Operating System and System Programming

Project Work

Department of Software Engineering, Bahir Dar University Institution of Technology

SENG2032: OSSP

Instructor: Mr. Wendimu B.

Sep 28, 2024

Table of Contents

Introduction

1. Background:

The Red Hat Enterprise Linux (RHEL) operating system is one of the most reliable and secure distributions in the Linux family, designed primarily for enterprise environments. RHEL is favored by many organizations due to its robust security features, extensive hardware support, and long term stability, making it ideal for both small scale and large scale applications.

What is RHEL?

RHEL is a commercially supported Linux distribution developed by **Red Hat** for the business sector. It is built from open source technologies and focuses on security, stability, and performance. Red Hat offers subscription based support, with regular updates and patches that ensure security and the ability to meet compliance needs. It is an excellent choice for businesses running critical applications like databases, web services, and network infrastructure.

The origins of RHEL trace back to the mid 1990s when Linux based distributions started gaining traction in both academic and corporate settings. By the early 2000s, Red Hat had established itself as a leader in open source software, culminating in the release of **Red Hat Linux**, which eventually transitioned into the enterprise focused **RHEL** distribution. The advent of **RHEL** signaled a shift toward the development of Linux systems that could serve as dependable backbones for corporations and government institutions alike.

Key Features of RHEL:

- **Security Features**: Enhanced security policies, SELinux (Security Enhanced Linux), and support for cryptographic frameworks.
- **Filesystem Support**: Wide support for multiple filesystems like ext4, XFS, and Btrfs, allowing flexibility in how storage is handled.
- **Long Term Support**: Red Hat offers 10 years of support for RHEL versions, ensuring that businesses can maintain a stable production environment.
- **Package Management**: RHEL uses the **YUM (Yellowdog Updater Modified)** package manager, which simplifies the installation and management of software packages.
- **Virtualization Support**: RHEL integrates with tools like **KVM (Kernel based Virtual Machine)** for virtualization, enabling enterprises to efficiently manage virtualized environments.

2. Motivation:

The motivation behind learning and using RHEL is driven by its relevance in the enterprise IT landscape. System administrators, developers, and engineers are increasingly required to work with Linux based environments, with RHEL being a common choice. By installing and working with RHEL, students and professionals can:

- Gain hands on experience with one of the most widely used enterprise Linux distributions.
- Develop skills in system administration, including user management, networking, and security.
- Understand the unique features of RHEL and how it supports enterprise level workloads.

The experience of setting up a virtual environment to install RHEL also provides familiarity with virtualization, which is a key skill in cloud computing, data center management, and DevOps.

3. Objectives

The objectives of this project include:

1. **Understanding the RHEL Installation Process**: Gain practical knowledge of installing RHEL, from setting up the virtual environment to completing the post installation configuration.
2. **Virtualization Setup**: Learn to use virtualization software (VMware or VirtualBox) to manage virtual machines and understand how virtualization allows for safe and isolated installations.
3. **Filesystem Choices in RHEL**: Investigate the filesystem options available during installation (ext4, XFS, Btrfs) and make an informed decision on which to use based on performance, stability, and scalability.
4. **Problem Solving During Installation**: Encounter potential installation issues and document troubleshooting steps.
5. **Gain Practical Experience in System Administration**: Learn basic Linux commands, user management, network configuration, and system updates in a RHEL environment.
6. **Documentation of the Process**: Create a detailed report of the installation process, covering all key aspects such as prerequisites, setup, and post installation configuration.

Requirements

I. Hardware Requirements

To install RHEL in a virtual environment, specific hardware specifications are needed to ensure smooth operation. Below are the recommended minimum hardware requirements:

1. **Processor (CPU)**:

A multi core processor is essential to allocate sufficient resources to both the host machine and the virtual machine. RHEL requires at least **1 GHz CPU** for basic operations, but **Intel Core i5/i7** or **AMD Ryzen** processors are recommended for a better experience.

2. **Memory (RAM)**:

The RHEL installation process requires a minimum of **2 GB RAM**; however, allocating **4 GB or more** ensures smoother performance, especially when running other applications on the host machine. More memory may be required if testing multiple virtual machines simultaneously or running resource intensive applications.

3. **Storage (HDD/SSD)**:

A minimum of **20 GB of free disk space** is recommended for the installation of RHEL, but larger storage sizes (e.g., **50 GB**) are ideal for storing additional packages, updates, and files.

SSD storage is preferred for faster read/write speeds, particularly when managing multiple VMs or running I/O intensive applications.

4. **Virtualization Support**:

Ensure the host system's **BIOS** supports **hardware virtualization** (Intel VT x or AMD V) and that it is enabled. This allows for better CPU performance and resource allocation in virtual environments.

5. **Networking**:

An internet connection is optional but recommended to download updates, packages, and additional software during or after the installation.

II. Software Requirements

1. **Virtualization Software**:
   - **VMware Workstation Player** or **Oracle VM VirtualBox** are the two primary tools used to create and manage virtual machines.

- ✓ **VMware Workstation Player** is known for its ease of use and high performance. It offers features such as drag and drop file transfer and multi monitor support.
- ✓ **Oracle VM VirtualBox** is an open source alternative with excellent support for a variety of guest operating systems.

2. **RHEL ISO Image**:
   - Download the latest version of **Red Hat Enterprise Linux** from the official **Red Hat Customer Portal**. A Red Hat Developer Subscription provides free access to RHEL for individual development purposes. The ISO image file is typically around **6 10 GB** depending on the version and desktop environment selected (GNOME or minimal).
3. **Host Operating System**:
   - The host machine should have a stable OS (e.g., **Windows 10/11**, **macOS**, or **Linux**) with the necessary virtualization software installed. Ensure that the host OS has sufficient resources to run the virtual machine alongside other applications.

4. Installation Steps (RHEL)

Step 1: Downloading the RHEL ISO Image

- Visit the **Red Hat Developer Portal** and create an account (if not already created).
- Navigate to the **Downloads** section, select **Red Hat Enterprise Linux**, and download the **ISO file**. This file will be used to install the OS in the virtual environment.

Step 2: Installing VirtualBox

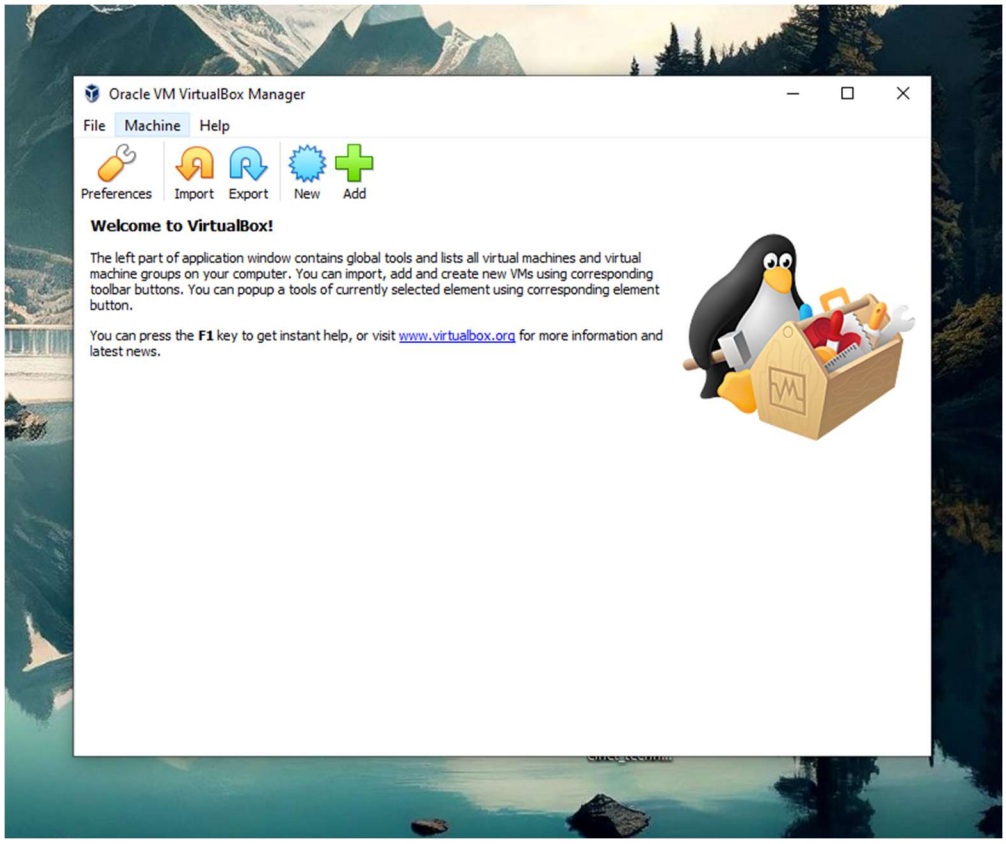
- Download and install **VirtualBox**. I have chosen **VirtualBox** for this topic and the setup can be obtained from their official websites I tried to add it on GitHub for easy access but I couldn't do it because of the size quota, but you could find using this link https://www.virtualbox.org/wiki/Downloads

Installation of VirtualBox:

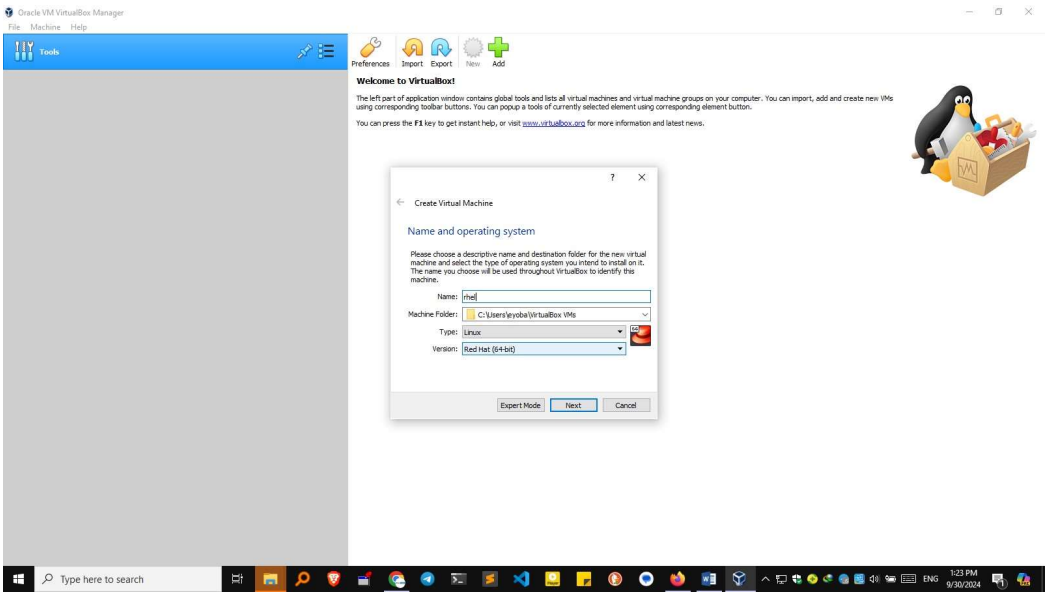
- Download the **VirtualBox** installer.
- Run the installer and follow the prompts, accepting the default installation settings.
- After installation, launch **VirtualBox** and create a new virtual machine.

Step 3: Creating a New Virtual Machine in VirtualBox

- Open **VirtualBox**.
- Select **Create New Virtual Machine by clicking the star icon**.



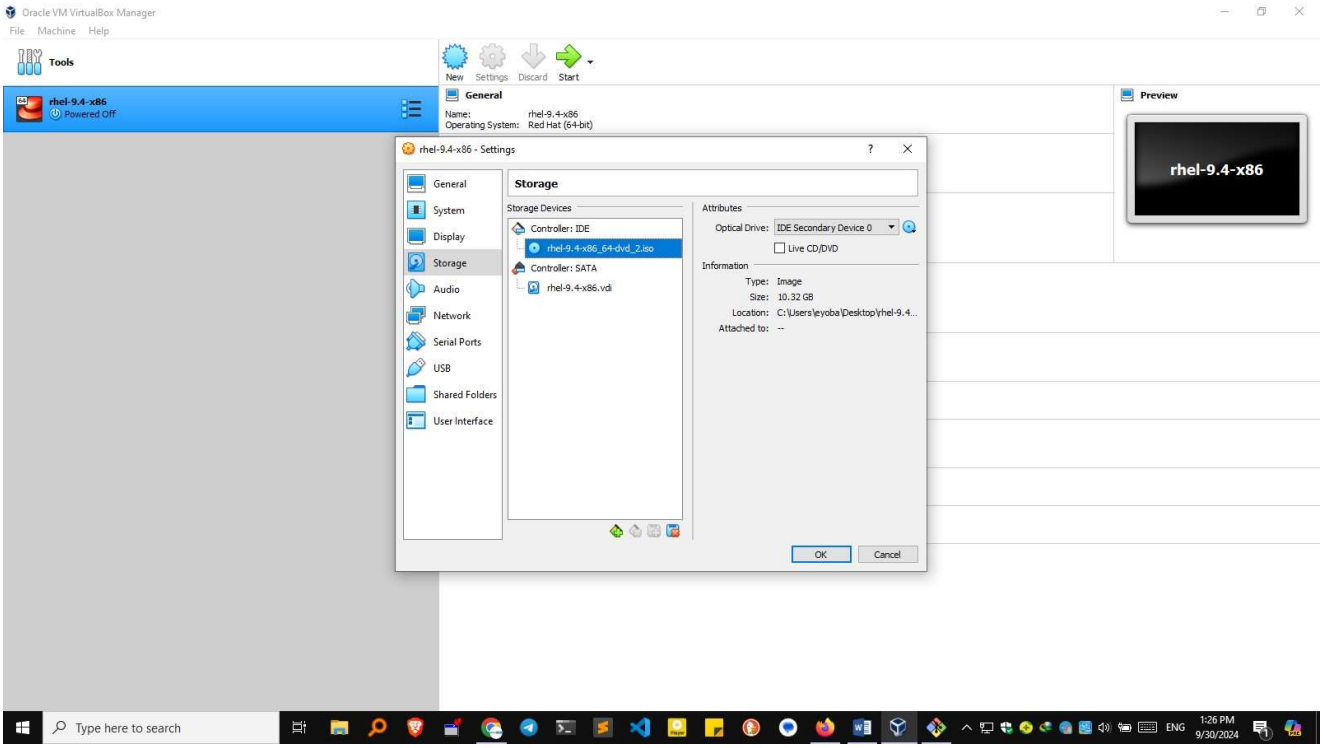
- Then enter “rhel” for the name section it the virtualbox will automatically understand what we are trying to install and it will Choose the type and version by itself if there is anything we want to change we can update it but for now let’s click next and go to the next step.



- Configure the hardware settings for the VirtualBox:
    - Allocate **2 4 CPUs**.
    - Set **4 GB of RAM**.
    - Choose **20 50 GB of disk space**.

Step 4: Boot the VirtualBox with the RHEL iso

- Now we can add the iso file by going to settings > storage and under controller we find empty disk when we click that we have an option to add the iso file
- After configuring the virtual machine's hardware and attaching the RHEL ISO image as the boot device, you can now proceed to start the VirtualBox.
- **VirtualBox**, click **Start** or **Power On** to boot the VirtualBox. The VirtualBox should boot directly into the **Red Hat Enterprise Linux Installer**.

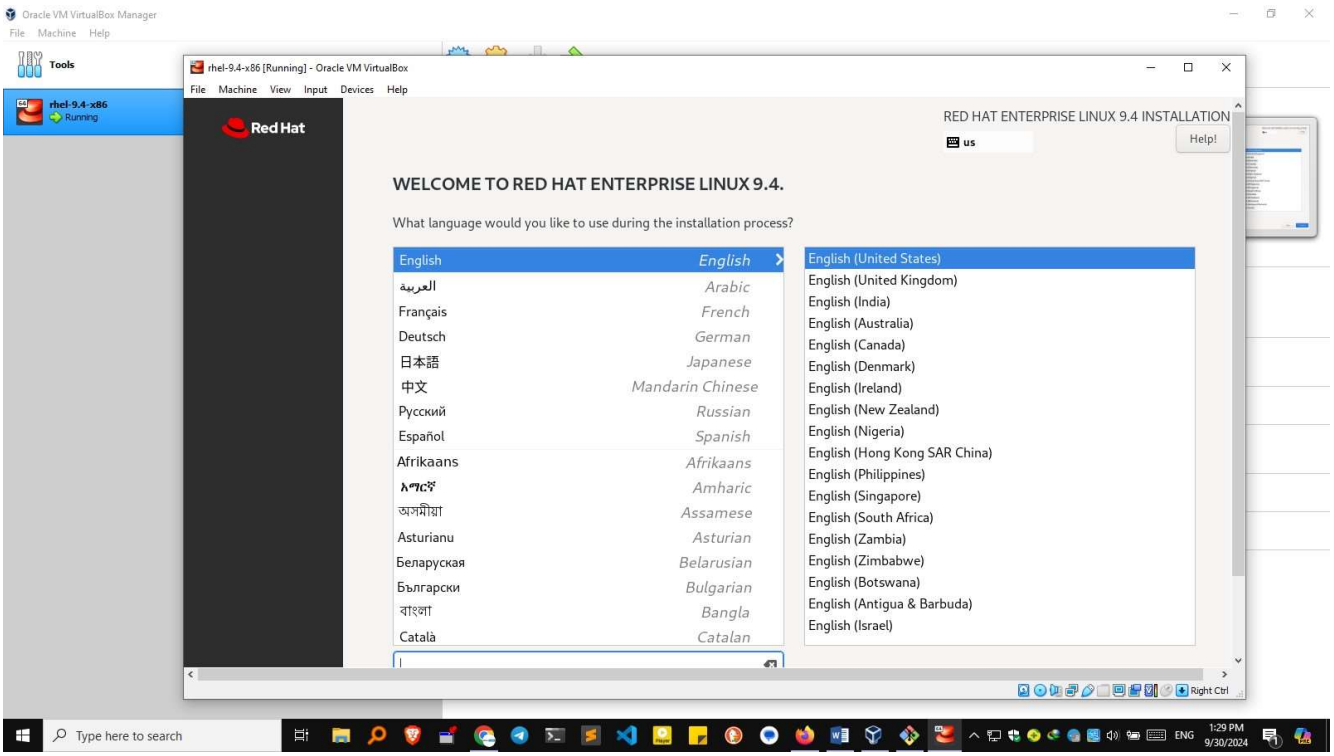


- At this stage, you should be presented with an option to either **Install Red Hat Enterprise Linux** or **Test Media and Install RHEL**. If you are confident that the downloaded ISO is fine, select **Install Red Hat Enterprise Linux** and press **Enter**.

Step 5: Language and Keyboard Layout Selection

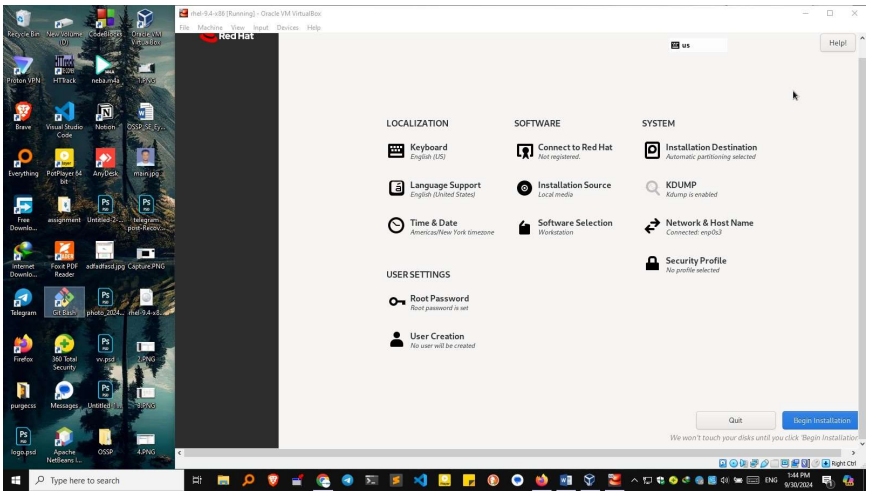
- Once the installer boots, you will be asked to select the **language** and **keyboard layout** for the installation process. The default language is typically **English (United States)**, but other languages can be selected if necessary.

- After choosing your preferred language, click **Continue** to proceed.



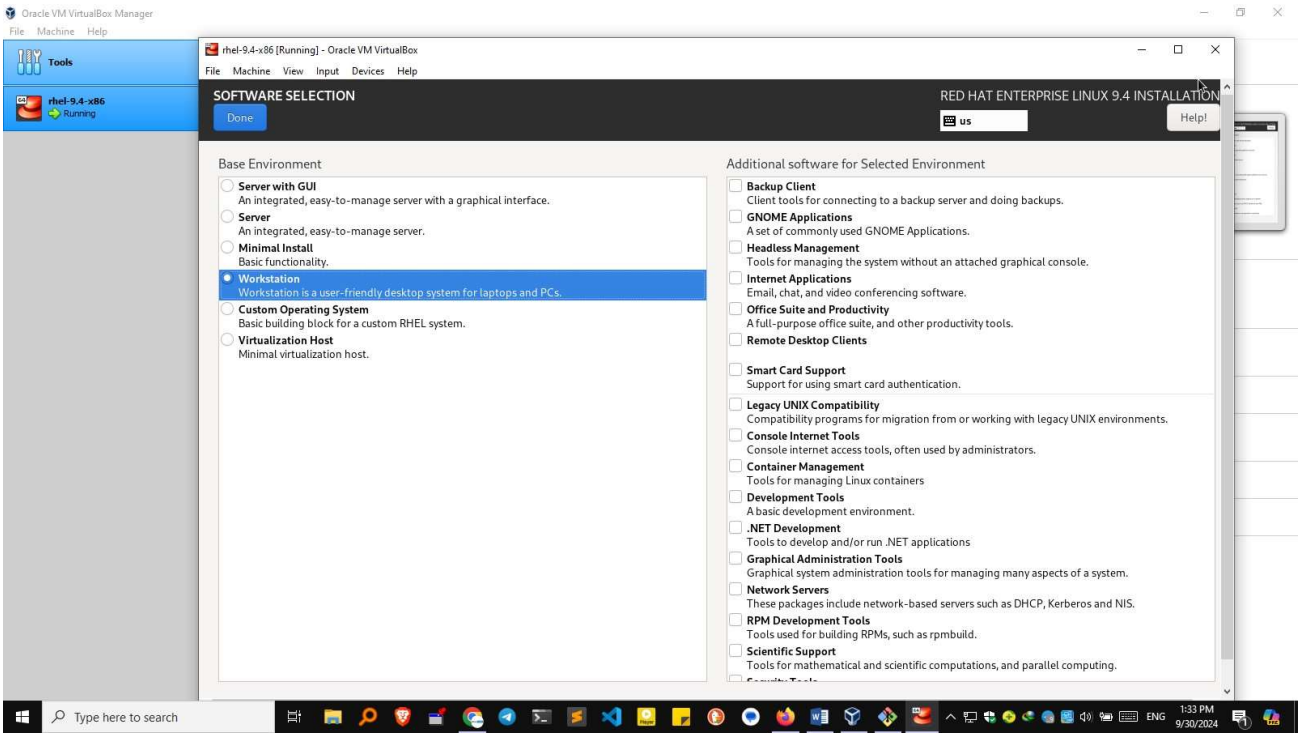
Step 6: Installation Summary – Customize the Installation

- The next screen you will encounter is the **Installation Summary** screen. This page allows you to configure critical aspects of the installation, such as **Date & Time**, **Software Selection**, **Installation Destination**, **Network & Hostname**, and more.



- **Date & Time**: Verify the time zone and adjust it if necessary. Ensure that the system clock is synchronized correctly for your location.
- **Software Selection**: Here, you can choose the type of RHEL installation you want:
  - **Server with GUI**: A server focused installation with a graphical user interface (GUI), ideal for users who want to manage the server visually.
  - **Minimal Installation**: A minimalistic installation with essential tools and packages, often preferred for production servers.
  - Other environments such as **Workstation** or **Custom** can be selected if needed.

For the purpose of this project, selecting i will choose **workstation** but I recommend **Server with GUI** because it offers a user friendly experience and all essential tools for learning I choose workstation because I wanted to test myself .



Step 7: Partitioning the Disk – Filesystem Selection

- In the **Installation Destination** section, you'll configure the disk partitioning. You can opt for **Automatic Partitioning** or **Custom Partitioning** if you want to have full control over the partition layout since we already already Partitioned it on virtualbox we will skip this step
  - For most users, **Automatic Partitioning** is sufficient. This automatically creates root (/), swap, and boot partitions, typically using the **ext4** filesystem.

- If you choose **Custom Partitioning**, you can manually allocate space to the root (/), home (/home), and other directories. You can also choose which filesystem to use. **ext4** is the default option, known for its reliability and speed, but **XFS** or **Btrfs** can be chosen for specific performance or scalability needs.

- **Filesystem Explanation**:
  - **ext4** (Fourth Extended Filesystem): The default choice for many Linux distributions, **ext4** is stable, reliable, and performs well under a variety of workloads.
  - **XFS**: A high performance filesystem designed for large scale storage environments. It is known for its excellent scalability, particularly with large files.
  - **Btrfs**: A newer filesystem with advanced features like snapshots and self healing capabilities, though it is still less common in enterprise environments compared to **ext4** and **XFS**.

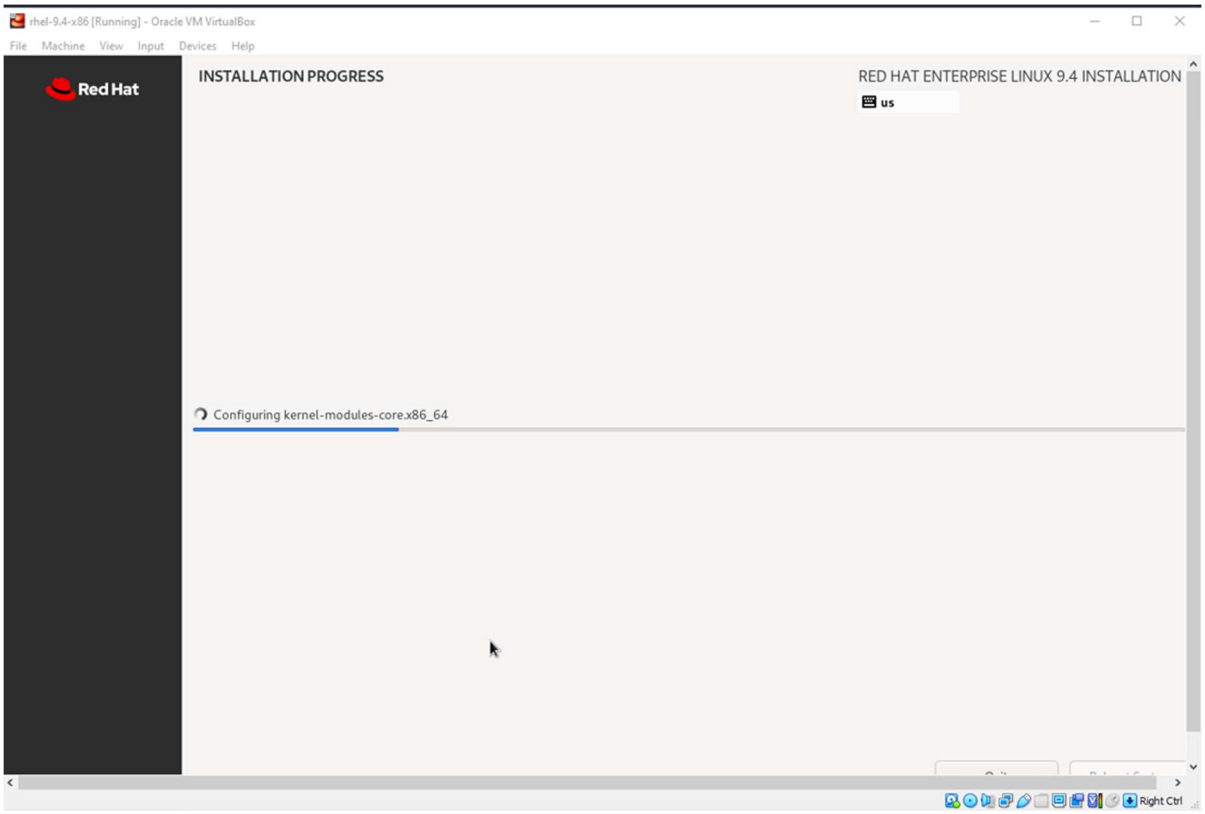
  Recommendation: For this project, using **ext4** is the best option because of its widespread use, compatibility, and performance balance.

Step 8: Network Configuration

- If your VM has network connectivity (either through NAT or Bridged networking), you can configure the network settings at this point. By default, the network configuration should be handled automatically via **DHCP**, but you can opt to configure static IP addresses if needed for your setup.
- Additionally, set the **Hostname** for your VM. The hostname is a unique identifier for your system within a network and can be customized to your preference.

Step 9: Begin Installation

- Once all settings have been configured, click the **Begin Installation** button at the bottom of the screen. The installer will now format the disk, apply the partitioning scheme, and install the selected software packages.

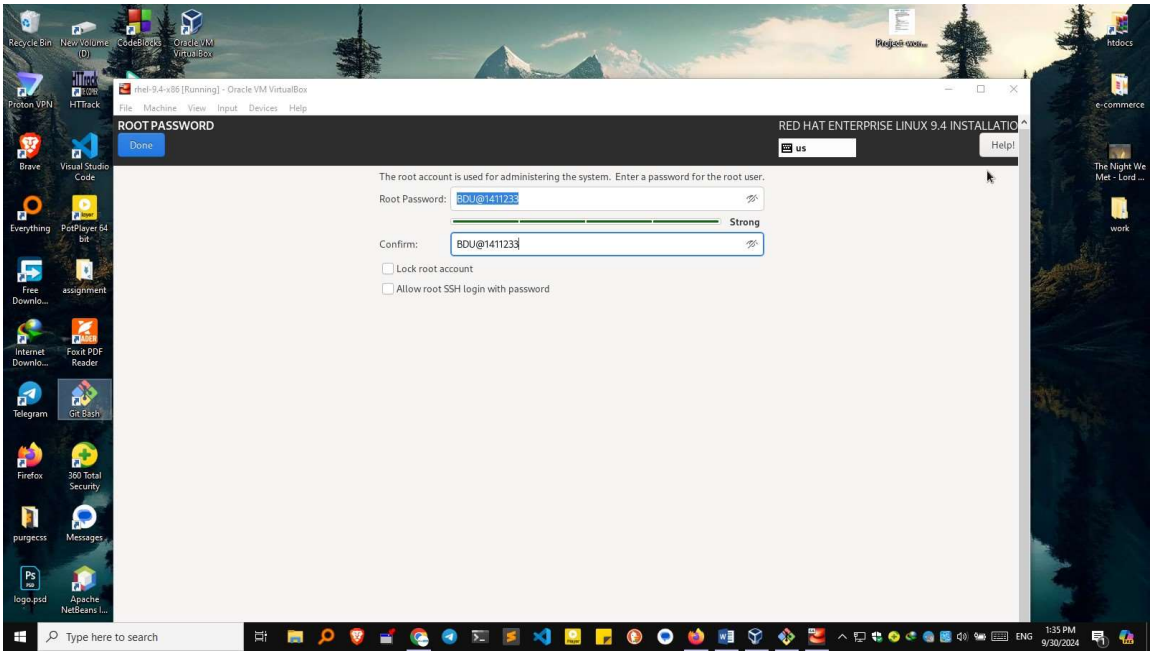


Step 10: Create User Account and Set Root Password

- While the software installation is underway, you'll be asked to create a **root password** and a **user account**.
    - The **root account** is the system administrator account and should have a strong password.
    - When creating the user account, ensure that the **Full Name** field is set to your name, as required by the project.

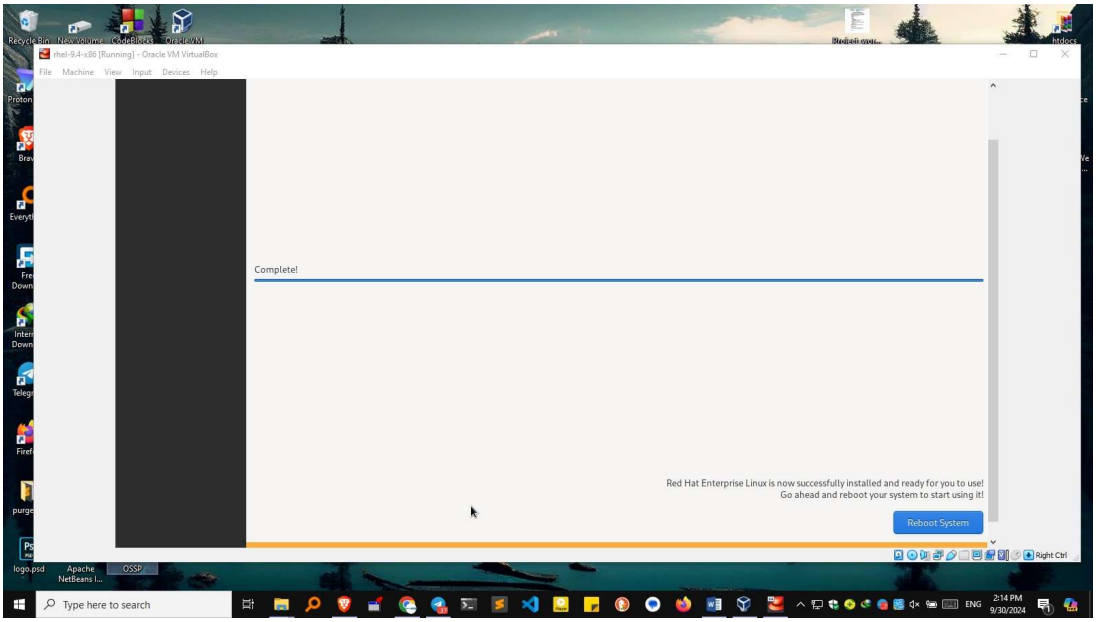
Best Practices for Root Password:

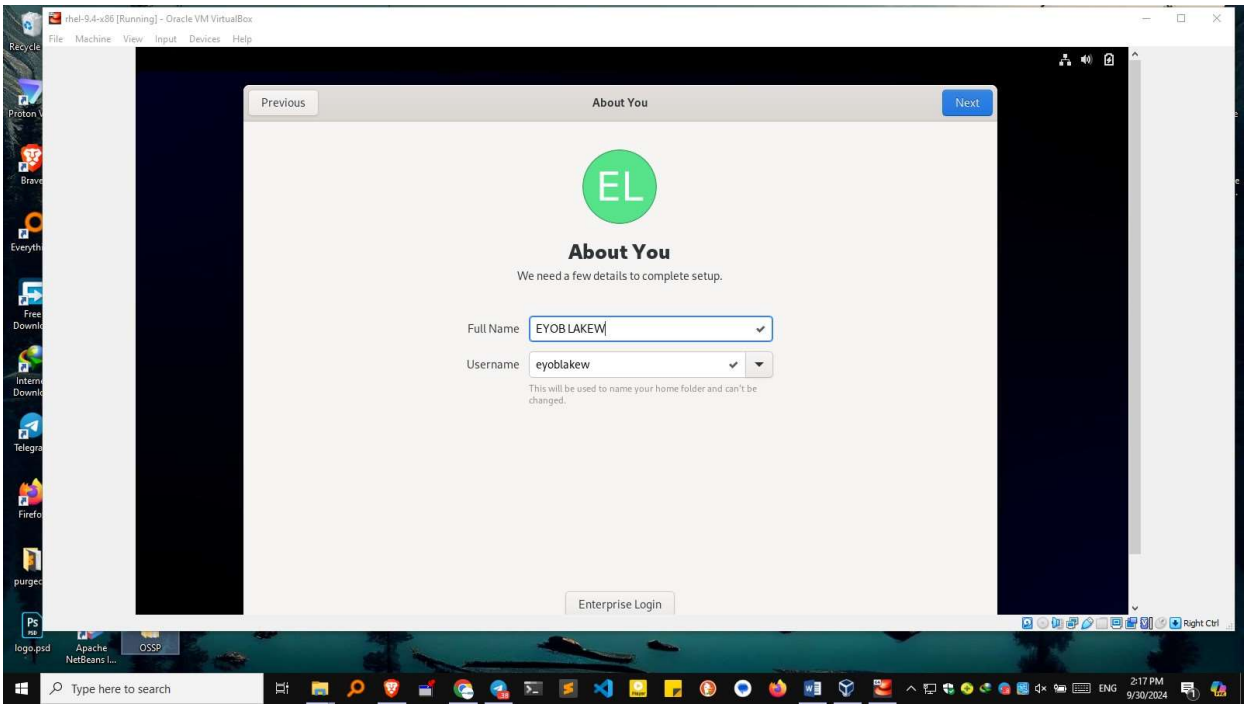
- Use a combination of upper and lowercase letters, numbers, and special characters.
- The password should be at least 8 characters long and difficult to guess.

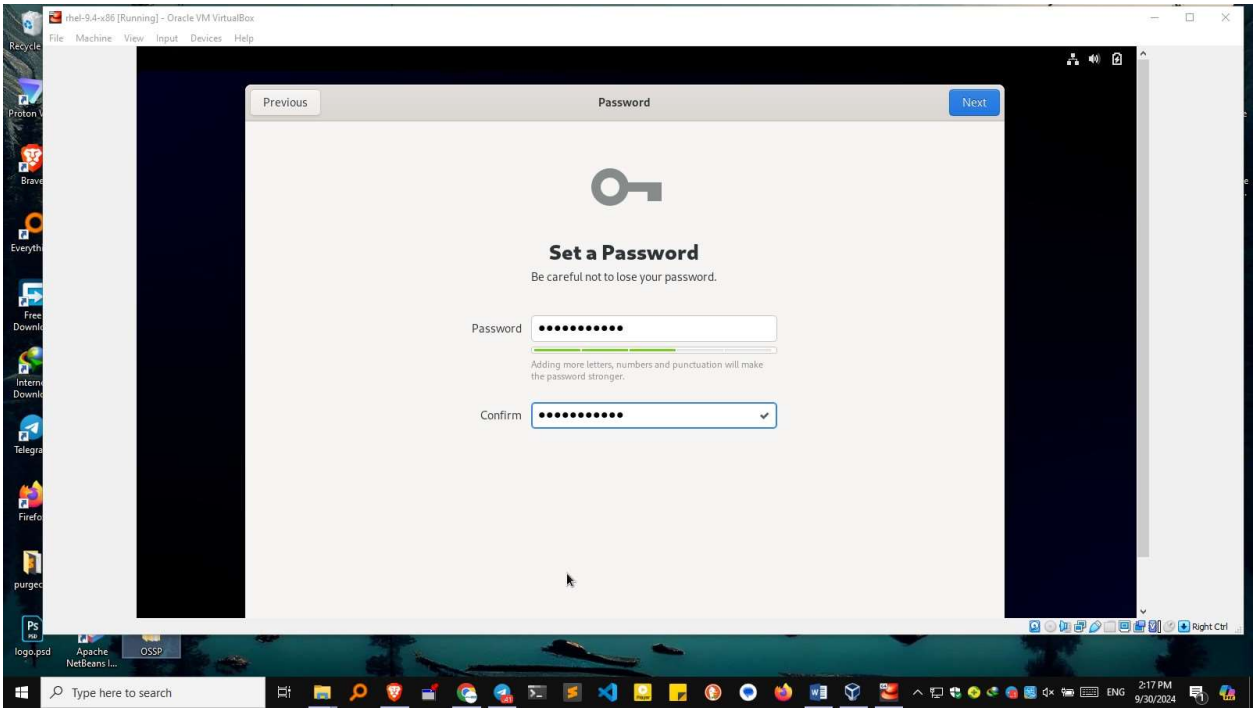


Step 11: Complete the Installation and Reboot

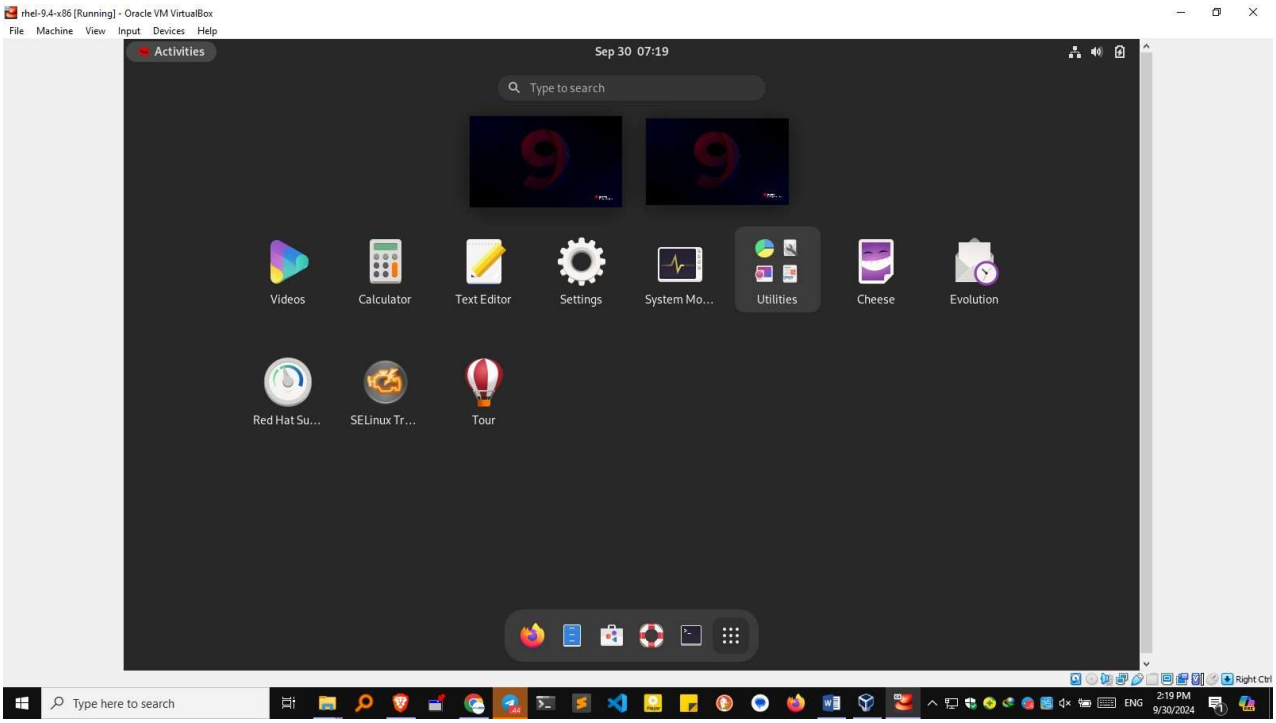
- After the installation completes, you will be prompted to **reboot the system**. Upon rebooting, the system will boot into the installed RHEL environment.




rhel-9.4-x86 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Previous
About You
Next
EL
About You
We need a few details to complete setup.
Full Name
EYOB LAKEW
Username
eyoblakew
This will be used to name your home folder and can't be changed.
Enterprise Login
Right Ctrl
Type here to search
ENG
2:17 PM
9/30/2024


rhel-9.4-x86 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Previous
Password
Next
Set a Password
Be careful not to lose your password.
Password
Adding more letters, numbers and punctuation will make the password stronger.
Confirm
Right Ctrl
Type here to search
ENG
2:17 PM
9/30/2024

- You'll be asked to log in either with the root account or the newly created user account. For general usage, it's recommended to log in with the user account to avoid using the root account unnecessarily. Now we have finished it



5. Issues Encountered

Installation Issue: Internet Connection Problems

During the installation process, the only significant issue I encountered was related to my internet connection. Since the setup file was 10 GB, it was challenging to download it using my home Wi Fi. I experienced multiple frustrations:

- The first attempt stalled at 80% after hours of waiting.
- On the second try, it stopped at 46%.

6. Solutions

After these setbacks, I decided to go to Poli Cumpas for a better connection. Fortunately, there I was able to successfully complete the download without further issues.

7. Filesystem Support

The Red Hat Enterprise Linux (RHEL) installer offers several filesystem options, and selecting the right one depends on the use case, performance requirements, and system constraints. Below is a detailed explanation of the different filesystems supported by RHEL and why each one might be chosen.

1. ext4 (Fourth Extended Filesystem)

- **Description**: The **ext4** filesystem is an evolution of the earlier ext3 filesystem. It is widely used in many Linux distributions due to its reliability, efficiency, and broad support. It balances performance and features, making it the default choice for RHEL installations.
- **Features**:
  - **Backward compatibility** with ext3 and ext2.
  - **Improved performance** over ext3, especially in terms of faster file system checks (fsck) and support for larger file sizes (up to 16TB) and volume sizes (up to 1EB).
  - **Journaled filesystem**, meaning it keeps a record of changes not yet committed to the filesystem, which helps in recovery after crashes.
- **Why choose ext4**:
  - **Stability**: ext4 is a mature and well supported filesystem, making it a safe choice for both personal and enterprise environments.
  - **Performance**: Although it is not the fastest filesystem, its performance is more than sufficient for most server and desktop workloads.
  - **Ease of use**: ext4 is well documented, with plenty of community support and tools available for troubleshooting and configuration.
- **Use cases**: ext4 is ideal for general purpose systems, such as web servers, databases, or desktop workstations, where performance is important but not the highest priority.

2. XFS

- **Description**: **XFS** is a high performance 64 bit filesystem that was developed by Silicon Graphics in 1993. It has become one of the default filesystems for enterprise Linux distributions, including RHEL, especially for large scale storage environments.
- **Features**:
  - **High scalability**: XFS is designed to handle large files and directories efficiently. It supports very large file sizes (up to 8 exabytes) and is optimized for multi threaded workloads.
  - **Efficient file allocation**: It uses an extent based allocation system, which reduces fragmentation and improves performance for large files.

    - **Journaling**: Like ext4, XFS uses journaling to ensure filesystem integrity and quick recovery after crashes.
    - **Real time sub volume support**: XFS allows for real time operations, which are useful in environments requiring consistent performance, such as media streaming.
- **Why choose XFS**:
    - **Performance**: XFS is highly optimized for parallel I/O operations, making it ideal for high performance systems where I/O speed is critical, such as in data warehouses, large scale databases, or media servers.
    - **Large files**: XFS is particularly suitable for systems that manage very large files, such as multimedia content, backup systems, or data analytics.
    - **Enterprise use**: RHEL, by default, uses XFS for its root filesystems in server environments, indicating its suitability for demanding enterprise workloads.
- **Use cases**: XFS is best suited for systems requiring high throughput and efficient handling of large files, such as video editing servers, scientific computing, and big data systems.

3. Btrfs (B Tree Filesystem)

- **Description**: **Btrfs** is a modern copy on write (CoW) filesystem that offers advanced features not available in older filesystems like ext4 and XFS. It is designed for flexibility, scalability, and easy management of storage. Despite its advanced features, Btrfs is still considered experimental in some cases, though it has been gaining stability.
- **Features**:
    - **Copy on write**: Btrfs uses a copy on write mechanism, which means it creates new data copies rather than overwriting existing data. This allows for features like snapshots and cloning.
    - **Snapshots and subvolumes**: One of the most powerful features of Btrfs is the ability to take snapshots of the filesystem at any point in time, making it easy to roll back to a previous state.
    - **Data integrity**: Btrfs includes checksums for data and metadata, allowing for automatic error detection and correction, which enhances data integrity.
    - **Compression**: Btrfs supports inline data compression, which can help save storage space, particularly for systems handling text based files or databases.
- **Why choose Btrfs**:
    - **Snapshots**: If frequent backups or rollbacks are needed, Btrfs's snapshot capability is very useful. It allows users to easily revert to a previous state in case of failure or data corruption.
    - **Data integrity**: For systems that prioritize data accuracy and reliability, such as in scientific research or legal databases, Btrfs offers a high level of data integrity checking and repair tools.

    - **Scalability**: Btrfs supports scaling across multiple devices, making it a good choice for systems where storage needs may grow dynamically over time.
- **Use cases**: Btrfs is ideal for systems that require advanced data management, such as database servers, virtual machines, or systems that require frequent snapshots (e.g., development and testing environments).

4. Other Filesystems Supported by RHEL

- **NTFS**: Though primarily a Windows filesystem, RHEL can read and write to NTFS partitions through external libraries and drivers (such as **ntfs 3g**). It is used primarily for interoperability between Linux and Windows systems, especially on dual boot machines or shared external drives.
- **FAT32 and exFAT**: These filesystems are mostly used for compatibility with external storage devices (such as USB drives) that need to be accessed by multiple operating systems, including Windows and macOS. FAT32 is an older filesystem with limitations on file size (up to 4GB), while **exFAT** supports larger files but lacks some features like journaling.

5. Choosing the Right Filesystem for RHEL Installation

For this installation project, choosing **ext4** is the best option for general use. It is the default for many RHEL installations and offers the right balance of reliability, performance, and ease of use. While **XFS** is a strong candidate for enterprise environments requiring high throughput, **ext4** is more commonly used in educational and development environments due to its flexibility and support across different Linux distributions.

8. Advantages and Disadvantages of Red Hat Enterprise Linux (RHEL)

RHEL, like any operating system, has both advantages and disadvantages. Understanding these aspects is crucial when deciding whether RHEL is the right choice for your projects or enterprise needs.

1. Advantages of RHEL

- **Enterprise Grade Stability and Support**: One of the biggest advantages of RHEL is its stability. Red Hat ensures that RHEL undergoes rigorous testing before release, making it a highly reliable platform for mission critical applications. Additionally, Red Hat offers extensive **technical support** and **security updates** as part of its subscription model.
- **Security**: RHEL comes with robust security features such as **SELinux (Security Enhanced Linux)**, which provides mandatory access controls (MAC) to enhance system

security. Red Hat also provides timely security patches, ensuring that the system is protected against vulnerabilities.

- **Long Term Support**: RHEL offers **up to 10 years of support** for each major release, making it a great choice for businesses that require long term stability and do not want to frequently upgrade their systems. This long term support includes security patches, bug fixes, and technical support.
- **Enterprise Integration**: RHEL is designed to integrate smoothly with various enterprise solutions, including database management systems (e.g., MySQL, PostgreSQL), cloud platforms (AWS, Azure), and virtualization technologies (KVM, VMware).
- **Performance**: RHEL is optimized for performance in enterprise environments. Whether used for running web servers, databases, or other workloads, RHEL can handle large scale applications and manage resources efficiently.
- **Package Management with YUM/DNF**: The **YUM** package manager (or **DNF** in newer versions) simplifies software management. It allows easy installation, updating, and removal of software packages, making system maintenance straightforward.
- **Compliance and Certification**: Red Hat is certified for various industry standards (such as **FIPS**, **HIPAA**, and **PCI DSS**), making it an ideal choice for organizations that must meet strict regulatory compliance requirements.

2. Disadvantages of RHEL

- **Subscription Costs**: One of the most cited disadvantages of RHEL is its cost. Unlike some other Linux distributions (such as Ubuntu or CentOS), RHEL requires a **subscription** for full access to updates, patches, and Red Hat's support services. This can be a barrier for small businesses or developers looking for free solutions.
- **Learning Curve**: Although RHEL is user friendly for those experienced with Linux, it can be daunting for new users. The focus on enterprise features and the command line interface can make it harder to pick up for beginners compared to more desktop oriented Linux distributions like **Ubuntu** or **Fedora**.
- **Limited Desktop Focus**: While RHEL can be used as a desktop operating system, it is primarily designed for servers and enterprise environments. As such, it lacks some of the user friendly features and pre installed software found in desktop distributions like Ubuntu or Linux Mint.
- **Slower Release Cycles**: RHEL follows a slower release cycle compared to some other Linux distributions. While this ensures stability and reliability, it can mean that newer features and software packages are not available immediately.
- **Proprietary Drivers and Software**: RHEL's strict focus on open source software can sometimes be limiting when proprietary drivers or software are needed. While there are workarounds, users may have to go through additional steps to install things like proprietary graphics drivers or multimedia codecs.

9. Conclusion

Installing and working with **Red Hat Enterprise Linux (RHEL)** in a virtual environment provides a deep understanding of how enterprise grade operating systems function. The project introduced the various steps involved in setting up a virtual machine, installing RHEL, and configuring the system post installation. Throughout the process, I explored different filesystems supported by RHEL, with a focus on choosing **ext4** for general use due to its reliability and performance.

By understanding both the advantages and disadvantages of RHEL, we can appreciate its significance in enterprise environments. The system's robust security features, long term support, and powerful performance optimization make it an industry leader. However, challenges like subscription costs and a steeper learning curve mean that RHEL is best suited for enterprise environments rather than casual use.

The hands on installation and configuration tasks, along with troubleshooting common issues, have equipped us with the technical skills needed to manage RHEL systems in real world scenarios. This experience lays the groundwork for further exploration of RHEL's powerful tools, including system administration, networking, and security management.

10. Future Outlook / Recommendations

As I look forward, there are several areas in which knowledge and expertise in Red Hat Enterprise Linux can be expanded:

- **Advanced System Administration**: Future work should focus on learning advanced system administration tasks in RHEL, such as user and group management, system monitoring, performance tuning, and automation using **Ansible** or shell scripts.
- **RHEL in the Cloud**: As cloud computing becomes more prevalent, learning how to deploy and manage RHEL systems in cloud environments like **AWS**, **Microsoft Azure**, and **Google Cloud Platform** can be invaluable. Red Hat's strong integration with these platforms makes RHEL an excellent choice for cloud based applications.
- **Containerization with RHEL**: Understanding how RHEL supports modern development and deployment techniques such as containerization and microservices using **Docker** or **Podman** is critical. RHEL is commonly used in environments that deploy containerized applications due to its stability and support for **OpenShift**, Red Hat's Kubernetes distribution.
- **Security and Compliance**: Future exploration could focus on enhancing RHEL system security by using tools like **SELinux** and **firewalld**, as well as ensuring compliance with industry standards.

- **Exploring Alternatives**: While RHEL is a powerful system, it is also useful to experiment with **CentOS Stream** (the upstream version of RHEL) or other distributions like **Fedora** and **Ubuntu** to compare and contrast their features and performance in different environments.

11. How Virtualization Works in Modern Operating Systems

Virtualization is made possible through several key technologies that allow the abstraction of hardware resources and efficient management of multiple operating systems.

1. The Role of the Hypervisor

The hypervisor is the core technology behind virtualization. There are two main types of hypervisors:

- **Type 1 (Bare metal Hypervisor)**: This type runs directly on the physical hardware without needing a host OS. It provides better performance because it doesn't need to go through another layer (host OS) to access hardware. Examples include **VMware ESXi**, **Microsoft Hyper V**, and **Xen**.
- **Type 2 (Hosted Hypervisor)**: This type runs on top of an existing operating system (the host OS). It is easier to set up and use, but the additional layer (the host OS) can introduce some performance overhead. Examples include **Oracle VM VirtualBox** and **VMware Workstation**.

2. Hardware Virtualization

Modern processors from Intel and AMD include hardware support for virtualization (e.g., **Intel VT x** and **AMD V**). These hardware extensions allow VMs to run directly on the CPU, reducing the overhead associated with software emulation and improving performance.

Key Features of Hardware Virtualization:

- **Ring Levels**: Modern CPUs operate at different privilege levels (rings). Typically, operating systems run in ring 0 (kernel space) for direct access to hardware, while user applications run in ring 3 (user space). Virtualization adds an additional layer, allowing the hypervisor to run in ring 1, giving it control over both the guest OS and the hardware.
- **Nested Page Tables**: These allow the hypervisor to efficiently manage memory for VMs, reducing the performance overhead associated with memory management.

3. Virtual Machine Monitors (VMMs)

The hypervisor manages **Virtual Machine Monitors (VMMs)**, which are responsible for ensuring that each VM behaves as if it were running on physical hardware. VMMs handle tasks such as memory management, CPU scheduling, and I/O operations for each VM.

Options in Virtualization

There are several approaches and tools available for virtualization in modern operating systems. The choice of technology depends on the requirements, such as the type of workloads, performance needs, and available hardware.

1. VMware

- **VMware ESXi**: A bare metal hypervisor that provides enterprise grade virtualization. It offers features such as live migration (vMotion), high availability, and fault tolerance.
- **VMware Workstation/Player**: A Type 2 hypervisor that runs on top of an operating system. It is popular for desktop virtualization and testing environments.

2. KVM (Kernel based Virtual Machine)

- **KVM** is a Linux based virtualization solution built into the Linux kernel. It turns the Linux kernel into a hypervisor, allowing the host machine to run multiple virtual machines. KVM is known for its performance and is commonly used in enterprise and cloud environments.

3. Microsoft Hyper V

- **Hyper V** is a Type 1 hypervisor built into Windows Server and Windows 10 Pro/Enterprise editions. It is commonly used in Windows environments for server and desktop virtualization.

4. Oracle VM VirtualBox

- **VirtualBox** is a Type 2 hypervisor that supports a wide range of guest operating systems. It is open source and popular for personal use, testing, and development environments.

Primitive System Data Types in Virtualization

In the context of virtualization, primitive system data types refer to the fundamental hardware resources that are virtualized and abstracted by the hypervisor. These include:

1. **CPU**: The hypervisor allocates virtual CPUs (vCPUs) to each virtual machine. The vCPUs are time shared among the guest operating systems, allowing multiple VMs to use the same physical CPU cores efficiently.

2. **Memory**: Each VM is allocated a portion of the system's RAM. The hypervisor uses techniques like **memory ballooning** and **overcommitting** to manage memory usage across multiple VMs.
3. **Storage**: Virtual machines use **virtual disks**, which are files stored on the host machine. These files represent the VM's hard drive and can be resized, moved, or backed up like regular files. The hypervisor ensures that each VM has access to its virtual disk while abstracting the physical storage device.
4. **Network**: Virtual machines are connected to virtual networks created by the hypervisor. These virtual networks can be isolated or connected to the host's network, allowing VMs to communicate with each other or external systems. Virtual network interfaces are assigned to each VM, mimicking physical network cards.

Conflicts between Standards in Virtualization

Virtualization technologies are governed by a variety of standards that sometimes conflict with one another. These conflicts can arise in terms of interoperability, performance, or management. Some of the common conflicts between virtualization standards include:

1. **Virtual Machine File Formats**:
   - Different hypervisors use different file formats for virtual machine images. For example, **VMware** uses the **VMDK** format, while **KVM** uses **QCOW2**. This can create issues when migrating VMs from one platform to another.
   - **Solution**: Conversion tools like **qemu img** can convert between different VM formats, though this can be a complex process with potential for data loss or performance degradation.
2. **Network Virtualization Standards**:
   - Different vendors use different methods for implementing virtual networking. For example, **VMware NSX** and **OpenStack Neutron** use different approaches to software defined networking (SDN), leading to compatibility issues when integrating solutions across platforms.
   - **Solution**: Open standards such as **OVS (Open vSwitch)** aim to provide a standard networking layer that can work across multiple platforms.
3. **Cloud APIs**:
   - Cloud platforms like **AWS**, **Azure**, and **Google Cloud** each have their own proprietary APIs for managing virtual machines and services. These differences make it difficult to move workloads between clouds or integrate multi cloud solutions.
   - **Solution**: Tools like **Terraform** and **Ansible** help bridge these gaps by providing a unified interface for provisioning resources across different cloud providers.
4. **Security Standards**:

- Different virtualization platforms implement security features in varying ways. For example, **SELinux** and **AppArmor** are two different Linux security modules that may conflict with certain virtualization tools.
- **Solution**: It's important to understand the security models of each virtualization platform and ensure compatibility with the underlying OS's security mechanisms.

11. System Call Implementation – dup2

In this part of the project, we will dive into the implementation and detailed workings of the **dup2** system call in Red Hat Enterprise Linux (RHEL). We'll discuss its importance, its usage in system programming, and how to implement and test it in a C program.

---

a. What is dup2?

The dup2 system call is used in Unix like operating systems, including Linux, to duplicate file descriptors. File descriptors are integers that uniquely identify an open file within a process. Every process has a set of standard file descriptors:

- **0**: Standard input (stdin)
- **1**: Standard output (stdout)
- **2**: Standard error (stderr)

dup2 provides the ability to duplicate a file descriptor to a specific value. In practice, it's commonly used in shell redirection and system programming to control input and output streams.

System Call Signature:

int dup2(int oldfd, int newfd);

- **oldfd**: The file descriptor that you want to duplicate.
- **newfd**: The file descriptor you want oldfd to be duplicated to. If newfd is already open, dup2 closes it before duplicating oldfd.

Return Value:

- On success, dup2 returns the value of newfd.
- On failure, it returns 1, and sets errno to indicate the error.

Purpose:

- The dup2 system call is used primarily for redirecting file descriptors in system programming. For instance, in shell scripts or programs, it allows redirecting the standard output (stdout) or standard error (stderr) to a file or another output stream.

Example Use Case:

- Redirecting standard output (file descriptor 1) to a file, so that the output of a program gets written to a file instead of the terminal.

---

b. Why Use dup2?

The dup2 system call is crucial in several programming contexts, particularly when dealing with input/output redirection, inter process communication (IPC), and forking processes. Below are a few key reasons why dup2 is important:

1. **Redirection of Input/Output**:
    - In command line environments, you often need to redirect the output of a program to a file or redirect input from a file. dup2 allows you to reassign a file descriptor like stdout or stderr to point to a file, enabling you to capture output or error messages.
2. **Forking Processes and Executing Commands**:
    - In systems programming, it is common to use the fork() system call to create a child process. The child process can inherit the parent's file descriptors. Using dup2, you can redirect the input or output streams in the child process to different file descriptors before executing another program via exec().
3. **Simplified File Handling**:
    - Instead of writing a custom function for file redirection, dup2 offers a straightforward way to manage multiple file descriptors and close them efficiently when needed.
4. **Socket Programming**:
    - In network programming, dup2 can be used to redirect standard input/output to a network socket, making it possible to pipe data between programs over a network connection.

---

c. How dup2 Works Internally

When dup2 is invoked, the kernel follows these steps:

1. **Check the validity of oldfd**:

- The kernel verifies that oldfd refers to an open file descriptor. If oldfd is invalid, the system call fails and sets the errno variable accordingly.

2. **Close newfd if necessary**:
   - If newfd is already open and points to a different file, the kernel closes newfd before duplicating oldfd to it. This ensures that no file descriptor leaks occur (i.e., keeping unnecessary files open).

   Note: If newfd is the same as oldfd, the system call does nothing and immediately returns the value of newfd.

3. **Duplicate oldfd to newfd**:
   - The file descriptor oldfd is copied to newfd, and the new descriptor is ready to use. The underlying file structure now has two references: one for oldfd and another for newfd, both pointing to the same file.
4. **Return the newfd**:
   - If everything is successful, the value of newfd is returned. On failure, 1 is returned, and errno is set to indicate the specific error.

Common Errors:

- **EBADF**: One or both of the file descriptors is not valid.
- **EMFILE**: The process has too many open files, and cannot open any more file descriptors.

---

d. Implementation of dup2 in C

Let's implement a simple C program that demonstrates how the dup2 system call works. This program will open a file, duplicate its file descriptor, and redirect standard output (stdout) to the file using dup2.

Code Example:

```
#include <stdio.h>
#include <fcntl.h>   // For open()
#include <unistd.h>  // For dup2(), close()

int main() {
   // Step 1: Open a file to redirect stdout to it
   int file_fd = open("output.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
   if (file_fd ==  1) {
```

```
        perror("Error opening file");
        return 1;
    }

    // Step 2: Use dup2 to redirect stdout to the file
    if (dup2(file_fd, STDOUT_FILENO) ==  1) {
        perror("Error with dup2");
        close(file_fd);
        return 1;
    }

    // Step 3: Print a message, which will now be written to the file
    printf("This output will be redirected to the file 'output.txt'\n");

    // Step 4: Close the file descriptor
    close(file_fd);

    return 0;
}
```

Feel free to check the example alongside the result on my GitHub repository = > **https://github.com/eyobelakew/RHEL dup2 System Call Implementation**

Explanation:

1. **Opening the File**:
    - The program opens the file output.txt in write only mode. If the file doesn't exist, it is created (O_CREAT). The file is truncated (O_TRUNC), which means any previous content is deleted, and new content starts at the beginning.
2. **Calling dup2**:
    - The system call dup2(file_fd, STDOUT_FILENO) is used to duplicate the file descriptor of file_fd to the standard output (STDOUT_FILENO, which is file descriptor 1). After this call, anything written to stdout is redirected to output.txt.
3. **Writing Output**:
    - The program prints a message using printf(). However, because dup2 has redirected stdout, the message gets written to output.txt instead of the terminal.
4. **Closing File Descriptors**:
    - After completing the redirection, the file descriptor for output.txt is closed using close(file_fd).

Compilation and Execution:

To compile and run the program:

```
gcc  o dup2_example dup2_example.c
./dup2_example
```

- After running the program, check the contents of output.txt:

```
cat output.txt
```

You should see the text "This output will be redirected to the file 'output.txt'" printed in the file instead of on the terminal I have included all the results on github feel free to check them out.