RHEL to install security updates automatically
- Implement secure disk partitions
- Set up and configure the firewall service firewalld
- Implement nftables for a network-wide firewall service

- Set up RHEL in Federal Information Processing Standard mode
- Implement and manage SELinux in RHEL
- Implement OpenSSH for secure communications
- Configure OpenSSH clients with system roles
- Implement SSL and TLS
- Set up a VPN with IPSec
- Secure NFS services

Objectives

Topics to be covered:

- **Monitor and apply security updates**: Use **Red Hat Security Advisories** (**RHSA**) to ensure that your RHEL systems are secure.
- **Identify and install critical security updates**: Recognize and apply pending security updates to maintain system integrity.
- **Automate security updates**: Configure RHEL to automatically install security patches, reducing manual overhead.
- **Secure disk partitions**: Implement secure disk partitioning strategies and encrypt partitions using LUKS.
- **Manage firewall settings with firewalld**: Set up and configure **`firewalld`** to manage incoming and outgoing network traffic.
- **Use nftables for advanced firewall management**: Replace **`iptables`** with **`nftables`** for more efficient firewall configurations.
- **Enable FIPS mode for compliance**: Configure RHEL to operate in FIPS mode to comply with federal cryptographic standards.
- **Manage SELinux policies**: Enforce strict access controls using SELinux to secure your RHEL environment.
- **Secure remote access with OpenSSH**: Configure OpenSSH to secure remote communications and manage SSH clients using system roles.
- **Implement SSL/TLS for secure communications**: Configure SSL/TLS on web servers to encrypt data transmitted over the network.
- **Set up a secure VPN**: Establish a secure VPN using IPSec to protect data during transmission.
- **Secure Network File System (NFS) services**: Apply best practices for securing NFS to prevent unauthorized access to shared files.

Recipe #77: Check available security advisories from the console

Maintaining the security of enterprise systems requires regular monitoring and application of security updates. Red Hat provides a comprehensive system for managing these updates through RHSA. **Red Hat Security** (**RHS**) notifications document security flaws identified and fixed in Red Hat products and services. These notifications are part of Red Hat's security advisory system, called **RHSA** (**Red Hat Security Advisory**). Each RHSA contains critical information, including:

- **Severity**: The level of threat posed by the security flaw.
- **Type and status**: The nature and current status of the issue.
- **Affected products**: List of Red Hat products impacted by the flaw.
- **Summary of fixed issues**: Brief details about the issues addressed.
- **CVE numbers**: Common vulnerabilities and exposures numbers, providing additional details such as attack complexity and impact.

Security advisories are also available on the Red Hat Customer Portal, where they can be filtered by product, variant, version, and architecture.

To check which security updates have already been applied to your system, you can use the `dnf updateinfo` command as described in the following recipes.

Procedure:

1. Run the following command to list all security updates that are installed on the host:

   ```
   dnf updateinfo list security –installed
   ```

2. The output will include details of the installed security advisories, showing which packages have been updated for security reasons:

   ```
   RHSA-2019:1234 Important/Sec. libssh2-1.8.0-7.
   module+el8+2833+c7d6d092
   
   RHSA-2019:4567 Important/Sec. python3-libs-3.6.7.1.el8.x86_64
   ```

Regularly monitoring and applying security updates is critical for maintaining the security of your Red Hat systems. The `dnf updateinfo` utility is a powerful tool allowing administrators to list pending and installed security updates, ensuring that systems remain protected against known vulnerabilities.

Recipe #78: Identify available security updates not yet installed in RHEL

To identify available security updates that still need to be installed on an RHEL system, you can use the **`dnf`** package manager. The following steps will guide you through the process:

Open a Terminal and run the following command to list all available security updates that have not yet been installed on your system:

```
dnf updateinfo list updates security
```

1. **Review the output**:

   a) The command will produce a list of security advisories that apply to your system but have not yet been installed. The output will include important details such as the advisory ID, the severity of the update, and the packages affected.

   Example output:

   ```
   RHSA-2024:0123 Important/Sec. kernel-5.14.0-70.13.1.el9_0.x86_64
   RHSA-2024:0456 Moderate/Sec. systemd-249-19.el9_1.1.x86_64
   RHSA-2024:0789 Low/Sec.  openssl-3.0.1-41.el9_0.x86_64
   ```

   b) Here, **`RHSA-2024:0123`** refers to a Red Hat Security Advisory ID, **`Important/Sec.`** indicates the severity and that it is a security related update, and the package affected is **`kernel-5.14.0-70.13.1.el9_0.x86_64`**.

This command provides a straightforward way to check for pending security updates, ensuring that you can keep your RHEL system secure by applying necessary patches on time.

Recipe #79: Install a specific security update from an advisory

You can investigate a specific security advisory and retrieve detailed information using its ID.

Procedure:

1. Run the following command to display details of a specific advisory:

   ```
   dnf updateinfo info RHSA-2019:0997
   ```

2. The output will include comprehensive information about the advisory, such as the update ID, type, date, bugs, CVEs, and a detailed description.

```
====================================================================
  Important: python3 security update
====================================================================
  Update ID: RHSA-2019:0997
       Type: security
    Updated: 2019-05-07 05:41:52
        Bugs: 1688543 - CVE-2019-9636 python: Information Disclosure
due to urlsplit improper NFKC normalization
       CVEs: CVE-2019-9636
Description: …
```

After a careful review of the RHSA, you can proceed with installing the update in the system, specifying the exact ID via the command:

```
dnf update --advisory=RHSA-2019:0997
```

Alternatively, you can specify the minor version change using the command:

```
dnf upgrade-minimal --advisory=RHSA-2019:0997
```

3. You can list processes that require a restart of the system after installing the updates:

```
dnf needs-restarting
```

Recipe # 80: Set up RHEL to install security updates automatically

We can configure the RHEL system to download and install security updates automatically. Ensuring your system remains up to date with the latest security patches is critical for maintaining a secure environment.

Prerequisites:

Before proceeding, ensure the following prerequisites are met:

1. **Red Hat subscription**: The system must have an active Red Hat subscription attached to it.
2. **dnf-automatic package**: The **`dnf-automatic`** package should be installed on your system.

Procedure:

Step 1: Verify and configure the dnf-automatic.conf file

1. Open the configuration file **`/etc/dnf/automatic.conf`** in your preferred text editor:

```
sudo vi /etc/dnf/automatic.conf
```

2. Locate the **`[commands]`** section within the file. This section controls the type of updates that will be performed automatically.
3. Ensure that the **`upgrade_type`** option is set to security. This setting restricts updates to only security-related packages:

   ```
   [commands]

   upgrade_type = security
   ```

   Tip: If you wish to install all available updates, set `upgrade_type` to default instead.

4. Save and close the file.

Step 2: Enable and start the dnf-automatic systemd timer

To ensure that the system checks for and installs security updates automatically, you must enable and start the **`dnf-automatic-install.timer`** unit.

1. Run the following command to enable and start the timer:

   ```
   sudo systemctl enable --now dnf-automatic-install.timer
   ```

   This command ensures that the timer is active and will start immediately.

Step 3: Verify the timer status

After enabling the timer, verify that it is active and running correctly:

1. Use the following command to check the status of the timer:

   ```
   sudo systemctl status dnf-automatic-install.timer
   ```

2. The output should indicate that the timer is active. Look for a status line indicating Active: active (waiting).

Verification:

To confirm that security updates are being applied automatically:

1. Monitor the system logs or the output of the **`journalctl`** command for entries related to **`dnf-automatic`**.
2. Optionally, inspect the **`/var/log/dnf.rpm.log`** file to see which packages have been installed or updated automatically.

Additional resources:

For more detailed information, consult the **`dnf-automatic(8)`** man page available on your system:

```
man dnf-automatic
```

Following these steps, RHEL automatically keeps your system secure by applying the latest security updates without requiring manual intervention.

Recipe # 81: Implement secure disk partitions

Secure disk partitioning is a critical aspect of hardening your RHEL system. Configuring your system with appropriate partitions can isolate key directories, improve performance, and enhance security. This recipe recaps the recommended partitioning practices and explains how to use **Linux Unified Key Setup** (**LUKS**) on-disk format to encrypt partitions.

Recommended partitioning scheme

Red Hat recommends creating separate partitions for the following directories:

- **`/boot`**
- **`/`**
- **`/home`**
- **`/tmp`**
- **`/var/tmp`**

/boot partition

The **`/boot`** partition is essential because it stores the boot loader and kernel images required to start your system.

Alert: This partition should not be encrypted to ensure that your system can boot correctly. Including `/boot` within an encrypted / partition may prevent the system from booting if the encrypted partition becomes inaccessible.

/home partition

Storing user data in a separate **`/home`** partition offers several advantages:

- **Data protection**: If the root partition (**`/`**) becomes corrupted, your user data remains intact, reducing the risk of data loss.
- **Ease of upgrade**: Keeping user data in a separate partition simplifies system upgrades, as the **`/home`** directory is not overwritten during the installation of a new RHEL version.
- **System stability**: If user data is stored in **`/`** and the partition fills up, the entire system may become unstable. Using a separate **`/home`** partition prevents this issue.

/tmp and /var/tmp partitions

The **`/tmp`** and **`/var/tmp`** directories store temporary files that do not need long term storage. These directories should be in separate partitions to prevent them from filling up the root partition, which could lead to system instability or crashes.

Encrypting partitions with LUKS

LUKS provides a robust method for encrypting block devices, ensuring that disk data is protected. RHEL uses LUKS to encrypt partitions, and by default, it supports LUKS2, the latest version with enhanced features and security.

Advantages of LUKS:

- **Full disk encryption**: LUKS encrypts entire block devices, making it ideal for securing data on mobile devices, laptops, and removable storage media.
- **Multiple user keys**: LUKS allows multiple user keys to decrypt the master encryption key, enabling secure access for different users.
- **Robust security**: LUKS provides passphrase strengthening to protect against dictionary attacks and supports multiple key slots for backup keys or passphrases.

Configuring LUKS encryption during installation

During the installation of RHEL, you can configure partitions for encryption using the Anaconda installer. The steps below outline how to encrypt your partitions:

1. **Select partitions to encrypt**:
    a) During installation, you have the option to encrypt specific partitions.
    b) You can choose to encrypt partitions such as `/home`, `/var/tmp`, and `/tmp` to enhance security.
2. **Choose a passphrase**:
    a) You will be prompted to enter a passphrase, which serves as the key to unlock the encryption. This passphrase is required each time the system boots.
3. **Default encryption settings**:
    a) The default encryption cipher is `aes-xts-plain64` with a key size of 512 bits. These settings are secure and optimized for most use cases.
4. **Partitioning example**:
    a) For example, to encrypt the `/home` partition during installation, select the partition and choose the encryption option. Enter a strong passphrase when prompted.

Post-installation encryption with LUKS

If you need to encrypt a partition after installation, you can use the `cryptsetup` utility.

1. **Identify the device**:

   a) Use the **`lsblk`** command to list available block devices and identify the one you wish to encrypt.

   ```
   Lsblk
   ```

2. **Encrypt the device**:

   a) Run the **`cryptsetup luksFormat`** command to encrypt the device:

   ```
   sudo cryptsetup luksFormat /dev/nvme0n1p1
   ```

   b) Enter and confirm the passphrase when prompted. This step overwrites all data on the device.

3. **Open the encrypted device**:

   a) Use the **`cryptsetup open`** command to unlock the encrypted device.

   ```
   sudo cryptsetup open /dev/nvme0n1p1 nvme0n1p1_encrypted
   ```

   b) Enter the passphrase to unlock the device, and it will be mapped to a new device path under **`/dev/mapper/`**.

4. **Create a file system**:

   a) Format the unlocked device with a file system, such as **`ext4`**:

   ```
   sudo mkfs.ext4 /dev/mapper/nvme0n1p1_encrypted
   ```

5. **Mount the device**:

   a) Mount the encrypted device to the desired mount point:

   ```
   sudo mount /dev/mapper/nvme0n1p1_encrypted /mnt/encrypted
   ```

Encrypting existing data

To encrypt an existing partition without losing data, follow these steps:

1. **Backup data**: Ensure you have a reliable backup of the data.
2. **Unmount the file system**: Unmount the partition you plan to encrypt.

   ```
   sudo umount /dev/sdx1
   ```

3. **Resize the file system**: If necessary, resize the file system to create space for the LUKS header.
4. **Initialize encryption**: Use **`cryptsetup reencrypt`** to encrypt the device while preserving existing data:

   ```
   sudo cryptsetup reencrypt --encrypt --reduce-device-size 32M /dev/sdx1
   ```

5. **Resume encryption**: If the process is interrupted, use the **`cryptsetup reencrypt --resume-only`** command.

After encrypting partitions, manage them using **`cryptsetup`** commands. Key management, status checks, and passphrase changes are all handled through this utility. Partitioning your RHEL system with separate partitions for key directories and encrypting them using LUKS significantly enhances the security of your environment. By isolating directories and securing them with robust encryption, you protect your data from unauthorized access, system crashes, and potential data loss. Always back up your data before making changes to partitioning or encryption configurations.

Recipe # 82: Set up and configure the firewall service firewalld

The **`firewalld`** service on RHEL provides a dynamic and highly customizable host-based firewall solution. It allows administrators to control incoming and outgoing network traffic by defining a set of rules and zones. **`firewalld`** is a firewall service daemon that offers a dynamic firewall management tool with support for network zones, allowing administrators to assign different security levels to different network interfaces. Unlike static firewall solutions, **`firewalld`** can modify rules on the fly without needing to restart the service, ensuring uninterrupted traffic management.

Key concepts:

- **Zones**: Zones represent different levels of trust for network interfaces or sources. They define which incoming traffic is allowed depending on the network's trust level.
- **Services**: Predefined sets of rules that allow traffic for specific services, such as HTTP or SSH. Services automatically open the required ports and set the necessary protocols.

To effectively manage **`firewalld`** on your system, follow the steps below to install, configure, and manage firewall settings:

1. **Installing firewalld**

   Step 1: Installing the package

   On RHEL, **`firewalld`** is typically installed by default. If it is not installed, you can install it using the following command:

   ```
   sudo dnf install firewalld
   ```

   Step 2: Starting and enabling firewalld

   Ensure that **`firewalld`** starts at boot and is running by executing:

   ```
   sudo systemctl start firewalld
   sudo systemctl enable firewalld
   ```

Step 3: Verifying the firewalld status

To verify that `firewalld` is active and running:

```
sudo systemctl status firewalld
```

2. **Configuring firewall zones**

   Zones define the level of trust for network interfaces or source IP addresses. Each zone has its own set of rules. Below is an overview of some common predefined zones:

   a) **public**: For use in public areas, only selected incoming connections are accepted.

   b) **home**: For home networks, incoming connections from other trusted devices are allowed.

   c) **dmz**: For computers in a **demilitarized zone** (**DMZ**), limited access to the internal network is allowed.

   Example: Assigning an interface to a zone

   To assign a network interface to a specific zone, use the following commands:

   1. **List active zones and interfaces**:

      ```
      sudo firewall-cmd --get-active-zones
      ```

   2. **Assign an interface to a zone**:

      ```
      sudo firewall-cmd --zone=home --change-interface=eth0 --permanent
      ```

   3. **Reload the configuration**:

      ```
      sudo firewall-cmd --reload
      ```

3. **Managing firewall services**

   Services in `firewalld` are predefined rules that enable or disable network access for specific applications. For example, enabling the SSH service will open port 22.

   Example: Allowing SSH traffic

   1. **Enable the SSH service**:

      ```
      sudo firewall-cmd --zone=public --add-service=ssh –permanent
      ```

   2. **Reload the configuration**:

      ```
      sudo firewall-cmd –reload
      ```

   3. **Verify the service**:

      ```
      sudo firewall-cmd --zone=public --list-services
      ```

4. **Opening and closing ports**

Besides services, you can manually open or close specific ports for traffic.

Example: Opening a port for HTTPS (port 443)

1. **Open port 443**:

   ```
   sudo firewall-cmd --zone=public --add-port=443/tcp –permanent
   ```

2. **Reload the configuration**:

   ```
   sudo firewall-cmd --reload
   ```

3. **Verify the port status**:

   ```
   sudo firewall-cmd --zone=public --list-ports
   ```

Example: Closing an unused port

1. **Remove a port**:

   ```
   sudo firewall-cmd --zone=public --remove-port=443/tcp –permanent
   ```

2. **Reload the configuration**:

   ```
   sudo firewall-cmd --reload
   ```

Introduction to the firewall RHEL system role

RHEL system roles are a set of predefined Ansible roles that simplify the configuration of various system services. The **`rhel-system-roles.firewall`** role allows for automated configuration of **`firewalld`** across multiple RHEL systems.

Key features:

- **Automated configuration**: Apply consistent firewall settings across multiple systems using a single Ansible playbook.
- **Zone management**: Define zones, services, ports, and rules centrally.
- **Scalable**: Efficiently manage large-scale environments.

Example: Resetting firewalld settings using Ansible

Below is an example of an Ansible playbook that automates the process of opening port 443 in the public zone of **`firewalld`**, reloading the configuration, and then verifying the status of the port. Below is the Ansible playbook code to open port 443 and verify:

```
---
- name: Open Port 443 in firewalld using RHEL system role
  hosts: all
  become: true  # Ensure the playbook runs with sudo privileges
  tasks:
    - name: Configure firewalld to open port 443
      ansible.builtin.include_role:
```

```
        name: rhel-system-roles.firewall
      vars:
        firewall:
          - port: 443/tcp
            state: enabled
            runtime: true
            permanent: true
    - name: Verify that port 443 is open
      ansible.builtin.command:
        cmd: firewall-cmd --zone=public --list-ports
      register: firewalld_ports_output
    - name: Display the open ports in the public zone
      ansible.builtin.debug:
        msg: "{{ firewalld_ports_output.stdout }}"
```

Explanation:

1. **Open port 443/tcp in the public zone**: The first task uses the **`rhel-system-roles.firewall`** role to open port 443 for TCP traffic. The role ensures that the port is enabled across system reboots at runtime and permanently.
2. **Verify that port 443 is open in the public zone**: The second task checks the list of open ports in the public zone by running the **`firewall-cmd --zone=public --list-ports`** command. The output of this command is stored in the **`firewalld_ports_output`** variable for verification.
3. **Display the open ports in the public zone**: The final task uses the debug module to print out the list of open ports in the public zone, confirming whether port 443 was successfully added. This step provides visibility into the current firewall configuration.

Running the playbook:

Save the playbook as **`open_port_443.yml`**, and run it with the following command:

```
ansible-playbook open_port_443.yml
```

The **`become:  true`** directive ensures that all commands are executed with elevated privileges (i.e., as root or using **`sudo`**).

This Ansible playbook makes it easy to apply the same configuration across multiple managed nodes in your environment by automating the firewall configuration.

The **`firewalld`** service on RHEL provides robust and flexible firewall management with its dynamic rule application and zone based configuration. For administrators managing multiple systems, the **`rhel-system-roles.firewall`** role in Ansible offers a powerful way to ensure consistent and automated firewall settings across environments. Understanding and implementing the practices can effectively secure your RHEL systems against unwanted network traffic.

Recipe #83: Implement nftables for a network-wide firewall service

The evolution of firewall management in RHEL has led to the adoption of **`nftables`**, a powerful and flexible framework for packet filtering and **network address translation** (**NAT**). Replacing the legacy **`iptables`**, **`nftables`** offers numerous improvements, such as built-in lookup tables, atomic rule changes, and a more consistent syntax. **`nftables`** brings several advantages over its predecessors:

- **Built-in lookup tables**: Instead of processing rules linearly, **`nftables`** uses lookup tables for faster packet filtering.
- **Unified framework**: **`nftables`** supports both IPv4 and IPv6 under a single framework, simplifying rule management.
- **Atomic rule changes**: All rule changes are applied atomically, ensuring the firewall is always in a consistent state.
- **Enhanced debugging**: With built-in tracing and debugging tools, **`nftables`** makes it easier to troubleshoot firewall rules.

Migrating from iptables to nftables

If your existing firewall setup relies on **`iptables`**, migrating to **`nftables`** is straightforward using the **`iptables-restore-translate`** utility. This tool translates your current **`iptables`** rules into **`nftables`** syntax, ensuring a smooth transition.

Steps to migrate:

1. **Save current iptables rules**:
   ```
   iptables-save > /root/iptables.dump
   ip6tables-save > /root/ip6tables.dump
   ```
2. **Convert iptables rules to nftables**:
   ```
   iptables-restore-translate -f /root/iptables.dump > /etc/nftables/ruleset-migrated-from-iptables.nft
   ip6tables-restore-translate -f /root/ip6tables.dump > /etc/nftables/ruleset-migrated-from-ip6tables.nft
   ```
3. **Update nftables configuration**: Include the translated rules in your **`nftables`** configuration:
   ```
   include "/etc/nftables/ruleset-migrated-from-iptables.nft"
   ```

4. **Disable iptables and enable nftables**:

   ```
   systemctl disable --now iptables
   systemctl enable --now nftables
   ```

5. **Verify the migration**:

   ```
   nft list ruleset
   ```

Writing and executing nftables scripts

One of the strengths of **`nftables`** is the ability to write scripts for your firewall configuration. These scripts can be executed directly or passed to the **`nft`** utility.

Example script format:

```
#!/usr/sbin/nft -f
# Flush the existing rule set
flush ruleset
table inet example_table {
  chain input {
    type filter hook input priority 0; policy drop;
    # Allow SSH
    tcp dport 22 accept
  }
}
```

Running the script:

- **Direct execution**:

  ```
  /etc/nftables/example_script.nft
  ```

- **Using the nft utility**:

  ```
  nft -f /etc/nftables/example_script.nft
  ```

Configuring NAT with nftables

NAT is essential for managing how traffic is routed between networks. **`nftables`** supports several types of NAT configurations, including masquerading, SNAT, and DNAT. SNAT (Source Network Address Translation) changes the source IP address, while DNAT (Destination Network Address Translation) modifies the destination IP address of packets, both facilitating efficient network traffic routing and communication.

Example: Configuring masquerading

Masquerading is typically used when your internal network uses private IP addresses, and you want to enable access to the public internet.

```
nft add table ip nat
nft add chain ip nat postrouting { type nat hook postrouting priority 100
\; }
nft add rule ip nat postrouting oifname "ens3" masquerade
```

Example: Forwarding ports

To forward incoming traffic on a specific port (e.g., port 80) to a different internal IP address:

```
nft add table ip nat
nft add chain ip nat prerouting { type nat hook prerouting priority -100 \; }
nft add rule ip nat prerouting tcp dport 80 dnat to 192.0.2.1:8080
```

Advanced features: using sets and maps

`nftables` allows you to manage complex firewall configurations using sets and verdict maps. Sets allow for grouping similar elements, such as IP addresses or ports, while verdict maps provide a way to assign actions based on matching criteria.

Example: Using sets

```
nft add set inet example_table example_set { type ipv4_addr \; }
nft add element inet example_table example_set { 192.0.2.1, 192.0.2.2 }
nft add rule inet example_table example_chain ip saddr @example_set drop
```

Example: Using verdict maps

```
nft add map inet example_table example_map { type ipv4_addr : verdict \; }
nft add rule inet example_table example_chain ip saddr vmap @example_map
nft add element inet example_table example_map { 192.0.2.1 : accept, 192.0.2.2
: drop }
```

Debugging and monitoring

`nftables` provides robust tools for debugging and monitoring your firewall rules, including counters and tracing.

Example: Adding a counter

```
nft add rule inet example_table example_chain tcp dport 22 counter accept
```

Example: Enabling tracing

```
nft replace rule inet example_table example_chain handle 4 tcp dport 22
meta nftrace set 1 accept
nft monitor
```

`nftables` represents a significant advancement in firewall management on RHEL, offering greater flexibility, performance, and ease of use compared to its predecessors. Whether you are migrating from `iptables` or starting fresh, `nftables` provides the tools necessary to manage complex networking environments effectively.

By embracing **`nftables`**, administrators can leverage its powerful features to create secure, efficient, and scalable firewall configurations tailored to their specific needs.

Recipe #84: Set up RHEL in Federal Information Processing Standard mode

Federal Information Processing Standards (**FIPS**) 140 is a series of security standards developed by the *National Institute of Standards and Technology* (*NIST*) to validate the quality and security of cryptographic modules. Running RHEL in FIPS mode ensures that all cryptographic operations comply with these standards. This guide will walk you through setting up RHEL in FIPS mode, ensuring your system meets cryptographic security requirements. Enabling FIPS mode on your RHEL system ensures that:

- All cryptographic operations use FIPS-approved algorithms.
- The system performs self-checks to validate the integrity and correctness of cryptographic operations.
- Cryptographic key material is generated in a secure and compliant manner.

Important considerations

- **Pre-installation configuration**: The recommended method for ensuring full FIPS compliance is to enable FIPS mode during the installation of RHEL. This guarantees that all cryptographic key material is generated using FIPS approved algorithms from the start.
- **Post-installation enabling**: If you enable FIPS mode after installation, note that the system may not fully comply with FIPS 140 as some cryptographic keys generated prior to enabling FIPS mode might not meet the standards.
- **Irreversibility**: Once FIPS mode is enabled, disabling it without compromising the system's integrity is not straightforward. If you need to disable FIPS mode, a complete reinstallation of the system is recommended.

Enabling FIPS mode during installation

To ensure that all cryptographic keys are generated using FIPS-approved algorithms, it is best to enable FIPS mode during the installation of RHEL.

Procedure:

1. **Add FIPS kernel parameter**: During the RHEL installation process, add the **`fips=1`** option to the kernel command line. This can be done at the GRUB boot menu by editing the boot options.

   ```
   linux /vmlinuz ... fips=1
   ```

2. **Avoid third-party software**: During the software selection phase, avoid installing any third-party software. This ensures that only RHEL-approved packages that are compliant with FIPS standards are used.
3. **Complete the installation**: Finish the installation process as usual. The system will automatically start in FIPS mode.
4. **Verification**: After the system boots, verify that FIPS mode is enabled by running:

   ```
   fips-mode-setup --check
   Expected output:
   FIPS mode is enabled.
   ```

Switching an installed system to FIPS mode

If your RHEL system is already installed, you can still enable FIPS mode using the `fips-mode-setup` utility. However, this method may not fully guarantee FIPS compliance if cryptographic keys were generated before enabling FIPS mode.

Procedure:

1. **Enable FIPS mode**:

   Run the following command as the root user to enable FIPS mode:

   ```
   fips-mode-setup –enable
   ```

   You should see output indicating that the kernel `initramdisks` are being regenerated.
2. **Reboot the system**: After enabling FIPS mode, reboot the system to apply the changes:

   ```
   reboot
   ```
3. **Verification**: Once the system reboots, verify that FIPS mode is enabled by running:

   ```
   fips-mode-setup --check
   Expected output:
   FIPS mode is enabled.
   ```

Enabling FIPS mode in a container

FIPS mode can also be enabled within containers running on a host system that is already in FIPS mode. The host system must already have FIPS mode enabled. The `podman` utility automatically enables FIPS mode for supported containers.

- **Container creation**: When you create a container on a FIPS-enabled host, `podman` will automatically enable FIPS mode within the container.

- **Verification**: To verify that FIPS mode is enabled inside a container, you can check the status within the container's shell.

While most core cryptographic components in RHEL adhere to FIPS 140 standards, there are exceptions. Certain applications, such as MySQL, Kerberos, and Samba, may use cryptographic algorithms that are not compliant with FIPS 140-3. Be mindful of these exceptions when configuring your RHEL system for environments requiring strict FIPS compliance.

Setting up RHEL in FIPS mode is crucial for organizations that need to comply with federal cryptographic standards. By following the procedures outlined above, you can ensure that your RHEL system is configured to use only FIPS approved cryptographic algorithms, providing the necessary security and compliance for sensitive environments.

Remember, enabling FIPS mode during installation is the best way to guarantee full compliance. Post-installation enabling of FIPS mode is possible but may not cover all aspects of cryptographic compliance. If you require detailed guidance on specific applications or further assistance, consult Red Hat's official documentation or support services.

Recipe #85: Implement and manage SELinux in RHEL

Security-Enhanced Linux (SELinux) is a security architecture integrated into the Linux kernel that provides a robust mechanism for enforcing access control policies. SELinux in RHEL enhances security by applying **mandatory access control** (**MAC**) policies, which limit the access of resources that users and processes can access.

SELinux operates on the principle of least privilege, ensuring that every process and user has only the necessary permissions required for their functions. This approach helps mitigate the impact of vulnerabilities by containing potential exploits within the boundaries of strict access controls.

SELinux states and modes

In RHEL, SELinux can operate in three different modes:

- **Enforcing**: SELinux policy is enforced, and access violations are both logged and denied.
- **Permissive**: SELinux policy is not enforced; however, violations are logged. This mode is useful for troubleshooting.
- **Disabled**: SELinux is completely disabled.

To check the current status of SELinux, use the **`sestatus`** command:

```
$ sestatus
```

To switch SELinux modes, the **setenforce** command is used:

- Switch to enforcing mode:

  ```
  $ sudo setenforce 1
  ```

- Switch to permissive mode:

  ```
  $ sudo setenforce 0
  ```

For permanent changes, modify the **/etc/selinux/config** file.

Managing SELinux users and roles

SELinux differentiates between confined and unconfined users. Confined users are restricted by SELinux policies, while unconfined users are not.

To manage SELinux users:

- **Adding a new SELinux user**:

  Use the **semanage** command to add a new user mapped to a SELinux role:

  ```
  $ sudo semanage user -a -R “SELinux_Role” new_user
  ```

- **Assigning SELinux roles**:

  ```
  $ sudo semanage login -a -s SELinux_User login_name
  ```

SELinux roles can be used to enforce additional restrictions on users, enhancing the security model.

Configuring SELinux for applications

SELinux policies can be customized for applications requiring specific permissions or those not conform to standard configurations.

- **Customizing policies for Apache HTTP server**:

  If Apache needs to serve content from a non-standard directory, update its SELinux context using:

  ```
  $ sudo semanage fcontext -a -t httpd_sys_content_t "/custom/
  directory(/.*)?"
  $ sudo restorecon -Rv /custom/directory
  ```

- **Using SELinux Booleans**:

  Booleans are a set of policy variables that can be toggled on or off to allow or restrict certain features without altering the core SELinux policy.

  List available Booleans:

  ```
  $ sudo semanage boolean -l
  ```

Enable a Boolean:

```
$ sudo setsebool boolean_name on
```

This method allows you to fine-tune the security policy according to the application's requirements.

Troubleshooting SELinux

When SELinux denies access to a resource, it logs the event. These logs can be analyzed to understand and resolve the issue.

- **Identifying and analyzing denials**:

  SELinux denials are logged in **`/var/log/audit/audit.log`**. The **`ausearch`** command can help filter these logs:

  ```
  $ sudo ausearch -m avc -ts recent_date
  ```

  For more detailed analysis, use the **`sealert`** command, which provides suggestions on how to resolve issues:

  ```
  $ sudo sealert -a /var/log/audit/audit.log
  ```

- **Creating custom policies**:

  In cases where the existing policies do not suffice, you may need to create custom SELinux policy modules using tools like **`audit2allow`**:

  ```
  $ sudo audit2allow -M mymodule < /path/to/log/file
  $ sudo semodule -i mymodule.pp
  ```

  This command generates and installs a module from the audit log, allowing the previously denied actions.

SELinux in RHEL provides powerful mechanisms to secure your system against unauthorized access. Properly configuring and managing SELinux is crucial for maintaining a secure environment, especially in systems with high security requirements. Regular monitoring and analysis of SELinux logs, combined with correct policy adjustments, ensure the system remains secure and functional. Administrators can significantly enhance the security posture of their RHEL systems, making them resilient against various security threats.

Recipe # 86: Implement OpenSSH for secure communications

OpenSSH (**Open Secure Shell**) is a suite of tools that provides encrypted communication sessions over a computer network using the SSH protocol. It is a critical component for securely managing remote servers and is widely used and tunnelling. Implementing OpenSSH on RHEL ensures secure communication and administrative tasks across systems.

Installing and configuring OpenSSH

1. **Installation of OpenSSH server and client**

   RHEL typically comes with OpenSSH installed by default. However, if it is not installed, you can install it using the `dnf` package manager:

   ```
   $ sudo dnf install openssh-server openssh-clients
   ```

   Ensure that the OpenSSH server (`sshd`) is enabled and running:

   ```
   $ sudo systemctl enable sshd
   $ sudo systemctl start sshd
   ```

   To verify the status of the SSH service:

   ```
   $ sudo systemctl status sshd
   ```

2. **Firewall configuration**

   To allow SSH traffic through the firewall, you need to configure `firewalld`:

   ```
   $ sudo firewall-cmd --permanent --add-service=ssh
   $ sudo firewall-cmd –reload
   ```

   This ensures SSH traffic is permitted through the server's firewall, enabling remote connections.

3. **Basic SSH configuration**

   The main configuration file for the OpenSSH server is located at /etc/ssh/sshd_config. You can modify this file to enhance security and customize the SSH service. Here are some key configuration options:

   a. **Changing the default port**: By default, SSH runs on port 22. For security purposes, you might want to change it to a non-standard port:

      ```
      Port 2222
      ```

      After changing the port, restart the SSH service:

      ```
      $ sudo systemctl restart sshd
      ```

   b. **Disabling root login**: Disabling root login via SSH is a common security practice. Instead, administrators should use a regular user account and escalate privileges as needed:

      ```
      PermitRootLogin no
      ```

   c. **Enforcing SSH protocol version 2**: Ensure that only SSH protocol version 2 is allowed, as it is more secure than version 1:

      ```
      Protocol 2
      ```

d. **Configuring authentication methods**: You can enforce stricter authentication methods by disabling password authentication and using SSH keys instead:

```
PasswordAuthentication no
PubkeyAuthentication yes
```

SSH keys provide a more secure authentication mechanism than passwords.

4. **Generating SSH keys**

SSH key pairs (public and private keys) are a more secure method of authentication. To generate a key pair, use the following command on the client machine:

```
$ ssh-keygen -t rsa -b 4096 -C your_email@example.com
```

This command generates a key pair with a 4096-bit RSA key. The private key is stored on the client, and the public key is added to the **`~/.ssh/authorized_keys`** file on the server.

a. **Copying the public key to the server**: Use the **`ssh-copy-id`** command to copy your public key to the remote server:

```
$ ssh-copy-id user@server_ip
```

This adds the public key to the **`authorized_keys`** file on the server, allowing for key based authentication.

5. **Securing SSH with SELinux**

RHEL uses SELinux for additional security. Ensure SELinux policies are correctly configured for SSH:

a. **Allowing SSH home directory access**: By default, SELinux might prevent SSH from accessing a user's home directory if it is in a non-standard location. Use the following command to allow SSH access:

```
$ sudo setsebool -P ssh_home_tty 1
```

b. **Troubleshooting SELinux denials**:

If SELinux blocks an SSH operation, check the audit logs for denials:

```
$ sudo ausearch -m avc -c sshd
```

If necessary, generate an SELinux policy module to allow the desired behaviour:

```
$ sudo audit2allow -M mypol < /path/to/audit.log
$ sudo semodule -i mypol.pp
```

6. **Managing SSH access control**

Control who can access your system via SSH by configuring the **`/etc/ssh/sshd_config`** file:

a. **Allowing or denying users**: To allow or deny specific users or groups:

```
AllowUsers user1 user2
AllowGroups admin
DenyUsers user3
DenyGroups guests
```

b. **Using TCP wrappers**:

TCP Wrappers can further restrict SSH access based on hostname or IP address. Configure `/etc/hosts.allow` and `/etc/hosts.deny` to control access.

Example:

```
sshd: 192.168.1.0/24
sshd: ALL : DENY
```

This allows SSH access only from the specified subnet and denies all others.

7. **Auditing and monitoring SSH**

Regularly audit SSH access to detect unauthorized attempts:

a. **Monitoring SSH logs**: SSH logs can be found in `/var/log/secure` or `/var/log/auth.log`. Use grep to filter for SSH-specific entries:

```
$ sudo grep sshd /var/log/secure
```

b. **Using Fail2Ban for SSH protection**:

Install and configure Fail2Ban to protect against brute force attacks:

```
$ sudo dnf install fail2ban
```

Configure Fail2Ban to monitor SSH and ban IPs after too many failed attempts by editing `/etc/fail2ban/jail.local`:

```
[sshd]
enabled = true
port = 2222
filter = sshd
logpath = /var/log/secure
maxretry = 5
```

Restart Fail2Ban to apply the changes:

```
$ sudo systemctl restart fail2ban
```

Implementing OpenSSH in RHEL is crucial for maintaining secure, encrypted network communications. By carefully configuring the SSH server, employing best practices for security, and leveraging RHEL security features like SELinux and `firewalld`, you can ensure robust protection against unauthorized access and attacks. Regular auditing and monitoring further enhance the security posture of your SSH implementation, making RHEL a secure platform for remote system management.

Recipe # 87: Configure OpenSSH clients with system roles

System roles in RHEL are predefined Ansible roles that simplify the management and configuration of various system services, including OpenSSH. Using system roles, you can efficiently configure OpenSSH clients across multiple systems consistently and automatically. This guide outlines the steps to configure OpenSSH clients using system roles in RHEL.

Prerequisites:

1. **Install Ansible and RHEL system roles**: Ensure that Ansible and the necessary system roles are installed on the control node. This can be done with the following command:

   ```
   $ sudo dnf install ansible rhel-system-roles
   ```

2. **Set up SSH access**: Ensure that SSH access is properly configured between the control node and the target systems, allowing Ansible to communicate with them.

Step 1: Identify the OpenSSH system role

RHEL provides a specific system role for OpenSSH, known as **`rhel-system-roles.ssh`**. This role can be used to configure both the SSH client and server settings.

Step 2: Create an Ansible playbook for OpenSSH client configuration

Create a YAML playbook to apply the OpenSSH configuration across your clients:

```
---
- name: Configure OpenSSH Clients with System Roles
  hosts: all
  become: true
  roles:
    - rhel-system-roles.ssh
  vars:
    ssh_client:
      - option: Host
        value: "*"
      - option: ForwardX11
        value: "yes"
      - option: Protocol
        value: "2"
```

```
      - option: ServerAliveInterval
        value: "60"
      - option: StrictHostKeyChecking
        value: "no"
```

Step 3: Define variables for SSH client configuration

In the Ansible playbook, the **`ssh_client`** variable is used to define the configuration options for the OpenSSH client. You can adjust these variables according to your specific requirements:

- **Host**: Defines the hosts to which the configuration applies.
- **ForwardX11**: Enables X11 forwarding.
- **Protocol**: Enforces the use of SSH protocol version 2.
- **ServerAliveInterval**: Sets a timeout interval to keep the connection alive.
- **StrictHostKeyChecking**: Disables strict host key checking, useful in certain environments.

You can add more options as needed by expanding the **`ssh_client`** list.

Step 4: Apply the Ansible playbook

Run the Ansible playbook to apply the configuration across all target systems:

```
$ ansible-playbook configure_ssh_clients.yml
```

Step 5: Verify the configuration

After the playbook is executed, verify that the SSH client configuration has been applied correctly on the target systems:

```
$ cat ~/.ssh/config
```

Ensure that the options defined in the playbook are reflected in the SSH client configuration.

Advanced configuration options:

You can further customize the OpenSSH client configuration by adding more complex options to the playbook. For example, if you want to configure specific settings for different hosts, you can define multiple **`Host`** sections within the **`ssh_client`** variable.

Example:

```
ssh_client:
  - option: Host
    value: "server1.example.com"
  - option: User
    value: "admin"
  - option: Port
```

```
    value: "2222"
  - option: Host
    value: "server2.example.com"
  - option: User
    value: "user2"
  - option: Port
    value: "2200"
```

Using system roles in RHEL to configure OpenSSH clients allows for a consistent and efficient setup across multiple systems. By leveraging Ansible and the **`rhel-system-roles.ssh`** role, administrators can automate the management of SSH configurations, reducing manual effort and ensuring that best practices are uniformly applied. This method facilitates easy updates and modifications to the SSH configuration as requirements evolve.

Recipe #88: Implement SSL and TLS

Secure Sockets Layer (**SSL**) and **Transport Layer Security** (**TLS**) are protocols used to secure communication over a network. TLS is the successor to SSL and provides stronger encryption and security mechanisms. Implementing these protocols in RHEL ensures that your network communications, especially over HTTP, email, and other services, are encrypted and secure.

Step 1: Install required packages

First, ensure that the necessary packages for SSL/TLS and web server (e.g., Apache or Nginx) are installed.

For Apache HTTP server:

```
sudo dnf install httpd mod_ssl openssl
```

For NGINX:

```
sudo dnf install nginx openssl
```

Step 2: Generate an SSL/TLS certificate

You can use OpenSSL to generate a self-signed SSL/TLS certificate. While self-signed certificates are suitable for internal testing, it is recommended to obtain a certificate from a trusted **certificate authority** (**CA**) for production environments.

1. **Generate a private key**:

   ```
   sudo openssl genpkey -algorithm RSA -out /etc/ssl/private/server.key
   -aes256
   ```

2. **Create a certificate signing request (CSR)**:

   ```
   sudo openssl req -new -key /etc/ssl/private/server.key -out /etc/ssl/
   certs/server.csr
   ```

During this step, you will be prompted to enter details such as your country, state, organization, and common name (usually the domain name of your server).

3. **Generate a self-signed certificate**:

   ```
   sudo openssl x509 -req -days 365 -in /etc/ssl/certs/server.csr -signkey
   /etc/ssl/private/server.key -out /etc/ssl/certs/server.crt
   ```

   This certificate will be valid for 365 days.

Step 3: Configure Apache to use SSL/TLS

If you are using Apache, follow these steps to enable SSL/TLS:

1. **Edit the SSL configuration file**:

   Open the SSL configuration file:

   ```
   sudo nano /etc/httpd/conf.d/ssl.conf
   ```

2. **Configure the SSL/TLS settings**:

   Ensure the following directives are correctly set:

   ```
   SSLEngine on
   SSLCertificateFile /etc/ssl/certs/server.crt
   SSLCertificateKeyFile /etc/ssl/private/server.key
   ```

   Optionally, you can configure additional SSL/TLS parameters for stronger security:

   ```
   SSLProtocol all -SSLv3 -TLSv1 -TLSv1.1
   SSLCipherSuite HIGH:!aNULL:!MD5
   SSLHonorCipherOrder on
   ```

3. **Restart Apache**:

   Apply the changes by restarting Apache:

   ```
   sudo systemctl restart httpd
   ```

Step 4: Configure NGINX to use SSL/TLS

If you are using NGINX, follow these steps:

1. **Edit the NGINX server block**:

   Open the configuration file for your site, typically located in **`/etc/nginx/conf.d/`** or **`/etc/nginx/sites-available/`**:

   ```
   sudo nano /etc/nginx/conf.d/your_domain.conf
   ```

2. **Add the SSL/TLS configuration**:

   Update the server block to include the SSL settings:

   ```
   server {
   ```

```
    listen 443 ssl;
    server_name your_domain.com;

    ssl_certificate /etc/ssl/certs/server.crt;
    ssl_certificate_key /etc/ssl/private/server.key;

    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers HIGH:!aNULL:!MD5;
    ssl_prefer_server_ciphers on;

    location / {
        root /var/www/html;
        index index.html;
    }
}
```

Ensure that port 80 is redirected to port 443 for HTTPS:

```
server {
    listen 80;
    server_name your_domain.com;
    return 301 https://$server_name$request_uri;
}
```

3. **Restart NGINX:**

   Apply the changes by restarting NGINX:

   ```
   sudo systemctl restart nginx
   ```

Step 5: Test SSL/TLS configuration

After configuring SSL/TLS, it is important to test the setup:

1. **Use OpenSSL to test connectivity**:

   ```
   openssl s_client -connect your_domain.com:443
   ```

   This command will show details about the SSL/TLS connection, including the certificate used and the protocols supported.

2. **Check for SSL/TLS vulnerabilities**:

   Use tools like SSL Labs' SSL test to analyze your server's SSL/TLS configuration for vulnerabilities and ensure compliance with best practices.

Step 6: Automate SSL/TLS certificate renewal (optional)

For production environments, it is recommended to use *Let's Encrypt* to automate SSL/TLS certificate issuance and renewal.

1. **Install Certbot**:

For NGINX:

```
sudo dnf install certbot python3-certbot-nginx
```

2. **Obtain and install a certificate**:

   For Apache:

   ```
   sudo certbot –apache
   ```

   For NGINX:

   ```
   sudo certbot –nginx
   ```

   Follow the prompts to obtain and configure your SSL/TLS certificate automatically.

3. **Automate Renewal:**

   Certbot automatically sets up a cron job to renew the certificates. You can manually test the renewal process with:

   ```
   sudo certbot renew --dry-run
   ```

   Implementing SSL and TLS on RHEL is crucial for securing network communications, especially for web servers. You can ensure that your data is encrypted and secure by configuring SSL/TLS with Apache or NGINX and using tools like OpenSSL. Additionally, automating certificate management with *Let's Encrypt* simplifies the process of maintaining SSL/TLS security over time.

Recipe #89: Set up a VPN with IPSec

Internet Protocol Security (**IPSec**) is a suite of protocols used to secure Internet Protocol (IP) communications by authenticating and encrypting each IP packet in a communication session. Setting up a **virtual private network** (**VPN**) with IPSec on RHEL ensures secure communication over potentially insecure networks.

This guide will walk you through the process of setting up an IPSec based VPN using Libreswan, a popular IPSec implementation.

Step 1: Install required packages

First, install the necessary packages for IPSec:

```
sudo dnf install libreswan
```

Libreswan is a widely used implementation of IPSec, and it provides the tools needed to set up an IPSec VPN.

Step 2: Configure the IPSec VPN

The IPSec configuration is typically handled through the `/etc/ipsec.conf` file and individual connection definitions in the `/etc/ipsec.d/` directory.

1. **Edit the main configuration file**:

   Open the **`/etc/ipsec.conf`** file for editing:

   ```
   sudo nano /etc/ipsec.conf
   ```

   Ensure the configuration includes the following:

   ```
   config setup
       protostack=netkey
       uniqueids=no
   ```

   a. **protostack=netkey**: Specifies the use of the native IPSec stack in the Linux kernel.

   b. **uniqueids=no**: Allows multiple connections from the same client.

2. **Create a connection definition**:

   Define the VPN connection by creating a new file in the **`/etc/ipsec.d/`** directory, for example, **`vpn.conf`**:

   ```
   sudo nano /etc/ipsec.d/vpn.conf
   ```

   Add the following configuration to define your IPSec connection:

   ```
   conn myvpn
       authby=secret
       auto=start
       left=your.server.ip
       leftid=@yourserverdomain.com
       leftsubnet=your.server.subnet/24
       right=%any
       rightid=@yourclientdomain.com
       rightsubnet=your.client.subnet/24
       ike=aes256-sha2_512;modp2048
       phase2alg=aes256-sha2_512;modp2048
       ikelifetime=8h
       salifetime=1h
       pfs=yes
       dpddelay=30
       dpdtimeout=120
       dpdaction=restart
   ```

   a. **left**: The IP address of your VPN server.

   b. **leftid**: The identifier for your server (usually the domain name).

   c. **leftsubnet**: The subnet behind the VPN server that should be accessible via VPN.

d. **right**: The client IP address or **`%any`** to allow any client to connect.
e. **rightid**: The identifier for the VPN client.
f. **rightsubnet**: The client's subnet that should be accessible via VPN.
g. **ike and phase2alg**: Define the encryption and hashing algorithms used.
h. **dpd**: Dead Peer Detection, which ensures that the VPN connection is terminated if the peer becomes unresponsive.

3. **Define the shared secret**:

   The shared secret for the VPN is defined in the `/etc/ipsec.secrets` file:

   ```
   sudo nano /etc/ipsec.secrets
   ```

   Add the following line:

   ```
   @yourserverdomain.com @yourclientdomain.com : PSK "yoursharedsecret"
   ```

   Replace **`yoursharedsecret`** with a strong, unique passphrase.

Step 3: Start and enable the IPSec service

Once the configuration is complete, start the **`ipsec`** service and enable it to start on boot:

```
sudo systemctl start ipsec
sudo systemctl enable ipsec
```

You can check the status of the IPSec service to ensure it is running correctly:

```
sudo systemctl status ipsec
```

Step 4: Firewall configuration

Ensure that the necessary ports for IPSec are open on the firewall:

```
sudo firewall-cmd --permanent --add-service=ipsec
sudo firewall-cmd --reload
```

Step 5: Verify the VPN connection

To verify the IPSec VPN is working correctly, you can use the following command:

```
sudo ipsec status
```

This command will display the status of IPSec connections, including whether they are up and running.

Step 6: Configure the VPN client

On the client side, you will also need to configure an IPSec connection. If the client is another RHEL system, the configuration steps will be similar. For Windows or other operating systems, follow the appropriate steps to configure an IPSec VPN using the same shared secret and encryption settings.

Setting up an IPSec VPN on RHEL with Libreswan provides a secure method for remote communication. By carefully configuring the VPN server, defining strong encryption and hashing algorithms, and ensuring proper firewall configuration, you can create a robust and secure VPN that protects your network communications. Regular monitoring and maintenance will help ensure the VPN continues to operate securely and efficiently.

Recipe #90: Secure NFS services

Securing network services like NFS is crucial to protect sensitive data and prevent unauthorized access. In RHEL, you can secure these services by using a combination of firewall rules, SELinux policies, and encryption mechanisms like TLS. NFS allows remote hosts to mount filesystems over a network, which can expose data to unauthorized access if not properly secured.

Step 1: Install NFS server

Ensure that the NFS server packages are installed:

```
sudo dnf install nfs-utils
```

Step 2: Configure NFS exports

Edit the `/etc/exports` file to define the directories to be shared and their access controls:

```
sudo nano /etc/exports
```

Add a line for each directory you want to export, specifying the permissions:

```
/data/nfs_share 192.168.1.0/24(rw,sync,no_root_squash,no_subtree_check)
```

a. **rw**: Grants read-write access.

b. **sync**: Ensures changes are written to disk before the client is notified.

c. **no_root_squash**: Preserves root privileges for remote users (use with caution).

d. **no_subtree_check**: Disables subtree checking, which improves performance.

Step 3: Apply export configuration

After editing the `/etc/exports` file, apply the configuration:

```
sudo exportfs -rav
```

Step 4: Secure NFS with firewall

Open necessary ports in the firewall:

```
sudo firewall-cmd --permanent --add-service=nfs
sudo firewall-cmd --permanent --add-service=mountd
sudo firewall-cmd --permanent --add-service=rpc-bind
sudo firewall-cmd --reload
```

Step 5: Secure NFS with SELinux

Ensure SELinux is correctly configured to allow NFS access:

a. **Enable NFS home directory access**:

```
sudo setsebool -P nfs_export_all_rw 1
```

b. **Labeling files for NFS**:

Make sure that files in the NFS directory are correctly labelled:

```
sudo semanage fcontext -a -t nfs_t "/data/nfs_share(/.*)?"
sudo restorecon -Rv /data/nfs_share
```

Step 6: Use Kerberos for NFS authentication (optional)

For added security, configure NFS to use Kerberos for authentication:

1. **Install Kerberos packages**:

   ```
   sudo dnf install krb5-server krb5-workstation
   ```

2. **Configure Kerberos**:

   Edit the **`/etc/krb5.conf`** file and set up your Kerberos realm.

3. **Configure NFS to use Kerberos**:

   In the **`/etc/exports`** file, configure NFS with Kerberos:

   ```
   /data/nfs_share 192.168.1.0/24(rw,sync,sec=krb5p)
   ```

   Note: The `sec=krb5p` parameter that requires Kerberos for encryption and integrity.

Conclusion

This chapter explored various security hardening techniques for RHEL. We began by learning how to monitor and apply security updates using RHSA and how to automate the process of installing these updates. We learned to secure disk partitions with LUKS encryption, configure firewalls with **`firewalld`** and **`nftables`**, and enable FIPS mode for cryptographic compliance. Further, we examined the implementation and management of SELinux to enforce access controls, the configuration of OpenSSH for secure remote communication, and the application of SSL/TLS protocols to encrypt network traffic. We also covered setting up a secure VPN using IPSec and securing NFS services to protect shared files.

In the next chapter, we will discuss capacity planning, log analysis, and system audits, focusing on key practices for monitoring system performance, tracking user activities, and ensuring compliance with security standards. This includes setting baseline capacity requirements, configuring performance monitoring tools like Prometheus and **`sysstat`**, defining audit rules with **`auditctl`**, and analyzing logs for maintaining robust system security and efficiency.

Points to remember

- Regularly monitor and apply security updates to protect your system from known vulnerabilities: **`sudo dnf update --security`** and **`dnf updateinfo`**.
- Automate the installation of security updates to reduce the risk of unpatched systems: **`sudo systemctl enable --now dnf-automatic-install.timer`**.
- Use LUKS to encrypt critical disk partitions, ensuring secure data stored: **`sudo cryptsetup luksFormat /dev/sdX`**.
- Configure **`firewalld`** and **`nftables`** to manage and secure network traffic effectively: **`sudo firewall-cmd --permanent --add-service=http`**.
- Enable FIPS mode during installation for full cryptographic compliance: **`sudo fips-mode-setup --enable`**.
- Manage SELinux policies to enforce strict access controls and minimize the risk of unauthorized access: **`sudo setenforce 1`**.
- Configure OpenSSH and use system roles to secure remote communications and automate SSH client configurations: **`sudo systemctl restart sshd`**.
- Implement SSL/TLS on web servers to encrypt data transmitted over the network: **`sudo systemctl restart httpd`**.
- Set up an IPSec VPN to secure data in transit between networks: **`sudo systemctl enable --now ipsec`**.
- Secure NFS services by following best practices for access control and encryption: **`sudo exportfs -rav`**.

Chapter 15
Capacity Planning, Log Analysis, and System Audits

Introduction

RHEL provides robust tools and frameworks that allow system administrators to monitor, audit, and control the various aspects of system management, including capacity planning, performance monitoring, and auditing, to maintain our system integrity and security. This chapter delves into the essential practices and tools for monitoring system performance, tracking user activities, and ensuring compliance with security standards. Through practical examples and configurations, you will learn how to effectively manage and audit your RHEL environment, ensuring that your systems are secure and efficient.

Structure

In this chapter, we will cover the following topics:

- Set the baseline requirement for capacity
- Set up performance monitoring with sysstat
- Install Prometheus metrics and performance monitoring
- Plot and analyze monitoring data with Orca
- Set performance thresholds and alerting
- Set up and configure auditd

- Define rules and functions of the audit with audtctl
- Using pre-configured audit rules based on certification standards
- Monitor user login activity
- Monitor software installation and updates

Objectives

By the end of this chapter, you will be able to set and monitor baseline capacity requirements to ensure system efficiency, configure and use tools like auditd and auditctl for comprehensive auditing of system activities, implement performance monitoring using Prometheus and sysstat to maintain optimal system operations, monitor user login activities to detect unauthorized access attempts, track and audit software installation and update activities for compliance and security, and apply pre-configured audit rules based on certification standards to meet organizational and regulatory requirements.

Recipe #91: Set the baseline requirement for capacity

Effective capacity planning in IT infrastructure management is pivotal to ensuring systems can efficiently handle current and future workloads. Setting a baseline requirement for capacity in RHEL is a critical step in this process. The principles and methodologies described below establish a capacity baseline as a foundation for scaling, resource allocation, and performance optimization.

A capacity baseline is a set of metrics that define the minimum resource levels necessary to support a specific workload or service under normal operating conditions. It includes CPU, memory, disk I/O, and network bandwidth metrics. Establishing this baseline is crucial for identifying trends, planning for future growth, and preventing resource bottlenecks.

Steps to set the baseline requirement for capacity

1. **Identify critical workloads**: The first step is identifying which workloads are critical to your organization's operations. These could include databases, web servers, application servers, or any service that needs to be continuously available and performant.

2. **Monitor current resource utilization**: Utilize RHEL's native tools such as **`top`**, **`vmstat`**, **`iostat`**, and **`sar`** to gather data on current resource utilization. This data collection should cover various times and operational conditions to ensure a comprehensive understanding of the system's behavior.

    - **CPU usage**: Monitor the percentage of CPU utilization, load averages, and context switches.

- **Memory usage**: Track metrics like memory usage (both used and available), swap usage, and page faults.
- **Disk I/O**: Gather data on disk read/write rates, disk queue lengths, and disk latency.
- **Network utilization**: Measure network throughput, error rates, and packet drops.

3. **Establish historical baseline data**: Historical data provides insight into how resource utilization trends over time. Use the `sysstat` package to collect long-term data on resource usage. Tools like `rrdtool` can help visualize trends and identify periods of high demand or underutilization.
4. **Analyze peak usage patterns**: Identify the periods of peak usage and correlate them with specific business activities or events. Understanding these patterns will help in adjusting the baseline to accommodate spikes in demand without over-provisioning resources.
5. **Define acceptable performance metrics**: Establish performance thresholds that align with business objectives. For instance, you may define that CPU utilization should not exceed 70% under normal conditions or that disk I/O latency must stay below a certain number of milliseconds.
6. **Document the baseline**: Once the data is collected and analyzed, document the baseline requirements. This documentation should include all relevant metrics, acceptable thresholds, and any assumptions made during the process. It serves as a reference for future capacity planning and performance tuning.
7. **Implement monitoring and alerts**: Implement monitoring solutions to track these metrics in real-time, such as Red Hat Satellite, Prometheus, or Grafana. Set up alerts to notify administrators if resource usage exceeds the baseline, indicating potential performance issues.
8. **Continuous review and adjustment**: Capacity planning is not a one-time activity. Regularly review the baseline, considering new workloads, changes in business requirements, or observed performance trends. Adjust the baseline as necessary to ensure it remains relevant and effective.

Tools for capacity baseline setting in RHEL

- **Performance Co-Pilot (PCP)**: A framework and toolkit for monitoring and managing system performance.
- **Red Hat Insights**: Provides predictive analytics and real-time monitoring to help maintain performance baselines.
- **Tuned**: A RHEL utility that automatically optimizes system settings based on a predefined or custom profile.

Setting a baseline requirement for capacity in RHEL is a fundamental aspect of maintaining a robust and efficient IT infrastructure. By systematically monitoring, analyzing, and documenting resource utilization, you can ensure that your systems are always prepared to meet current demands and future growth. This proactive approach to capacity planning improves system performance and helps avoid costly downtimes and resource shortages. Regularly revisiting and adjusting your baseline ensures that it evolves with your organization's needs, optimizing your RHEL environment for the long haul.

Recipe #92: Set up performance monitoring with sysstat

Performance monitoring is critical to managing a RHEL environment, ensuring that systems run efficiently and that any potential issues are identified and resolved quickly. One of the most powerful and versatile tools for performance monitoring in RHEL is the **`sysstat`** package. You set up performance monitoring using **`sysstat`**, enabling you to systematically track and analyze system performance. Sysstat is a collection of utilities that monitor system performance and usage activity. It includes tools like **`sar`**, **`iostat`**, **`mpstat`**, and others, which can collect and report various system metrics such as CPU usage, memory utilization, I/O operations, and network activity. These tools are essential for system administrators who need to keep a close eye on system performance.

Installing Sysstat

Before you can begin using **`sysstat`** for performance monitoring, you need to ensure that it is installed on your RHEL system.

1. **Install the Sysstat package**: The **`sysstat`** package can be installed using the DNF package manager.

   ```
   sudo dnf install sysstat
   ```

   This command installs the entire suite of **sysstat** tools on your system.

2. **Enable and start the Sysstat service**: After installation, the **`sysstat`** service must be enabled and started to begin collecting performance data.

   ```
   sudo systemctl enable sysstat
   sudo systemctl start sysstat
   ```

The **`sysstat`** service will now start collecting data at regular intervals, which is configured by default in the **`/etc/cron.d/sysstat`** file.

Configuring Sysstat

Sysstat can be configured to collect data at intervals that suit your monitoring needs. The

1. **Edit the Sysstat configuration**: The configuration file is located at **`/etc/sysconfig/sysstat`**. You can modify this file to adjust the collection interval and other settings.

   ```
   sudo vi /etc/sysconfig/sysstat
   ```

   Key parameters you might want to adjust include:

   - **`HISTORY`**: The number of days to retain collected data.
   - **`INTERVAL`**: The interval, in minutes, at which data is collected.

2. **Configure data retention**: By default, **`sysstat`** retains data for 7 days. You can increase this period by adjusting the `HISTORY` parameter in the configuration file.

   ```
   HISTORY=30
   ```

   This change will retain the performance data for 30 days.

3. **Restart the Sysstat service**: After making configuration changes, restart the **`sysstat`** service to apply them.

   ```
   sudo systemctl restart sysstat
   ```

Using Sysstat tools for performance monitoring

Once **`sysstat`** is installed and configured, you can begin using its tools to monitor various aspects of system performance.

1. **Monitoring CPU usage with sar**: The **`sar`** command collects, reports, and saves CPU usage data.

   ```
   sar -u 1 5
   ```

   This command reports CPU usage every second for 5 seconds. You can also view historical data by simply running **`sar`** with the appropriate options.

2. **Monitoring disk I/O with iostat**: The **`iostat`** command monitors and reports on CPU utilization and I/O statistics for devices and partitions.

   ```
   iostat -dx 5 3
   ```

   This command provides detailed I/O statistics every 5 seconds, three times.

3. **Monitoring memory usage with free**: While free is not part of **`sysstat`**, it is often used in conjunction with **`sar`** to monitor memory usage.

   ```
   free -m
   ```

   This command provides an overview of memory usage in megabytes.

4. **Monitoring network performance with sar**: **`sar`** can also monitor network interfaces, displaying metrics such as data transmitted, received, and error rates.

   ```
   sar -n DEV 1 3
   ```

This command shows network statistics for each interface every second for three iterations.

5. **Creating reports**: Sysstat's tools allow you to create detailed reports for historical analysis. You can generate reports from the collected data using the `sar` command with specific options to filter the data you are interested in.

   ```
   sar -A > /tmp/system_performance_report.txt
   ```

   This command generates a comprehensive report of all collected metrics and saves it to a file.

Automating and scheduling reports

To ensure continuous monitoring, you can automate the generation of performance reports using `cron` jobs. For example, you can schedule a daily report to be generated at midnight:

1. **Create a Cron job**: Edit the `crontab` for the root user or a monitoring user:

   ```
   sudo crontab -e
   ```

Add the following line to generate a report at midnight every day:

```
0 0 * * * /usr/bin/sar -A > /var/log/sysstat/daily_report_$(date +\%Y\%m\%d).
txt
```

This command will create a daily report with a timestamp in the filename, making it easy to archive and review historical data.

Setting up performance monitoring with `sysstat` in RHEL is an essential task for maintaining system health and performance. By effectively utilizing the `sysstat` suite, administrators can gain deep insights into system behavior, identify potential bottlenecks, and ensure that their systems are operating optimally. Regular monitoring, combined with the automated reporting capabilities of `sysstat`, helps in proactive management and efficient capacity planning.

Recipe # 93: Install Prometheus metrics and performance monitoring

Prometheus is a powerful open-source monitoring and alerting toolkit originally built at SoundCloud. It is particularly suited for monitoring dynamic cloud environments and works well for monitoring metrics and performance in RHEL environments. Let us install Prometheus on RHEL and set it up for basic metrics and performance monitoring.

Step 1: Install required dependencies

Before installing Prometheus, ensure that your system is updated and that all required dependencies are installed.

1. **Update the system**:

   ```
   sudo dnf update -y
   ```

2. **Install wget (used to download Prometheus binaries)**:

   ```
   sudo dnf install wget -y
   ```

Step 2: Download and install Prometheus

Prometheus does not come as a package in the default RHEL repositories, so you will need to download the binaries directly from the Prometheus website.

1. **Download Prometheus**: Navigate to the Prometheus download page and download the latest version of Prometheus. Alternatively, you can use the following command to download the binary:

   ```
   wget https://github.com/prometheus/prometheus/releases/download/
   v2.54.1/prometheus-2.54.1.linux-amd64.tar.gz
   ```

2. **Extract the downloaded archive**:

   ```
   tar xvf prometheus-2.54.1.linux-amd64.tar.gz
   ```

   This will extract the files into a directory named **`prometheus-2.54.1.linux-amd64`**.

3. **Move Prometheus binaries**: Move the Prometheus binaries to **`/usr/local/bin`**:

   ```
   sudo mv prometheus-2.54.1.linux-amd64/prometheus /usr/local/bin/
   sudo mv prometheus-2.54.1.linux-amd64/promtool /usr/local/bin/
   ```

4. **Move configuration files**: Move the configuration files to **`/etc/prometheus`**:

   ```
   sudo mkdir /etc/prometheus
   sudo mv prometheus-2.54.1.linux-amd64/prometheus.yml /etc/
   prometheus/
   sudo mv prometheus-2.54.1.linux-amd64/consoles /etc/prometheus/
   sudo mv prometheus-2.54.1.linux-amd64/console_libraries /etc/
   prometheus/
   ```

Step 3: Create a Prometheus system user

For security reasons, it is a good practice to run Prometheus under a dedicated system user.

1. **Create the user**:

   ```
   sudo useradd --no-create-home --shell /bin/false Prometheus
   ```

2. **Set permissions**: Change ownership of the Prometheus directories to the Prometheus user:

   ```
   sudo chown -R prometheus:prometheus /etc/prometheus
   ```

```
sudo chown prometheus:prometheus /usr/local/bin/prometheus
sudo chown prometheus:prometheus /usr/local/bin/promtool
```

Step 4: Create a systemd service file for Prometheus

To manage Prometheus as a service, you need to create a **`systemd`** service file.

1. **Create the service file**:
   ```
   sudo vi /etc/systemd/system/prometheus.service
   ```
2. **Add the following service configuration**:
   ```
   [Unit]
   Description=Prometheus
   Wants=network-online.target
   After=network-online.target
   [Service]
   User=prometheus
   Group=prometheus
   Type=simple
   ExecStart=/usr/local/bin/prometheus \
     --config.file /etc/prometheus/prometheus.yml \
     --storage.tsdb.path /var/lib/prometheus/ \
     --web.console.templates=/etc/prometheus/consoles \
     --web.console.libraries=/etc/prometheus/console_libraries
   [Install]
   WantedBy=multi-user.target
   ```
3. **Reload systemd to apply the new service**:
   ```
   sudo systemctl daemon-reload
   ```
4. **Enable and start Prometheus**:
   ```
   sudo systemctl enable prometheus
   sudo systemctl start prometheus
   ```
5. **Check the service status**: Verify that Prometheus is running:
   ```
   sudo systemctl status Prometheus
   ```

Step 5: Access the Prometheus web interface

Prometheus runs on port 9090 by default. To access the web interface:

1. **Open a browser**: Navigate to **`http://<your-server-ip>:9090`** in your web browser.
2. **Verify Prometheus operation**: You should see the Prometheus web UI, which allows you to explore metrics and configure alerts.

Step 6: Configure Prometheus for system monitoring

Prometheus uses a configuration file (`prometheus.yml`) to define what it should monitor.

1. **Edit the configuration file**: The default configuration is sufficient for basic monitoring, but you can edit `/etc/prometheus/prometheus.yml` to add more jobs or targets.

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s
scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9090']
```

2. **Reload Prometheus** when you make changes to the configuration:

```
sudo systemctl reload Prometheus
```

Step 7: (Optional) Set up Node Exporter

For more detailed system-level metrics, you can set up Node Exporter.

1. **Download and install Node Exporter**:

```
wget https://github.com/prometheus/node_exporter/releases/download/
v1.8.2/node_exporter-1.8.2.linux-amd64.tar.gz
tar xvf node_exporter-1.8.2.linux-amd64.tar.gz
sudo mv node_exporter-1.8.2.linux-amd64/node_exporter /usr/local/
bin/
```

2. **Create a Systemd service for Node Exporter**:

```
sudo vi /etc/systemd/system/node_exporter.service
```

3. **Add the service configuration**:

```
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target
[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter
[Install]
WantedBy=multi-user.target
```

4. **Enable and start Node Exporter**:

   ```
   sudo systemctl daemon-reload
   sudo systemctl enable node_exporter
   sudo systemctl start node_exporter
   ```

5. **Add Node Exporter to Prometheus targets**: `edit/etc/prometheus/prometheus.yml` to include it as a scrape target.

   ```
   - job_name: 'node_exporter'
     static_configs:
       - targets: ['localhost:9100']
   ```

6. **Reload Prometheus**:

   ```
   sudo systemctl reload prometheus
   ```

You installed Prometheus on RHEL and set it up to monitor system metrics. Prometheus and Node Exporter provide a powerful platform for monitoring system performance, alerting on issues, and analyzing trends over time. You can further extend this setup by adding additional exporters and integrating with Grafana for enhanced visualization capabilities.

Recipe # 94: Plot and analyze monitoring data with Orca

Orca is a command-line tool developed by the Prometheus project for exporting time series data from Prometheus in a format that can be used for plotting and analysis, typically within Jupyter Notebooks or other environments that can consume CSV or JSON data. While Prometheus itself handles the monitoring and data collection, Orca is useful for visualizing and analyzing this data outside of Prometheus' built-in web UI.

However, it is important to note that Orca is not a native plotting tool. Instead, it acts as an exporter that helps you retrieve data from Prometheus, so you can use Python libraries like Matplotlib, Plotly, or Pandas to plot and analyze the data.

Below is a step-by-step guide to exporting monitoring data from Prometheus using Orca and then plotting and analyzing it with Python libraries.

Step 1: Install Orca

Orca can be installed using `pip`:

```
pip install orca
```

Step 2: Query Prometheus with Orca

You can use Orca to query Prometheus for specific metrics. First, make sure your Prometheus server is running and accessible. Then, you can use Orca to export data to CSV or JSON.

Here is an example of how to query Prometheus for CPU usage data:

```
orca query \
  --server http://localhost:9090 \
  --start '2024-09-01T00:00:00Z' \
  --end '2024-09-01T01:00:00Z' \
  --step 60s \
  'rate(node_cpu_seconds_total{mode="idle"}[5m])' > cpu_usage.csv
```

This command retrieves CPU usage data for the specified time range and step interval and saves it as **`cpu_usage.csv`**.

Step 3: Load and analyze the data in Python

Once you have the data exported as a CSV file, you can load and analyze it using Python libraries like Pandas and Matplotlib.

1. **Load the data**:

   ```
   import pandas as pd
   # Load the CSV file into a DataFrame
   df = pd.read_csv('cpu_usage.csv')
   # Display the first few rows of the DataFrame
   print(df.head())
   ```

2. **Plot the data**:

   Using Matplotlib or another plotting library, you can visualize the data.

   ```
   import matplotlib.pyplot as plt
   # Assuming the DataFrame has columns 'time' and 'value'
   df['time'] = pd.to_datetime(df['time'])
   plt.figure(figsize=(10, 6))
   plt.plot(df['time'], df['value'], label='CPU Usage')
   plt.xlabel('Time')
   plt.ylabel('CPU Usage (%)')
   plt.title('CPU Usage Over Time')
   plt.legend()
   plt.grid(True)
   plt.show()
   ```

3. **Analyze the data**:

   You can perform various analyses on the data, such as calculating the average CPU usage, identifying trends, or detecting anomalies.

   ```
   avg_cpu_usage = df['value'].mean()
   print(f'Average CPU Usage: {avg_cpu_usage:.2f}%')
   # Detecting spikes in CPU usage
   ```

```
spikes = df[df['value'] > avg_cpu_usage + 2 * df['value'].std()]
print(f'Number of spikes detected: {len(spikes)}')
```

Step 4: Advanced visualization with Plotly

For more interactive and advanced visualizations, you can use Plotly:

```
import plotly.express as px
fig = px.line(df, x='time', y='value', title='CPU Usage Over Time')
fig.update_xaxes(title_text='Time')
fig.update_yaxes(title_text='CPU Usage (%)')
fig.show()
```

By using Orca to export monitoring data from Prometheus and then leveraging Python's rich ecosystem of data analysis and visualization libraries, you can perform in-depth analysis and create insightful visualizations of your system's performance. This approach is handy for creating custom reports or dashboards beyond what is available in the standard Prometheus UI.

Recipe #95: Set performance thresholds and alerting

Setting performance thresholds and configuring alerting mechanisms are essential tasks for maintaining the health and stability of your RHEL systems. With Prometheus and Alertmanager, you can define performance thresholds and automatically trigger alerts when these thresholds are crossed. These are the step-by-step guide to set up performance thresholds and configure alerting:

Step 1: Define performance thresholds in Prometheus

Performance thresholds are typically defined within Prometheus as alerting rules. These rules specify the conditions under which an alert should be triggered based on the metrics collected by Prometheus.

1. **Create alerting rules file**: First, create a new file for your alerting rules, typically stored in **`/etc/prometheus/alert.rules.yml`**.

   ```
   sudo vi /etc/prometheus/alert.rules.yml
   ```

2. **Define alerting rules**: In this file, you can define your alerting rules. Below is an example of how to define alerts for CPU usage and memory utilization:

   ```
   groups:
   - name: system_alerts
     rules:
     - alert: HighCPUUsage
       expr: 100 - (avg by (instance) (irate(node_cpu_seconds_
   ```

```
total{mode="idle"}[5m])) * 100) > 80
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "High CPU usage detected on instance {{ $labels.
instance }}"
      description: "CPU usage is above 80% for more than 5 minutes."
  - alert: HighMemoryUsage
    expr: (node_memory_MemAvailable_bytes / node_memory_MemTotal_
bytes) * 100 < 20
    for: 5m
    labels:
      severity: critical
    annotations:
      summary: "High Memory usage detected on instance {{ $labels.
instance }}"
      description: "Available memory is less than 20% for more than
5 minutes."
```

In this example:

- The **HighCPUUsage** alert triggers when CPU usage exceeds 80% for more than 5 minutes.
- The **HighMemoryUsage** alert triggers when available memory drops below 20% for over 5 minutes.

3. **Configure Prometheus to load the alerting rules**: You need to modify the Prometheus configuration file (**/etc/prometheus/prometheus.yml**) to include the alerting rules file.

```
rule_files:
  - "/etc/prometheus/alert.rules.yml"
```

4. **Reload Prometheus**: After setting up the alerting rules, reload Prometheus to apply the changes.

```
sudo systemctl reload Prometheus
```

Step 2: Install and configure Alertmanager

Alertmanager handles the alerts generated by Prometheus, including deduplication, grouping, and routing of alerts to the correct receivers (e.g., email, Slack, PagerDuty).

1. **Download and install Alertmanager**: Download the latest version of Alertmanager:

```
wget https://github.com/prometheus/alertmanager/releases/download/
```

```
v0.27.0/alertmanager-0.27.0.linux-amd64.tar.gz
tar xvf alertmanager-0.27.0.linux-amd64.tar.gz
sudo mv alertmanager-0.27.0.linux-amd64/alertmanager /usr/local/bin/
sudo mv alertmanager-0.27.0.linux-amd64/amtool /usr/local/bin/
```

2. **Create Alertmanager configuration**: Create the Alertmanager configuration file:

   ```
   sudo vi /etc/alertmanager/alertmanager.yml
   ```

   Add the following configuration as an example to send alerts via email:

   ```
   global:
     smtp_smarthost: 'smtp.example.com:587'
     smtp_from: 'alertmanager@example.com'
     smtp_auth_username: 'alertmanager@example.com'
     smtp_auth_password: 'yourpassword'
   route:
     receiver: 'email-alert'
   receivers:
   - name: 'email-alert'
     email_configs:
     - to: 'your-email@example.com'
   ```

   Replace the placeholders with your actual SMTP server details and recipient email.

3. **Create a Systemd service for Alertmanager**: Similar to Prometheus, you can manage Alertmanager as a service.

   ```
   sudo vi /etc/systemd/system/alertmanager.service
   ```

   Add the following configuration:

   ```
   [Unit]
   Description=Alertmanager
   Wants=network-online.target
   After=network-online.target
   [Service]
   User=alertmanager
   Group=alertmanager
   Type=simple
   ExecStart=/usr/local/bin/alertmanager \
     --config.file=/etc/alertmanager/alertmanager.yml \
     --storage.path=/var/lib/alertmanager
   [Install]
   WantedBy=multi-user.target
   ```

4. **Start and enable Alertmanager**:

```
sudo systemctl daemon-reload
sudo systemctl start alertmanager
sudo systemctl enable alertmanager
```

5. **Verify Alertmanager status**: Check that Alertmanager is running correctly:

```
sudo systemctl status alertmanager
```

Step 3: Integrate Prometheus with Alertmanager

To ensure that Prometheus can send alerts to Alertmanager, you need to configure Prometheus accordingly.

1. **Update Prometheus configuration**: Edit **`/etc/prometheus/prometheus.yml`** to include Alertmanager details:

```
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - localhost:9093
```

2. **Reload Prometheus**:

```
sudo systemctl reload Prometheus
```

Step 4: Test your alerts

To test whether your alerts are correctly set up, you can manually trigger an alert.

1. **Simulate high CPU usage**: You can simulate high CPU usage by running a CPU-intensive task on your system, such as:

```
stress --cpu 4 --timeout 600
```

   This command stresses 4 CPU cores for 10 minutes.

2. **Check Prometheus and Alertmanager**:

   - Visit the Prometheus UI (**`http://<your-server-ip>:9090`**) and go to the **Alerts** section to see if your alert is firing.
   - Visit the Alertmanager UI (**`http://<your-server-ip>:9093`**) to see if the alert has been received and processed.

Setting performance thresholds and configuring alerting with Prometheus and Alertmanager on RHEL ensures that your systems remain healthy and issues are addressed promptly. This highly customizable setup allows you to define thresholds that align with your specific needs and integrate with various notification systems. Regularly review and update your alerting rules to ensure they remain effective as your infrastructure evolves.

Recipe # 96: Set up and configure auditd

The **`auditd`** daemon is a crucial component for maintaining the security and integrity of a RHEL system. It provides robust logging of security-relevant events, helping administrators track unauthorized access and other suspicious activities. This guide will walk you through the steps to set up and configure **`auditd`** on RHEL.

Step 1: Install auditd

In most cases, **`auditd`** is installed by default on RHEL systems. However, if it is not installed, you can install it using the following command:

```
sudo dnf install audit
```

After installation, ensure that the **`auditd`** service is enabled and started:

```
sudo systemctl enable auditd

sudo systemctl start auditd
```

Step 2: Configure auditd

The main configuration file for **`auditd`** is located at **`/etc/audit/auditd.conf`**. This file controls the behavior of the audit daemon, including log file locations, maximum log sizes, and other parameters.

1. **Open the configuration file**:

   ```
   sudo vi /etc/audit/auditd.conf
   ```

2. **Key configuration options**:

   - **log_file**: Specifies the location of the audit log file. By default, it is set to **`/var/log/audit/audit.log`**.
   - **log_format**: Specifies the format of the log file. Options include **`RAW`**, **`NOLOG`**, **`ENRICHED`**. The default is **`ENRICHED`**.
   - **max_log_file**: Sets the maximum size of the log file in megabytes before it is rotated. The default is 8 MB.
   - **num_logs**: Specifies the number of rotated logs to keep. The default is **`5`**.
   - **space_left_action**: Defines the action to take when free space on the disk reaches a certain threshold. Options include **`IGNORE`**, **`SYSLOG`**, **`SUSPEND`**, **`HALT`**.
   - **admin_space_left_action**: Defines the action to take when free space reaches the admin-defined threshold.
   - **max_log_file_action**: Specifies the action to take when the maximum log file size is reached. Options include **`IGNORE`**, **`SYSLOG`**, **`ROTATE`**, **`SUSPEND`**, **`HALT`**.

Example configuration:

```
log_file = /var/log/audit/audit.log
log_format = ENRICHED
max_log_file = 20
num_logs = 10
space_left_action = SYSLOG
admin_space_left_action = SUSPEND
max_log_file_action = ROTATE
```

3. **Save and exit**: After making the necessary changes, save the file and exit the editor.

Step 3: Configure audit rules

Audit rules define what events should be logged by **`auditd`**. These rules are usually specified in **`/etc/audit/rules.d/ or /etc/audit/audit.rules`**.

1. **Create or edit audit rules**: You can create custom audit rules by editing files in **`/etc/audit/rules.d/`**. For example, you might create a file called **`audit.rules`**:

   ```
   sudo vi /etc/audit/rules.d/audit.rules
   ```

2. **Common audit rules**: Here are some examples of common audit rules:

   - **Monitor changes to /etc/passwd**:

     ```
     -w /etc/passwd -p wa -k passwd_changes
     ```

   - **Monitor execution of specific commands (e.g., chmod)**:

     ```
     -a always,exit -F arch=b64 -S chmod -S fchmod -S fchmodat -k
     perm_mod
     ```

   - **Monitor user logins and logouts**:

     ```
     -w /var/log/wtmp -p wa -k logins
     ```

   - **Monitor attempts to access /etc/shadow**:

     ```
     -w /etc/shadow -p r -k shadow_access
     ```

3. Each rule is composed of:

   - **`-w`** specifies the file or directory to watch.
   - **`-p`** specifies the permissions to watch for (**"r"** for read, **"w"** for write, **"x"** for execute, **"a"** for attribute changes).
   - **`-k`** specifies a key name to help identify and filter audit logs.
   - **`-a`** defines the action and list for the rule, often paired with system call filters.

4. **Load audit rules**: After saving the rules, load them into the running **`auditd`** service:

```
sudo augenrules –load
```

Alternatively, you can restart **auditd** to apply the rules:

```
sudo systemctl restart auditd
```

Step 4: Verify and monitor audit logs

Once **auditd** is configured and running, you can verify its operation and monitor the audit logs.

1. **Check auditd status**: Ensure that **auditd** is running without issues:

   ```
   sudo systemctl status auditd
   ```

2. **View audit logs**: Audit logs are stored in **/var/log/audit/audit.log**. You can view these logs using less or **ausearch**:

   ```
   sudo less /var/log/audit/audit.log
   ```

   To search for specific audit records using **ausearch**:

   ```
   sudo ausearch -k passwd_changes
   ```

   This command searches for all audit records with the key **passwd_changes**.

3. **Generate audit reports**: Use the **aureport** tool to generate summary reports from the audit logs:

   ```
   sudo aureport –summary
   ```

This command generates a summary report of the audit logs, including information on logins, file access, and system call activity.

Step 5: Implement additional security measures

Beyond basic configuration, you can further enhance the security of **auditd** by:

1. **Configuring email alerts**: Configure email alerts for specific audit events by using scripts that parse audit.log and send notifications. This is done by integrating **auditd** with tools like **auditd-plugins** or custom scripts.

2. **Integrating with a SIEM system**: For large-scale deployments, consider integrating **auditd** logs with a **security information and event management** (**SIEM**) system to centralize and analyze logs from multiple sources.

3. **Ensure audit logs are immutable**: Protect audit logs from being tampered with by making them immutable:

   ```
   sudo chattr +i /var/log/audit/audit.log
   ```

This command prevents any changes to the audit logs, ensuring that logs cannot be deleted or altered.

Setting up and configuring **`auditd`** on RHEL provides a powerful mechanism for monitoring and logging security-relevant events. By carefully crafting audit rules and ensuring proper log management, you can maintain a high level of security and accountability on your systems. Regularly review audit logs and adjust your audit rules as needed to adapt to changing security requirements and operational contexts.

Recipe # 97: Define rules and functions of the audit with audtctl

`auditctl` is a command-line utility used to control and configure the Linux kernel's audit system. It allows you to define rules and manage the behavior of **`auditd`**, the audit daemon. This utility is particularly powerful for setting up audit rules dynamically, without needing to restart **`auditd`**. Below is a detailed guide on how to define rules and functions using **`auditctl`**:

Basic syntax of auditctl

The general syntax for using **`auditctl`** is:

```
auditctl [options] [rules]
```

Types of audit rules

Audit rules can be broadly categorized into four types:

- **File/Directory watch rules**: Monitor access to specific files or directories.
- **System call rules**: Monitor specific system calls.
- **User/Group rules**: Monitor activities based on user or group IDs.
- **Keyed rules**: Rules that include a key for easier identification in the audit logs.

1. **File/Directory watch rules**

   These rules are used to monitor access to specific files or directories.

   Example: Monitor changes to **`/etc/passwd`**.

   ```
   auditctl -w /etc/passwd -p wa -k passwd_changes
   ```

   - **`-w /etc/passwd`**: Watch the file **`/etc/passwd`**.
   - **`-p wa`**: Monitor write (w) and attribute change (a) permissions.
   - **`-k passwd_changes`**: Assign a key named **`passwd_changes`** to this rule for easy identification in logs.

2. **System call rules**

   System call rules monitor specific system calls made by processes. These rules are more complex and allow fine-grained monitoring of system behavior.

Example: Monitor all **chmod** system calls.

```
auditctl -a always,exit -F arch=b64 -S chmod -S fchmod -S fchmodat -k
permission_change
```

- **-a always,exit**: Adds a rule to the audit system for both success and failure of the **syscall** exit.
- **-F arch=b64**: Filter by architecture (b64 for 64-bit systems).
- **-S chmod -S fchmod -S fchmodat**: Monitors the **chmod**, **fchmod**, and **fchmodat** system calls.
- **-k permission_change**: Assigns a key named **permission_change** for easy identification.

3. **User/Group rules**

   These rules monitor activities based on user or group IDs.

   Example: Monitor all commands executed by a specific user (user1).

   ```
   auditctl -a always,exit -F arch=b64 -F auid=$(id -u user1) -S execve
   -k user_commands
   ```

   - **-F auid=$(id -u user1):** Filter based on the audit user ID (**auid**) of **user1**.
   - **-S execve**: Monitor the **execve** system call, which is used to execute programs.
   - **-k user_commands**: Assign a key named **user_commands**.

4. **Keyed rules**

   Keyed rules are those that include a **-k** option to assign a key to the rule. This key can be used later to filter audit logs and reports.

   Example: Monitoring access to the **/var/log/secure** file.

   ```
   auditctl -w /var/log/secure -p rwxa -k secure_log_access
   ```

   - **-w /var/log/secure**: Watch the **/var/log/secure** file.
   - **-p rwxa**: Monitor read (r), write (w), execute (x), and attribute changes (a).
   - **-k secure_log_access**: Assigns a key named **secure_log_access**.

Functions of auditctl

auditctl provides several functionalities for managing audit rules and the audit system:

1. **Add rules**: You can add audit rules dynamically using the **auditctl** command. These rules immediately take effect without needing to restart **auditd**.
2. **List rules**:
   - List all active audit rules using:

```
auditctl -l
```

This displays all currently loaded rules in the audit system.

- **Delete rules**: Remove all audit rules:

  ```
  auditctl -D
  ```

  This command deletes all audit rules. You can also delete specific rules by providing the rule details.

3. **Enable/Disable auditing**:

   a. Enable auditing:

   ```
   auditctl -e 1
   ```

   b. Disable auditing:

   ```
   auditctl -e 0
   ```

   c. Check the status of auditing:

   ```
   auditctl -s
   ```

   This shows the current status of the audit system, including whether auditing is enabled, the number of active rules, and other key metrics.

4. **View audit status**:

   a. The **`auditctl -s`** command displays various statistics about the audit system, including:

      i. **enabled**: Whether auditing is enabled.

      ii. **failure**: The failure mode (what happens if **`auditd`** fails).

      iii. **pid**: The process ID of the **`auditd`** daemon.

      iv. **rate_limit**: The rate limit for audit messages.

      v. **backlog_limit**: The maximum number of audit messages that can be queued before they are discarded.

5. **Control logging**:

   a. Adjust the rate at which audit messages are logged:

   ```
   auditctl -r 100
   ```

   This sets the audit message rate limit to 100 messages per second.

6. **Control backlog**:

   a. Set the backlog limit:

   ```
   auditctl -b 8192
   ```

This sets the maximum number of audit messages in the backlog queue to **`8192`**. Using **`auditctl`**, administrators can define and manage audit rules in real time, providing granular control over which events are monitored on a RHEL system. The ability to dynamically adjust these rules makes **`auditctl`** a powerful tool for maintaining system security and ensuring compliance with organizational or regulatory requirements. By carefully defining audit rules, you can monitor critical files, system calls, and user activities and be alerted to any potentially suspicious behavior, thereby enhancing the overall security posture of your systems.

Recipe #98: Using pre-configured audit rules based on certification standards

When configuring an audit system in RHEL, it is often beneficial to use pre-configured audit rules that are designed to meet specific security certification standards, such as PCI-DSS, HIPAA, or CIS benchmarks. These pre-configured rules ensure that your system's auditing aligns with industry best practices and regulatory requirements. RHEL provides several profiles and pre-configured audit rulesets that can be used to quickly set up an audit system that is compliant with these standards. This is how to apply these pre-configured audit rules:

Step 1: Install the SCAP Security Guide

The SCAP Security Guide is a useful tool that provides pre-configured audit rules and security profiles that can be applied to your RHEL system to comply with various standards. To use it, you need to install the **`scap-security-guide`** package.

```
sudo dnf install scap-security-guide
```

Step 2: Apply a pre-configured audit ruleset

The **`scap-security-guide`** package provides several audit rules that are aligned with different security standards. You can find these rules in the **`/usr/share/audit/sample-rules/`** directory or apply them directly using the OpenSCAP tools.

1. **Identify the desired audit profile**: The **`scap-security-guide`** includes profiles for various standards, such as PCI-DSS, CIS, DISA STIG, etc. You can use the **`oscap`** tool to list these profiles.

   ```
   oscap info /usr/share/xml/scap/ssg/content/ssg-rhel9-ds.xml
   ```

   This command lists the available profiles and their descriptions.

2. **Apply the audit rules**: You can directly apply audit rules from a specific profile using **`oscap`**:

   Example: Apply the CIS benchmark audit rules:

   ```
   sudo oscap xccdf eval --profile xccdf_org.ssgproject.content_profile_
   ```

3. This command applies the CIS benchmark ruleset to your RHEL system.

4. **Manual application**: If you prefer to apply audit rules manually, you can copy the relevant ruleset from the **`/usr/share/audit/sample-rules/`** directory to your **`/etc/audit/rules.d/`** directory.

   Example: Apply the PCI-DSS audit rules manually:

   ```
   sudo cp /usr/share/audit/sample-rules/30-pci-dss-v31.rules /etc/
   audit/rules.d/
   sudo augenrules –load
   ```

This command copies the PCI-DSS rules to the audit rules directory and loads them into the **`auditd`** service.

Step 3: Verify and customize the audit rules

After applying the pre-configured audit rules, it is important to verify that they are correctly loaded and active.

1. **List active audit rules**:

   ```
   sudo auditctl -l
   ```

   This command lists all the active audit rules currently loaded in the audit system.

2. **Customize rules**: While pre-configured rules provide a solid foundation, you may need to customize them to fit your organization's specific needs better. You can edit the rules in **`/etc/audit/rules.d/`** and then reload them using **`augenrules`**:

   ```
   sudo vi /etc/audit/rules.d/30-pci-dss-v31.rules
   sudo augenrules --load
   ```

Ensure that any customizations you make do not conflict with the compliance requirements of the specific standard you are adhering to.

Step 4: Continuous monitoring and reporting

Once the audit rules are in place, it is important to continuously monitor and generate reports to ensure ongoing compliance.

1. **Use aureport for summary reports**:

   ```
   sudo aureport –summary
   ```

   This command generates a summary of the audit logs, which can be useful for regular compliance checks.

2. **Automate report generation**: Consider automating the generation of audit reports using **`cron`** jobs, so compliance reports are regularly available for review.

   Example: Create a daily summary report:

   ```
   echo "0 0 * * * root /usr/sbin/aureport --summary > /var/log/audit/
   ```

Step 5: Integrate with a SIEM or log management solution

To further enhance your audit logging capabilities, you may want to integrate **`auditd`** with a SIEM or a centralized log management solution.

1. **Forward audit logs**: Forwarding audit logs to a centralized log server ensures that logs are preserved and available for analysis even if the local system is compromised.

   Example: Use **`rsyslog`** to forward logs:

   ```
   sudo vi /etc/rsyslog.d/audit.conf
   Add the following lines:
   $ModLoad imfile
   $InputFileName /var/log/audit/audit.log
   $InputFileTag audit:
   $InputFileStateFile audit-state
   $InputFileSeverity info
   $InputFileFacility local7
   $InputRunFileMonitor
   *.* @logserver.example.com:514
   ```

   Then restart **`rsyslog`**:

   ```
   sudo systemctl restart rsyslog
   ```

2. **Alerting and notification**: Configure alerts in your SIEM or log management tool to notify administrators when critical audit events occur, ensuring prompt response to potential security incidents.

Using pre-configured audit rules based on certification standards such as PCI-DSS, HIPAA, or CIS benchmarks simplifies the process of setting up a compliant audit system in RHEL. By leveraging tools like **`scap-security-guide`** and **`oscap`**, you can quickly apply and manage these rules, ensuring that your system adheres to necessary security standards. Regular monitoring, customization, and integration with broader security tools further enhance your system's audit capabilities, helping to maintain compliance and protect against security threats.

Recipe #99: Monitor user login activity

Monitoring user login activity on a RHEL system ensures security and compliance. The **`auditd`** daemon, alongside other utilities like **`last`**, **`w`**, and **`ausearch`**, can track and monitor user login activity effectively—this step-by-step setup and using these tools to monitor user logins on your RHEL system.

Step 1: Configure audit rules for user login activity

To monitor user login activity using `auditd`, you need to define audit rules that log relevant events such as successful logins, failed login attempts, and session openings.

1. **Create audit rules for user logins**:

   Add the following rules to monitor login events. These rules can be added to the `/etc/audit/rules.d/audit.rules` file or directly using `auditctl`.

   Monitor the /var/log/secure file: The `/var/log/secure` file logs all login attempts, both successful and failed:

   ```
   auditctl -w /var/log/secure -p wa -k logins
   ```

   Monitor session opening (PAM): This rule monitors **pluggable authentication modules** (**PAM**) events related to session management, including user logins:

   ```
   auditctl -a always,exit -F arch=b64 -S execve -C uid!=euid -F euid=0
   -k pam
   ```

   Monitor useradd, usermod, and userdel commands: These commands are used to add, modify, and delete users on the system:

   ```
   auditctl -w /usr/sbin/useradd -p x -k user_mgmt
   auditctl -w /usr/sbin/usermod -p x -k user_mgmt
   auditctl -w /usr/sbin/userdel -p x -k user_mgmt
   ```

2. **Persist the audit rules**:

   If you used `auditctl` to set up these rules, you should add them to `/etc/audit/rules.d/audit.rules` to make them persistent across reboots:

   ```
   echo "-w /var/log/secure -p wa -k logins" | sudo tee -a /etc/audit/
   rules.d/audit.rules
   echo "-a always,exit -F arch=b64 -S execve -C uid!=euid -F euid=0 -k
   pam" | sudo tee -a /etc/audit/rules.d/audit.rules
   echo "-w /usr/sbin/useradd -p x -k user_mgmt" | sudo tee -a /etc/
   audit/rules.d/audit.rules
   echo "-w /usr/sbin/usermod -p x -k user_mgmt" | sudo tee -a /etc/
   audit/rules.d/audit.rules
   echo "-w /usr/sbin/userdel -p x -k user_mgmt" | sudo tee -a /etc/
   audit/rules.d/audit.rules
   ```

   Then, load the rules:

   ```
   sudo augenrules –load
   ```

Step 2: Monitor login activity using ausearch

The `ausearch` command is used to search the audit logs for specific events. You can use it to find login-related activities.

1. **Search for login events**:

To search for user login events based on the key you defined (logins):

```
sudo ausearch -k logins
```

2. **Search for specific user logins**:

   You can filter the search by a specific user:

   ```
   sudo ausearch -ua <username>
   ```

   Replace **`<username>`** with the actual username you want to investigate.

3. **Generate a detailed report**:

   You can also generate a detailed report of all login-related events:

   ```
   sudo aureport -l
   ```

   This command summarizes all login events, including successful and failed logins.

Step 3: Monitor real-time login activity with last and w

In addition to **`auditd`**, you can use the **`last`** and **`w`** commands to monitor user login activity.

1. **View recent logins with last**:

   The last command shows the history of user logins, reboots, and system run-level changes.

   ```
   $ last
   ```

   This command lists all successful logins, along with their start and end times.

2. **Monitor current user sessions with w**:

   The **`w`** command displays information about currently logged-in users and their processes.

   ```
   $ w
   ```

   This command shows who is logged on, what they are doing, and their current system load.

Step 4: Automate and centralize login monitoring

You can automate login monitoring and centralize the logs for more comprehensive monitoring.

1. **Set up a Cron job for regular reports**:

   You can schedule a **`cron`** job to run a report on login activity periodically.

   Example: Generate a daily login report:

   ```
   echo "0 0 * * * root /usr/sbin/aureport -l > /var/log/audit/login_
   report_$(date +\%Y\%m\%d).txt" | sudo tee -a /etc/crontab
   ```

   This job runs daily at midnight, saving the report in **`/var/log/audit/`**.

2. **Centralize logs with a log management system**:

   Consider forwarding audit logs to a centralized log server or SIEM system to correlate login activity across multiple systems.

   Example: Use `rsyslog` to forward logs:

   ```
   sudo vi /etc/rsyslog.d/audit.conf
   ```

   Add:

   ```
   $ModLoad imfile
   $InputFileName /var/log/audit/audit.log
   $InputFileTag audit:
   $InputFileStateFile audit-state
   $InputFileSeverity info
   $InputFileFacility local7
   $InputRunFileMonitor
   *.* @logserver.example.com:514
   ```

   Restart `rsyslog` to apply changes:

   ```
   sudo systemctl restart rsyslog
   ```

Step 5: Review and respond to alerts

Once your audit system is set up, regularly review the login reports and audit logs for any unusual activity, such as:

- Multiple failed login attempts which could indicate a brute-force attack.
- Logins at unusual times or from unexpected IP addresses.
- Unusual commands executed by users after login.

Respond to any suspicious activity immediately, following your organization's security protocols. Monitoring user login activity in RHEL is essential for maintaining the security and integrity of your systems. By configuring `auditd` with appropriate rules, using tools like `ausearch` and `aureport`, and supplementing with real-time commands like `last` and `w`, you can effectively track and audit user logins. Regularly reviewing these logs and setting up alerts for unusual activity will help you detect and respond to potential security threats promptly.

Recipe #100: Monitor software installation and updates

Monitoring software installations and updates on an RHEL system is crucial for maintaining security, compliance, and system stability. The `auditd` daemon, alongside other native tools, can be configured to monitor and log software installation and update activities

Step 1: Configure audit rules for monitoring package management

The **auditd** daemon can be configured to monitor the package management operations such as installations, updates, and removals. This is typically done by auditing the DNF package manager and related files.

1. **Monitor dnf command**:

   To monitor the use of DNF, the primary tool for package management in RHEL, you can set up audit rules that log when these commands are executed.

   Example: Monitor **dnf** command executions:

   ```
   auditctl -w /usr/bin/dnf -p x -k package_mgmt
   ```

   These rules monitor the execution (**x**) of the DNF command and tag the logs with the key **package_mgmt**. For YUM uses the same syntax.

2. **Monitor package database files**:

   You can also monitor changes to the package database files modified during installations, updates, and removals.

   Example: Monitor the RPM database:

   ```
   auditctl -w /var/lib/rpm/ -p wa -k rpm_db_changes
   ```

   This rule watches the **/var/lib/rpm/** directory for any write (**w**) and attribute change (**a**) operations, which indicates changes to the RPM database.

3. **Persist the Audit Rules**:

   To make these rules persistent across reboots, add them to **/etc/audit/rules.d/audit.rules**:

   ```
   echo "-w /usr/bin/dnf -p x -k package_mgmt" | sudo tee -a /etc/audit/rules.d/audit.rules
   echo "-w /usr/bin/yum -p x -k package_mgmt" | sudo tee -a /etc/audit/rules.d/audit.rules
   echo "-w /var/lib/rpm/ -p wa -k rpm_db_changes" | sudo tee -a /etc/audit/rules.d/audit.rules
   ```

   Then, reload the rules:

   ```
   sudo augenrules –load
   ```

Step 2: Monitor software installation and update logs

1. **Search for installation and update events using ausearch**:

   Use the **ausearch** command to search the audit logs for package management activities. Search for DNF executions:

   ```
   sudo ausearch -k package_mgmt
   ```

This command searches for all audit log entries tagged with the key **`package_mgmt`**, which corresponds to executions of DNF.

Search for changes to the RPM database:

```
sudo ausearch -k rpm_db_changes
```

This command searches for audit log entries related to changes in the RPM database.

2. **Generate reports with aureport**:

   The **`aureport`** tool can generate detailed reports from audit logs. You can use it to create reports specific to software management activities.

   Example: Generate a report on all software installation and update activities:

   ```
   sudo aureport -k --summary | grep -E "package_mgmt|rpm_db_changes"
   ```

   This command filters the summary report for entries related to package management and RPM database changes.

Step 3: Monitor logs with system tools

1. **Check logs with dnf.log**:

   In addition to **`auditd`**, you can directly check the logs generated by DNF:

   ```
   sudo cat /var/log/dnf.log
   ```

   The **`dnf.log`** file contains detailed information about all package installations, updates, and removals performed via DNF, for YUM use the **`/var/log/yum.log`** file.

2. **Monitor logs in real-time with tail**:

   To monitor package management activities in real time, you can use the tail command:

   Example: Monitor dnf.log in real-time:

   ```
   sudo tail -f /var/log/dnf.log
   ```

   This command will output new log entries to the terminal as they occur.

Step 4: Automate and centralize monitoring

1. **Set up Cron jobs for regular reports**:

   Automate the generation of reports on package management activities by setting up a **`cron`** job.

Example: Daily report of package management activities:

```
echo "0 0 * * * root /usr/sbin/aureport -k --summary > /var/log/audit/
package_mgmt_report_$(date +\%Y\%m\%d).txt" | sudo tee -a /etc/crontab
```

This `cron` job runs daily at midnight, generating a summary report and saving it in `/var/log/audit/`.

2. **Forward logs to a centralized log server**:

   Forwarding logs to a centralized log server or SIEM system allows you to monitor and analyze software installation and update activities across multiple systems.

   Example: Use `rsyslog` to forward logs:

   ```
   sudo vi /etc/rsyslog.d/audit.conf
   ```

   Add:

   ```
   $ModLoad imfile
   $InputFileName /var/log/audit/audit.log
   $InputFileTag audit:
   $InputFileStateFile audit-state
   $InputFileSeverity info
   $InputFileFacility local7
   $InputRunFileMonitor
   *.* @logserver.example.com:514
   ```

   Then restart `rsyslog`:

   ```
   sudo systemctl restart rsyslog
   ```

Step 5: Review and respond to software management activities

Regularly review the logs and reports for any suspicious activities, such as:

- Unexpected or unauthorized package installations or updates.
- Installation of packages from non-standard repositories.
- Modifications to critical system packages.

If any suspicious activity is detected, follow your organization's security incident response procedures to investigate and mitigate potential risks. By configuring `auditd` to monitor software installations and updates and by regularly reviewing system logs and reports, you can maintain tight control over package management activities on your RHEL systems. This helps to ensure system integrity, prevent unauthorized software changes, and comply with security and operational policies. Automating the monitoring process and centralizing logs further enhances your ability to manage and secure your environment effectively.

Conclusion

This chapter focused on critical components of system monitoring and auditing within RHEL. By establishing baseline capacity requirements, configuring performance monitoring, and implementing robust auditing mechanisms, you are now equipped to maintain a secure and efficient IT environment. Understanding these concepts is crucial for proactively managing system resources, detecting and responding to potential security incidents, and ensuring compliance with industry standards. As you continue to apply these practices, you enhance the reliability and security of your RHEL systems, thereby supporting your organization's broader IT goals.

In the next chapter, we will discuss artificial intelligence and machine learning, focusing on the tools and frameworks within RHEL for building, deploying, and managing AI solutions. Topics include setting up the RHEL AI environment, leveraging advanced features like GPU acceleration and cloud integration, and exploring use cases and best practices for effective AI implementation.

Points to remember

- **Baseline capacity**: Regularly assess system capacity using commands like **`sar`** and **`vmstat`** to ensure performance standards are met.
- **Audit rules**: Define and manage audit rules with **`auditctl`** to track critical system events and maintain detailed logs.
- **Performance monitoring**: Implement continuous monitoring with tools like **`prometheus`** and **`sysstat`** to capture real-time system metrics.
- **User login monitoring**: Use **`auditctl`** and **`ausearch`** to monitor and review user login activities, ensuring secure access control.
- **Software audit**: Monitor software installations and updates through **`auditctl`**, ensuring compliance and detecting unauthorized changes.
- **Pre-configured audit rules**: Apply standardized audit rules from **`scap-security-guide`** to align with security certification requirements.
- **Automated reporting**: Automate audit and performance reports using **`aureport`** and **`cron`** to maintain regular oversight of system activities.

Chapter 16
Artificial Intelligence and Machine Learning

Introduction

This chapter introduces the **artificial intelligence** (**AI**) capabilities within RHEL, focusing on how they empower enterprises to build, deploy, and manage AI solutions effectively. RHEL AI offers a comprehensive suite of tools and frameworks optimized for performance, security, and scalability, enabling organizations to harness the full potential of AI. As AI becomes increasingly vital in enterprise environments, RHEL AI provides the robust infrastructure to integrate AI into business operations, enhancing decision making, automating tasks, and driving innovation. This chapter will guide you through the foundational aspects of RHEL AI, preparing you to implement and utilize these capabilities to their fullest extent.

Structure

In this chapter, we will cover the following topics:

- Installing RHEL AI
- Setting up the environment
- Advanced RHEL AI features
- Developing custom AI applications
- Monitoring and maintenance

- Use cases and best practices
- Community and support

Objectives

By the end of this chapter, you will be able to actively contribute to the RHEL AI community through various means, including code contributions, testing, and documentation, while effectively accessing and utilizing official Red Hat support channels. You will enhance your knowledge by leveraging community forums, third-party resources, and engaging in discussions with other professionals. Additionally, you will understand the available training and certification options for RHEL AI, equipping you with the skills necessary for professional growth. Finally, you will learn how to stay informed about the latest trends and developments in AI by participating in webinars, workshops, and other educational opportunities.

Recipe #101: Installing RHEL AI

In this article, we will walk you through the process of installing RHEL AI, including all prerequisites, the installation steps, and how to configure AI components on your RHEL system.

Installation prerequisites

Before you begin the installation of RHEL AI, ensure that your system meets the necessary hardware and software requirements.

Hardware and software requirements

- **CPU**: A modern multi-core processor (64-bit)
- **Memory**: Minimum 8 GB of RAM; 16 GB or more is recommended for larger workloads
- **Disk Space**: At least 60 GB of free disk space
- **GPU (optional)**: NVIDIA, AMD, or Apple M-series GPU for accelerated AI workloads
- **Operating system**: RHEL 8.0 or later

Supported RHEL versions

RHEL AI is supported on the RHEL 8 version onward. Ensure your system is running one of these versions before installing.

Installing RHEL AI

Once you have confirmed that your system meets the prerequisites, you can proceed with installing RHEL AI.

Step-by-step installation guide

1. **Update your RHEL system**: First, make sure your RHEL system is up to date:
   ```
   sudo dnf update -y
   ```
2. **Install required dependencies**: Install the necessary development tools and libraries:
   ```
   sudo dnf groupinstall "Development Tools" -y
   sudo dnf install epel-release -y
   sudo dnf install python3 python3-devel gcc gcc-c++ make git -y
   ```
3. **Enable RHEL AI repositories**: Add the RHEL AI repositories to your system:
   ```
   sudo subscription-manager repos --enable=rhel-9-server-optional-rpms
   sudo subscription-manager repos --enable=rhel-9-server-extras-rpms
   ```
4. **Install RHEL AI components**: Install the RHEL AI components using the following command:
   ```
   sudo dnf install rhel-ai -y
   ```
5. **Verify installation**: After installation, verify that RHEL AI and its dependencies have been installed correctly:
   ```
   rhel-ai --version
   ```

Configuring AI components

Once RHEL AI is installed, you need to configure the AI components to match your system's requirements.

1. **Set up a virtual environment (optional)**: It is recommended to use a Python virtual environment to manage dependencies:
   ```
   python3 -m venv rhel-ai-env
   source rhel-ai-env/bin/activate
   ```
2. **Configure GPU acceleration (optional)**: If you have a supported GPU, configure RHEL AI to use it:
   a. For NVIDIA GPUs:
      ```
      sudo dnf install nvidia-driver nvidia-cuda-toolkit -y
      ```
      Configure the environment:
      ```
      export CUDA_HOME=/usr/local/cuda
      ```

```
export PATH=$CUDA_HOME/bin:$PATH
export LD_LIBRARY_PATH=$CUDA_HOME/lib64:$LD_LIBRARY_PATH
```

b. For AMD GPUs:

```
sudo dnf install rocm-dkms -y
```

Configure the environment:

```
export ROCM_HOME=/opt/rocm
export PATH=$ROCM_HOME/bin:$PATH
export LD_LIBRARY_PATH=$ROCM_HOME/lib64:$LD_LIBRARY_PATH
```

Initialize RHEL AI: Run the following command to initialize RHEL AI with the default configuration:

```
rhel-ai init
```

Download AI models: Download the pre-built AI models that you will use:

```
rhel-ai model download --model-name <model_name>
```

Replace **`<model_name>`** with the name of the AI model you want to download (e.g., **`default-ai-model`**).

3. **Test the installation**: Run a basic AI command to test the installation:

```
rhel-ai run –test
```

This command should execute a simple test and confirm that RHEL AI works correctly.

Following these steps, you have successfully installed and configured RHEL AI on your system. You are now ready to start leveraging the powerful AI tools and frameworks RHEL AI provides to build and deploy AI solutions. If you encounter any issues, consult the RHEL AI documentation or seek support from Red Hat.

Recipe #102: Setting up the environment

This article guides you through setting up your development environment and exploring the core components of RHEL AI. You will learn how to create and manage virtual environments, configure your setup, and understand the key AI modules, data processing tools, machine learning workflows, and model deployment strategies that RHEL AI offers.

Setting up the environment

Before diving into AI development, it is essential to set up a robust environment that can handle the complexities of AI workloads. Here's how you can do it effectively using RHEL AI.

Creating and managing virtual environments

Virtual environments are crucial for managing dependencies and ensuring that your AI projects remain isolated from systemwide configurations. Here is how to create and manage virtual environments in RHEL AI:

1. **Create a Python virtual environment**: It is recommended to use a Python virtual environment to manage dependencies for your AI projects:

   ```
   python3 -m venv rhel-ai-env
   source rhel-ai-env/bin/activate
   ```

   This creates and activates a virtual environment named `rhel-ai-env`.

2. **Install RHEL AI within the virtual environment**: Once the virtual environment is activated, install RHEL AI to ensure all necessary packages are contained within this isolated environment:

   ```
   pip install rhel-ai
   ```

3. **Managing dependencies**: With the environment activated, you can install additional packages required for your AI project using `pip`. For example, to install TensorFlow:

   ```
   pip install tensorflow
   ```

This ensures that your environment remains consistent and reproducible.

Initial configuration and setup

After setting up your virtual environment, it is important to configure RHEL AI for your specific use case:

1. **Initialize RHEL AI configuration**: Set up the default configuration for RHEL AI by running:

   ```
   rhel-ai init
   rhel-ai model download --model-name <model_name>
   ```

   Replace `<model_name>` with the desired AI model to be used in your project.

2. **Test the configuration**: Ensure everything is set up correctly by running a basic test command:

   ```
   rhel-ai run --test
   ```

With your environment set up, you are ready to explore the core components of RHEL AI.

Core components of RHEL AI

RHEL AI is packed with a variety of tools and modules designed to facilitate AI development. Here is an overview of these core components and how they can be utilized.

Overview of AI modules

RHEL AI includes several key modules that are essential for different stages of AI development:

- **Data processing**: Tools for ingesting, cleaning, and preparing data.
- **Machine learning**: Libraries and frameworks for training and validating AI models.
- **Model deployment**: Strategies and tools for deploying and serving AI models in production environments.

These modules are designed to work together seamlessly, providing a cohesive environment for AI development.

Data processing and management

Data is the backbone of any AI project. RHEL AI offers powerful tools for data processing and management:

1. **Tools for data ingestion, cleaning, and preparation**: RHEL AI supports a variety of tools to help you prepare your data:

    a. **Pandas**: For data manipulation and analysis.

    b. **NumPy**: For handling numerical computations.

    c. **Dask**: For processing large datasets efficiently.

    Example of data preparation using Pandas:

    ```
    import pandas as pd
    df = pd.read_csv('data.csv')
    df_clean = df.dropna()  # Remove missing values
    df_processed = df_clean.apply(lambda x: x*2)  # Example transformation
    df_processed.to_csv('processed_data.csv')
    ```

2. **Data processing workflow**: Use RHEL AI command line tools to streamline data processing tasks:

    ```
    rhel-ai data ingest --input data.csv --output clean_data.csv
    ```

Machine learning and model training

RHEL AI provides a robust environment for machine learning, supporting a range of libraries and frameworks.

1. **Available libraries and frameworks**: RHEL AI supports popular machine learning libraries such as:

    a. **TensorFlow**

b. **PyTorch**
c. **scikit-learn**

These libraries are optimized for performance on RHEL and can be easily integrated into your workflows.

2. **Workflow for model training and validation**: Here is an example of a typical workflow using scikit-learn:

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
# Load dataset
data = pd.read_csv('processed_data.csv')
X = data.drop('target', axis=1)
y = data['target']
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_
size=0.2)
# Train the model
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Validate the model
accuracy = model.score(X_test, y_test)
print(f'Model Accuracy: {accuracy}')
```

This script demonstrates how to load data, train a model, and validate its performance.

3. **Training models with RHEL AI**: You can also use RHEL AI command line interface to train models:

```
rhel-ai model train --data train_data.csv --model random_forest
```

Model deployment and serving

Deploying your AI models effectively is crucial for bringing your AI solutions into production:

1. **Strategies for deploying AI models**: RHEL AI provides various strategies for deploying AI models, whether on-premises or in the cloud. A typical deployment might involve:

```
rhel-ai model deploy --model random_forest --environment production
```

2. **Serving models with optimized performance**: RHEL AI allows you to serve your models efficiently, ensuring low latency and high throughput:

3. **Testing deployed models**: After deployment, it is important to test your models to ensure they are working as expected:

   ```
   rhel-ai model test --endpoint http://localhost:8000/predict
   ```

This command verifies that the model is correctly deployed and able to handle predictions.

Setting up a well-configured environment and understanding the core components of RHEL AI are crucial steps in developing effective AI solutions. By creating a virtual environment, configuring essential tools, and leveraging the powerful modules provided by RHEL AI, you can streamline your AI development process, from data preparation to model deployment. This setup will serve as a strong foundation as you continue to build, train, and deploy AI models efficiently within the RHEL ecosystem.

Recipe #103: Advanced RHEL AI Features

As organizations continue to adopt AI at scale, leveraging advanced features such as GPU acceleration, cloud integration, and stringent security measures becomes critical. This article delves into these advanced capabilities within RHEL AI, providing insights on how to enable, configure, and optimize them for enterprise environments.

GPU acceleration

GPU acceleration is a pivotal feature for AI workloads, significantly enhancing the performance of tasks such as training deep learning models. RHEL AI offers robust support for enabling and configuring GPU acceleration to meet the demands of modern AI workloads.

Enabling and configuring GPU support for AI workloads

To harness GPU acceleration in RHEL AI, follow these steps:

1. **Install the Required Drivers and Libraries** Ensure that your system has the necessary GPU drivers and libraries installed. For NVIDIA GPUs, install the CUDA toolkit:

   ```
   sudo dnf install cuda
   ```

   For AMD GPUs, ensure that ROCm is properly installed:

   ```
   sudo dnf install rocm
   ```

2. **Verify GPU availability**: After installation, verify that your system recognizes the GPU:

   ```
   nvidia-smi  # For NVIDIA
   rocminfo    # For AMD
   ```

3. **Configure RHEL AI for GPU acceleration**: Modify the configuration file to enable GPU support:

   ```
   rhel-ai config set gpu=true
   ```

   This command configures RHEL AI to leverage GPU resources during model training by specifying a batch size of 64:

   ```
   rhel-ai model train --batch-size 64
   ```

4. **Use mixed precision training**: Mixed precision training leverages the computational power of GPUs more efficiently by using half-precision (FP16) calculations:

   ```
   rhel-ai model train --precision half
   ```

5. **Monitor GPU performance**: Use tools like `nvidia-smi` or `rocminfo` to monitor GPU usage and temperature, ensuring that the GPUs are running within optimal parameters.

Integration with cloud services

Integrating RHEL AI with cloud services allows for scalable AI solutions that can seamlessly extend across on-premises and cloud environments. This hybrid approach offers flexibility and resilience in deploying AI models.

Utilizing cloud resources for Scalable AI

To leverage cloud resources for AI workloads, follow these steps:

1. **Configure cloud access**: Set up your cloud provider's CLI tool (e.g., AWS CLI, Azure CLI) and configure your credentials:

   ```
   aws configure  # For AWS
   az login       # For Azure
   ```

2. **Deploy models on cloud instances**: Use RHEL AI to deploy models on cloud instances:

   ```
   rhel-ai model deploy --cloud aws --instance-type p3.2xlarge
   ```

   This command deploys your model to an AWS EC2 instance optimized for GPU workloads.

3. **Scale automatically with cloud resources**: Configure auto-scaling policies to manage increasing workloads:

   ```
   aws autoscaling set-desired-capacity --auto-scaling-group-name
   myScalingGroup --desired-capacity 5
   ```

Hybrid AI models across on-premises and cloud

To build hybrid AI models that operate across on-premises and cloud environments:

1. **Data sync across environments**: Ensure that your data is synchronized between on-premises and cloud storage using tools like AWS DataSync or Azure Blob Storage:

   ```
   aws s3 sync /local/data s3://my-bucket/data
   ```

2. **Train models in a hybrid setup**: Split your training workload between on-premises GPUs and cloud GPUs:

   ```
   rhel-ai model train --data-source hybrid --on-prem-gpus 2 --cloud-gpus
   4
   ```

3. **Deploy models in a hybrid environment**: Deploy your models across both environments, ensuring high availability and fault tolerance:

   ```
   rhel-ai model deploy --hybrid --on-prem-server myServer --cloud-
   instance p3.8xlarge
   ```

Security and compliance

As AI models are increasingly used in sensitive and critical applications, ensuring security and compliance with industry standards is crucial. RHEL AI offers several features and best practices to help secure your AI applications.

Best practices for securing AI applications

To secure AI applications built with RHEL AI:

1. **Use encrypted data and models**: Ensure that all data and models are encrypted both at rest and in transit:

   ```
   rhel-ai encrypt --data /path/to/data --key myEncryptionKey
   ```

2. **Implement role-based access control (RBAC)**: Restrict access to AI models and data by implementing RBAC:

   ```
   rhel-ai access control --role AIEngineer --allow model.train,model.
   deploy
   ```

3. **Audit logs and monitoring**: Enable auditing and monitoring to track access and changes to your AI models:

   ```
   rhel-ai audit enable --log /var/log/rhel-ai/audit.log
   ```

Ensuring compliance with industry standards

To ensure compliance with industry standards such as GDPR or HIPAA:

1. **Data anonymization**: Anonymize personal data before using it in AI models:
   ```
   rhel-ai data anonymize --input raw_data.csv --output anon_data.csv
   ```
2. **Regular compliance audits**: Schedule regular audits of your AI applications to ensure ongoing compliance:
   ```
   rhel-ai compliance audit --standard GDPR --output audit_report.txt
   ```
3. **Documentation and reporting**: Maintain detailed documentation of all AI processes and generate compliance reports as required by industry standards:
   ```
   rhel-ai report generate --compliance --output compliance_report.pdf
   ```

By effectively utilizing GPU acceleration, integrating cloud services, and adhering to security and compliance best practices, you can maximize the performance, scalability, and safety of your AI applications in RHEL AI. These advanced features empower organizations to deploy AI at scale, ensuring robust and secure operations across diverse environments.

Recipe #104: Developing custom AI applications

As enterprises increasingly adopt AI, there is a growing need to create custom AI models and applications tailored to specific business needs. RHEL AI provides a robust platform for developing, extending, and automating AI workflows. This article explores how to leverage RHEL AI to create custom models, integrate third-party tools, and orchestrate complex AI workflows.

Creating custom models

Developing custom AI models is a critical step in tailoring AI to meet unique organizational needs. RHEL AI offers comprehensive tools and resources to guide you through the process of developing and training custom AI models.

Guide to developing and training custom AI models

To develop custom AI models using RHEL AI, follow these steps:

1. **Define the problem and dataset**: Start by clearly defining the problem your AI model will address and gathering the necessary data. Ensure your dataset is well labelled and representative of the problem space.
2. **Set up the development environment**: Set up a development environment on RHEL that includes all necessary libraries and tools. Use Python virtual environments to manage dependencies:
   ```
   python3 -m venv ai-env
   ```

```
source ai-env/bin/activate
pip install tensorflow keras scikit-learn
```

3. **Design the model architecture**: Depending on the complexity of the task, design a neural network architecture using frameworks like TensorFlow, PyTorch, or Keras:

```
from tensorflow.keras import layers, models
model = models.Sequential([
    layers.Dense(128, activation='relu', input_shape=(input_shape,)),
    layers.Dropout(0.5),
    layers.Dense(64, activation='relu'),
    layers.Dense(output_shape, activation='softmax')
])
```

4. **Train the model**: Train the model using your dataset. Leverage RHEL AI GPU acceleration capabilities to speed up training:

```
rhel-ai model train --dataset path/to/dataset --epochs 50 --gpus 2
```

5. **Evaluate and fine-tune**: After training, evaluate the model's performance using metrics like accuracy, precision, and recall. Fine-tune the model based on the evaluation results:

```
rhel-ai model evaluate --model path/to/model --dataset path/to/
validation_dataset
```

6. **Deploy the custom model**: Once the model is trained and fine-tuned, deploy it using RHEL AI's deployment tools:

```
rhel-ai model deploy --model path/to/custom_model --endpoint /ai/
model/predict
```

Extending RHEL AI capabilities

RHEL AI's flexibility allows you to integrate third-party libraries and tools, enhancing the capabilities of your AI applications.

Integrating third-party libraries and tools

To extend the capabilities of RHEL AI with third-party libraries and tools:

1. **Identify necessary extensions**: Determine which third-party libraries or tools will best complement your AI application, such as **natural language processing** (**NLP**) libraries or data visualization tools.
2. **Install the libraries**: Install the necessary third-party libraries using the package manager:

```
pip install spacy matplotlib seaborn
```

3. **Integrate with RHEL AI**: Integrate these libraries into your AI workflows. For example, use spaCy for advanced NLP tasks or Matplotlib for visualizing data:

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("RHEL AI makes AI development easy.")
print([(token.text, token.pos_) for token in doc])
```

4. **Extend model functionality**: Use the integrated tools to extend the functionality of your AI models, such as adding advanced text processing or creating detailed visualizations of model outputs:

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.show()
```

5. **Maintain compatibility**: Ensure that the integrated tools are compatible with your existing RHEL AI environment and workflows, testing each integration thoroughly.

Automation and orchestration

Automating and orchestrating AI workflows is essential for scaling AI applications and ensuring consistency in deployment and management.

Using RHEL AI for automated AI pipelines

Automation streamlines the development, training, and deployment of AI models. Here is how to set up automated AI pipelines with RHEL AI:

1. **Define the pipeline stages**: Break down your AI workflow into stages such as data preprocessing, model training, evaluation, and deployment.

2. **Automate with RHEL AI CLI**: Use RHEL AI command line interface to automate each stage of the pipeline:

```
rhel-ai data preprocess --input raw_data.csv --output preprocessed_
data.csv
rhel-ai model train --dataset preprocessed_data.csv --epochs 50
rhel-ai model deploy --model trained_model
```

3. **Schedule pipeline execution**: Schedule the pipeline to run at regular intervals or trigger it based on specific events using cron jobs or a CI/CD tool like Jenkins:

```
0 2 * * * /usr/local/bin/rhel-ai run_pipeline.sh
```

4. **Monitor and log pipelines**: Ensure that all stages of the pipeline are logged and monitored for performance issues or errors:

   ```
   rhel-ai monitor --pipeline my_pipeline --log /var/log/ai_pipeline.log
   ```

Orchestrating complex AI workflows with Ansible

For complex AI workflows, orchestrate tasks using Ansible, which can manage and automate multiple environments and processes.

1. **Set up Ansible playbooks**: Create playbooks to define the tasks involved in your AI workflows, such as setting up environments, deploying models, and executing pipelines:

   ```
   - name: Deplou AI Model
     hosts: ai-servers
     tasks:
       - name: Set up environment
          ansible.builtin.shell: rhel-ai model deploy --model /models/
   custom_model
   ```

2. **Automate workflow execution**: Use Ansible to execute the entire workflow across multiple servers or environments:

   ```
   ansible-playbook ai_workflow.yml
   ```

3. **Ensure consistency and compliance**: Ansible ensures that your AI environments and workflows are consistent and compliant across all nodes, reducing the risk of errors and simplifying maintenance.

By following these practices, organizations can develop, extend, and automate custom AI applications using RHEL AI, ensuring they are robust, scalable, and tailored to specific business needs. These capabilities empower enterprises to leverage the power of AI in their operations fully.

Recipe #105: monitoring and maintenance

Managing AI workloads in production requires a solid understanding of monitoring, troubleshooting, and maintaining the software stack. This article will guide you through the essential aspects of monitoring AI workloads, troubleshooting common issues, and managing updates for AI components using InstructLab.

Monitoring AI workloads

Tools for real-time monitoring of AI applications

Monitoring your AI workloads in real-time is crucial to ensure they are performing optimally and to detect issues early. InstructLab provides several tools and techniques to monitor AI applications effectively:

1. **Prometheus and Grafana integration**:

   a. **Prometheus** is used for collecting and storing metrics.

   b. **Grafana** can visualize these metrics, providing dashboards that show real-time performance data.

   Example setup:

   Install Prometheus

   ```
   sudo apt-get install Prometheus
   ```

   Install Grafana

   ```
   sudo apt-get install Grafana
   ```

   Start Prometheus and Grafana services

   ```
   sudo systemctl start Prometheus
   sudo systemctl start grafana-server
   ```

   Configure Prometheus to scrape metrics from your AI application. This involves adding a job in the `prometheus.yml` configuration file:

   ```
   scrape_configs:
     - job_name: 'instructlab'
       static_configs:
         - targets: ['localhost:9090']
   ```

   You can then visualize these metrics in Grafana by adding Prometheus as a data source.

2. **InstructLab's built-in monitoring commands**: InstructLab CLI provides built-in commands to monitor system resources while running AI workloads:

   View system information

   ```
   ilab system info
   # Monitor the usage of AI models
   ilab model serve –monitor
   ```

   These commands help you monitor CPU, memory, and GPU utilization in real time.

Analyzing performance metrics and logs

Analyzing performance metrics and logs is essential to understand how your AI workloads are functioning and where bottlenecks might occur.

1. **Accessing logs**: InstructLab logs all significant events, errors, and warnings to a log file. You can view these logs using standard tools like tail or grep:

   View real-time logs:

   ```
   tail -f /var/log/instructlab/instructlab.log
   ```

   Search for specific errors:

   ```
   grep "ERROR" /var/log/instructlab/instructlab.log
   ```

2. **Performance metrics**: Use the data collected by Prometheus to analyze performance over time. You can create Grafana dashboards to visualize CPU, memory, and GPU usage trends and set up alerts for anomalies:

   Example: Prometheus query for CPU usage

   ```
   rate(node_cpu_seconds_total{job="instructlab", mode="idle"}[5m])
   ```

   Set up alerts in Grafana for when resource usage exceeds a predefined threshold, ensuring timely intervention.

Troubleshooting common issues

Diagnosing and resolving AI-related problems: Despite the best monitoring tools, issues can still arise. Here is how to diagnose and troubleshoot common AI-related problems with InstructLab:

1. **Model loading issues**:

   a. **Symptom**: The model fails to load or crashes during serving phase.

      ```
      ilab model serve --verbose
      ```

      Use the `--verbose` flag to get detailed error messages and stack traces.

   b. **Solution**:

      i. Check for compatibility issues between the model format and your hardware.

      ii. Verify that all necessary dependencies are installed.

2. **Performance degradation**:

   a. **Symptom**: The AI model performs significantly slower over time.

   b. **Diagnosis**:

      ```
      ilab system info
      ```

      Check for memory leaks or excessive CPU/GPU usage.

   c. **Solution**:

      i. Restart the AI model server to clear any memory leaks.

      ii. Optimize the model by reducing the batch size or using a more efficient model format (e.g., quantized models).

3. **Unexpected output**:

   a. **Symptom**: The AI model produces incorrect or unexpected results.

   b. **Diagnosis**:

      ```
      ilab model chat --debug
      ```

      Use the `--debug` flag to trace input and output at each processing stage.

   c. **Solution**:

      i. Validate the input data format.

      ii. Retrain the model with additional or corrected training data.

Updating and upgrading AI components

Managing updates for AI tools and libraries: Keeping your AI tools and libraries up-to-date is essential for performance improvements, security patches, and new features.

1. **Updating InstructLab**:

   a. Use `pip` to update InstructLab to the latest version:

      ```
      pip install instructlab –upgrade
      ```

   b. Check the release notes for any breaking changes or new features:

      ```
      ilab –version
      ```

2. **Updating dependencies**:

   a. Regularly update Python packages using `pip`:

      ```
      pip list --outdated
      pip install --upgrade <package_name>
      ```

   b. Ensure you are using compatible versions of dependencies, especially for critical libraries like PyTorch or TensorFlow.

Ensuring backward compatibility and stability

When updating AI components, it is important to ensure that the updates do not break existing workflows.

1. **Version pinning**:
   a. Pin versions of critical dependencies in your **`requirements.txt`** to ensure stability:
      ```
      instructlab==1.0.0
      torch==2.0.1
      ```
   b. Use virtual environments to isolate dependencies for different projects:
      ```
      python3 -m venv instructlab_env
      source instructlab_env/bin/activate
      ```
2. **Testing after updates**:
   a. After updating, run a full suite of tests on your AI models to ensure they function as expected:
      ```
      ilab model test
      ```
   b. Test both the functionality and performance to ensure no regressions have been introduced.

By following these practices, you can maintain a robust AI infrastructure that is both up-to-date and reliable, minimizing downtime and maximizing the performance of your AI workloads.

Recipe #106: Use cases and best practices

Case studies: Real-world applications of RHEL AI: RHEL has become a robust platform for deploying AI workloads across various industries. Here are some case studies that highlight the power and flexibility of RHEL AI in real-world applications:

Case study 1: Healthcare diagnostics

Problem: A large healthcare provider needed to enhance diagnostic accuracy and speed in radiology, specifically in detecting anomalies in X-rays and MRIs.

Solution: The provider deployed AI models trained on RHEL to process medical images in real-time. Using containerized AI workloads on RHEL, the healthcare provider could scale their operations efficiently across multiple data centers.

Outcome: The AI solution reduced the time required to analyze images by 70%, significantly improving patient care and operational efficiency. The flexibility of RHEL allowed seamless integration with existing healthcare software, ensuring compliance with industry regulations.

Case study 2: Financial fraud detection

Problem: A global financial institution was facing increasing challenges in detecting fraudulent transactions in real-time due to the growing volume of transactions and sophisticated fraud tactics.

Solution: By leveraging RHEL's stability and security features, the institution deployed machine learning models to analyze transactional data streams. These models were trained on historical data and continuously updated to identify new fraud patterns.

Outcome: The implementation of AI on RHEL reduced false positives by 50% and significantly increased the accuracy of fraud detection, saving the institution millions of dollars annually. The scalability of RHEL allowed the AI system to handle peak transaction loads without performance degradation.

Case study 3: Manufacturing process optimization

Problem: A leading automotive manufacturer needed to optimize its production line to reduce waste and increase efficiency.

Solution: The manufacturer used AI models running on RHEL to analyze data from IoT devices on the production floor. These models predicted equipment failures and suggested optimal maintenance schedules, reducing downtime.

Outcome: The AI-driven approach on RHEL improved production efficiency by 20%, reduced unplanned downtime by 30%, and contributed to significant cost savings. The robustness of RHEL ensured high availability and reliability of the AI applications, critical in a production environment.

Best practices for AI deployment

Successfully deploying AI in an enterprise environment requires more than just choosing the right tools—it involves strategic planning and best practices to ensure scalability, security, and performance. Here are some strategies for successful AI adoption in enterprises:

Infrastructure scalability

AI workloads often require significant computational resources, which can vary depending on the application. Deploying AI on RHEL provides the flexibility to scale your infrastructure as needed:

- **Containerization**: Use containers to manage AI workloads. Tools like Podman, available on RHEL, allow you to deploy and scale AI models efficiently across different environments.
- **Hybrid cloud**: Consider a hybrid cloud approach, combining on-premises and cloud resources. RHEL supports seamless integration with cloud platforms, allowing for flexible scaling as your AI workloads grow.

Security and compliance

AI systems often handle sensitive data, making security a top priority:

- **Data encryption**: Ensure all data, both at rest and in transit, is encrypted. RHEL provides built-in security features like SELinux that can be configured to enforce strict access controls.
- **Regular audits**: Conduct regular security audits to identify and address vulnerabilities. Use RHEL’s auditing tools to monitor and log all critical activities in your AI systems.

Performance optimization

Optimizing the performance of AI models on RHEL involves tuning both hardware and software configurations:

- **Resource allocation**: Use RHEL's resource management tools to allocate CPU, memory, and GPU resources efficiently. Tools like cgroups and tuned profiles can help in optimizing resource usage based on workload requirements.
- **Model optimization**: Consider model compression techniques like quantization to reduce the computational load without sacrificing accuracy. This can be especially useful in deploying AI models on edge devices.

Continuous integration and deployment (CI/CD)

Adopt CI/CD practices to ensure that AI models are continuously updated and deployed without disrupting production systems:

- **Automation**: Automate the testing and deployment of AI models using tools like Jenkins or GitLab CI/CD, integrated with RHEL.
- **Monitoring and feedback loops**: Implement monitoring systems to gather feedback on AI model performance post-deployment. Use this feedback to continuously improve and retrain models.

Future trends in AI on RHEL

As AI technology evolves, so does the landscape in which it operates. Here are some emerging trends and how RHEL AI is expected to evolve:

Edge AI

With the proliferation of IoT devices, there is a growing trend towards deploying AI at the edge, where data is generated. RHEL is well-positioned to support edge AI due to its lightweight and scalable architecture:

- **Real-time processing**: RHEL's support for real-time processing makes it ideal for deploying AI models on edge devices that require immediate decision-making capabilities.
- **Edge-to-cloud integration**: RHEL's capabilities in hybrid cloud environments will enable seamless integration between edge devices and centralized cloud systems, ensuring that AI models are consistently updated and managed.

Explainable AI (XAI)

As AI becomes more integral to decision-making processes, there is a growing demand for explainable AI, where the decisions made by AI models can be understood and interpreted by humans:

- **Compliance and trust**: RHEL AI is expected to integrate more tools that provide transparency and accountability in AI models, helping enterprises comply with regulations and build trust with users.
- **Model interpretability**: Future updates to RHEL AI could include libraries and frameworks that make it easier to implement and deploy XAI techniques, ensuring that AI decisions are not just accurate but also explainable.

AI governance and ethics

With AI's increasing role in critical applications, governance and ethical considerations will become more prominent:

- **AI governance tools**: RHEL is likely to introduce more robust governance tools to manage AI models, ensuring they operate within defined ethical boundaries and comply with regulatory standards.
- **Bias mitigation**: Future AI solutions on RHEL will likely focus on identifying and mitigating biases in AI models, promoting fairness and inclusivity in AI-driven decisions.

Integration of quantum computing

Quantum computing is an emerging field with the potential to revolutionize AI:

- **Quantum AI**: RHEL is exploring ways to integrate quantum computing capabilities, allowing enterprises to leverage quantum algorithms for AI tasks that require immense computational power.
- **Hybrid quantum-classical workflows**: RHEL will likely support hybrid workflows, where quantum and classical computing resources are used together to solve complex AI problems more efficiently.

In conclusion, RHEL is not just a platform for deploying AI—it is evolving into a comprehensive ecosystem that supports the entire AI lifecycle, from development and deployment to monitoring and governance. By following best practices and staying ahead of emerging trends, enterprises can leverage RHEL to drive successful AI initiatives.

Recipe #107: Community and support

The RHEL AI community is a vibrant ecosystem of developers, data scientists, and IT professionals who collaborate to enhance the AI capabilities of RHEL. Whether you're a seasoned professional or just starting in the field of AI, there are numerous ways to contribute to and benefit from this community.

Contribute to open-source projects

RHEL AI is built on open-source technologies, and contributing to these projects is a great way to get involved:

- **Code contributions**: If you have programming expertise, you can contribute code to RHEL AI-related projects. This could involve adding new features, fixing bugs, or improving documentation. GitHub repositories are the primary platform for contributing to RHEL AI projects, where you can submit pull requests and engage in code reviews.
- **Documentation**: High-quality documentation is crucial for the success of any software project. You can contribute by writing or improving documentation, making it easier for others to understand and use RHEL AI tools.
- **Testing and reporting bugs**: Even if you're not a developer, you can help by testing RHEL AI tools and reporting bugs. This feedback is invaluable for maintaining the stability and performance of RHEL AI.

Join community discussions

Participating in community discussions is an excellent way to share knowledge, ask questions, and collaborate with others:

- **Forums and mailing lists**: RHEL AI has dedicated forums and mailing lists where community members discuss various topics related to AI deployment, best practices, and troubleshooting. Joining these platforms allows you to stay updated on the latest developments and connect with like-minded professionals.
- **Online communities**: Platforms like *Reddit*, *Stack Overflow*, and specialized AI communities on platforms like *Discord* and *Slack* provide additional venues for discussing RHEL AI-related topics. These communities are often active and provide real-time support and collaboration opportunities.

Attend and present at conferences and meetups

Engage with the broader AI community by attending or presenting at conferences and meetups:

- **Red Hat Summit**: The annual Red Hat Summit is a premier event where you can learn about the latest developments in RHEL AI, attend workshops, and network with industry professionals. Presenting at the summit is a great way to showcase your work and contribute to the community.
- **Local meetups and workshops**: Many cities host local RHEL or AI meetups where you can connect with other professionals, share your experiences, and learn from others. Participating in or organizing these events is an excellent way to contribute to the community.

Mentorship and collaboration

If you have experience in AI and RHEL, consider mentoring others:

- **Mentorship programs**: Engage in mentorship programs where you can guide newcomers through their journey in AI and RHEL. This could involve one-on-one mentoring or hosting workshops and webinars.
- **Collaborative projects**: Work on collaborative projects within the community. These projects often address real-world problems and provide a great way to apply your skills while contributing to the community.

Accessing support and resources: Official channels and community help

Navigating the resources and support available for RHEL AI is crucial for effective deployment and troubleshooting. Whether you are looking for official support or community-driven help, here is how to access the resources you need.

Official support channels

Red Hat provides several official support channels for RHEL AI users:

- **Red Hat Customer Portal**: The Red Hat Customer Portal is the primary platform for accessing official support. Here, you can submit support tickets, track the progress of your queries, and access a wealth of documentation and knowledge base articles tailored to RHEL AI.
- **Service level agreements (SLAs)**: Depending on your subscription, you may have access to different levels of support, ranging from standard to premium. These SLAs define the response times and types of support you can expect, ensuring that your AI workloads are supported according to your business needs.
- **Support contracts**: For enterprises with critical AI deployments, Red Hat offers support contracts that provide 24/7 access to expert assistance. This is particularly valuable for organizations that require high availability and quick resolution of issues.

Documentation and knowledge base

Red Hat offers extensive documentation and a knowledge base to help you get the most out of RHEL AI:

- **Official documentation**: The RHEL AI documentation is comprehensive, covering everything from installation and configuration to advanced usage and troubleshooting. This documentation is regularly updated to reflect the latest features and best practices.
- **Knowledge base**: The Red Hat knowledge base is a repository of articles that address common issues, configuration tips, and advanced use cases. It is an essential resource for troubleshooting and learning more about the intricacies of RHEL AI.

Community forums and discussion boards

The RHEL AI community is highly active, with many forums and discussion boards where you can seek help and share knowledge:

- **Red Hat Community Portal**: The Red Hat Community Portal hosts discussion boards where you can ask questions and participate in discussions about RHEL AI. It is a great place to get help from both Red Hat experts and community members.
- **Stack Overflow and Reddit**: For more general AI-related queries, platforms like Stack Overflow and Reddit have active communities where you can ask questions and find solutions to common problems. Tagging your questions with relevant keywords like *RHEL* and *AI* can help you get more targeted responses.

Training and certification

Red Hat offers training programs and certifications that can help you deepen your expertise in RHEL AI:

- **Red Hat Training**: Red Hat provides a range of training courses that cover various aspects of RHEL AI, from the basics to advanced deployment strategies. These courses are available in different formats, including online, classroom, and on-site training.
- **Certifications**: Earning a Red Hat certification in AI or related technologies can validate your skills and enhance your career prospects. Certifications like **Red Hat Certified Engineer** (**RHCE**) and Red Hat Certified Specialist in AI are recognized globally and can set you apart in the industry.

Webinars and workshops

Stay updated and continue learning through webinars and workshops:

- **Red Hat Webinars**: Red Hat regularly hosts webinars that cover the latest updates in RHEL AI, best practices, and case studies. These webinars are a convenient way to stay informed and learn directly from experts.
- **Workshops and hands-on labs**: Participating in workshops and hands-on labs offered by Red Hat and its partners allows you to gain practical experience with RHEL AI. These sessions are often designed to help you solve real-world problems using RHEL AI tools and techniques.

Third-party resources

In addition to official Red Hat resources, there are numerous third-party resources available:

- **Books and online courses**: Several books and online courses focus on AI deployment on RHEL. Platforms like *Coursera*, *Udemy*, and *edX* offer courses that can supplement your learning and provide practical insights into AI on RHEL.
- **Blogs and technical articles**: Many experts in the community share their knowledge through blogs and technical articles. These resources often provide unique insights, tutorials, and best practices that can be valuable for advanced users.

In conclusion, the RHEL AI community and support ecosystem provide a wealth of resources for users at all levels. Whether you are looking to contribute to the community, seek help, or advance your skills, there are numerous ways to get involved and access the support you need to succeed with AI on RHEL.

Conclusion

This chapter explores the vibrant community and support ecosystem surrounding RHEL AI, highlighting opportunities for contribution and collaboration. Readers are introduced to official support channels provided by Red Hat, including the Customer Portal and SLAs. The chapter also details the extensive documentation and knowledge base available to users, along with community forums and discussion boards that offer peer support. Additionally, the chapter discusses training and certification opportunities, webinars, and workshops that can help deepen expertise in RHEL AI. The final section touches on third-party resources like books, online courses, and technical articles that complement the official materials.

Points to remember

- RHEL AI integrates with RHEL to optimize AI workloads on enterprise systems.
- `sudo dnf install rhel-ai`: Install RHEL AI components.
- `rhel-ai –version`: Verify RHEL AI installation.

- `pip install rhel-ai`: Install RHEL AI in the virtual environment.
- `rhel-ai init`: Initialize RHEL AI.
- `rhel-ai model download --model-name <model_name>`: Download AI models.
- `rhel-ai run --test`: Test the RHEL AI installation.
- `sudo dnf install nvidia-driver nvidia-cuda-toolkit`: Install NVIDIA GPU drivers and CUDA toolkit.
- `rhel-ai config set gpu=true`: Enable GPU support in RHEL AI.
- `rhel-ai model train`: Train AI models.
- `rhel-ai model deploy`: Deploy AI models.
- `rhel-ai encrypt`: Encrypt data and models.
- `rhel-ai access control`: Implement role-based access control.
- `rhel-ai audit enable`: Enable audit logs and monitoring.
- `rhel-ai compliance audit`: Conduct compliance audits.
- `ilab system info`: View system information.
- `ilab model serve --monitor`: Monitor AI models.
- `rhel-ai-cicd`: Continuous integration and deployment of AI models.

Index