HTU Environment Rebuild: Server Restoration and Configuration Report

*By: Ahmad AL-Quraan*

*Email:* [*202508067@htu.edu.jo*](mailto:202508067@htu.edu.jo)

***Date***: 0**5/01/2025**

[1. INTRODUCTION 4](#_Toc219379601)

[2. VM Provisioning & OS Configuration 4](#_Toc219379602)

[2.1 Virtual Hardware Allocation 5](#_Toc219379603)

[2.2 Language & Localization 6](#_Toc219379604)

[2.3 Disk & Partitioning 7](#_Toc219379605)

[2.4 User Configuration 8](#_Toc219379606)

[2.5 Software selection 9](#_Toc219379607)

[2.6 Automated Installation (Kickstart) 10](#_Toc219379608)

[3. Initial System Setup 11](#_Toc219379609)

[3.1 System Update 11](#_Toc219379610)

[3.2 Required Software Installation 11](#_Toc219379611)

[4. Storage Configuration (40GB Disk) 12](#_Toc219379612)

[4.1 Disk Partitioning and Layout 12](#_Toc219379613)

[4.2 LVM Configuration 14](#_Toc219379614)

[4.3 Filesystem Creation 15](#_Toc219379615)

[5. Departmental Directory Structure 18](#_Toc219379616)

[5.1 Department Directories 18](#_Toc219379617)

[5.2 Access Control and Permissions 18](#_Toc219379618)

[6. User Groups and Accounts 21](#_Toc219379619)

[7. Backup Mechanism Implementation 23](#_Toc219379620)

[7.1 Backup Script 23](#_Toc219379621)

[7.2 Automation 24](#_Toc219379622)

[8. Server Optimization with TuneD 25](#_Toc219379623)

[8.1 Installation and Enablement of TuneD 25](#_Toc219379624)

[8.2 Performance Profile Configuration 25](#_Toc219379625)

[9. System Identity & Package Management 27](#_Toc219379626)

[9.1 Hostname Configuration 27](#_Toc219379627)

[9.2 Local Hostname Resolution 28](#_Toc219379628)

[10. YUM Repository Configuration 29](#_Toc219379629)

[10.1 BaseOS Repository 29](#_Toc219379630)

[10.2 AppStream Repository 30](#_Toc219379631)

[11. Temporary HR Web Application Placeholder 31](#_Toc219379632)

[11.1 Web Content 31](#_Toc219379633)

[11.2 Web Service Configuration 32](#_Toc219379634)

[11.3 SELinux 33](#_Toc219379635)

[11.4 Firewall Configuration 35](#_Toc219379636)

[12. SSH Service Hardening 36](#_Toc219379637)

[12.1 SSH Key Authentication 36](#_Toc219379638)

[12.2 Disable Password Login for sysadmin 37](#_Toc219379639)

[12.3 Disable Password Authentication System-Wide 37](#_Toc219379640)

[12.4 Restrict SSH Access to Administrative Users 38](#_Toc219379641)

[Appendix A 39](#_Toc219379642)

[Appendix B 42](#_Toc219379643)

[Appendix C 43](#_Toc219379644)

[References 44](#_Toc219379645)

# INTRODUCTION

The goal of this project is to rebuild an **HTU Red Hat Enterprise Linux (RHEL)** server from scratch and configure it according to professional system administration practices. The task was assigned after a system outage caused by missing documentation and improper maintenance, which required a complete reinstallation of the server environment.

This project focuses on setting up a stable and secure multi-user Linux server that can support different departments within the organization. The work includes installing the operating system, configuring storage, managing users and permissions, and ensuring that essential services are properly deployed and secured.

The scope of this project includes:

* Automated and repeatable system installation.
* Proper storage configuration to ensure data integrity.
* User and group management with controlled access privileges.
* System reliability through updates, backups, and performance tuning.

By following these steps, the server is transformed from an unmaintained state into a structured, secure, and well-documented system. This report documents the configuration process and demonstrates that all technical requirements have been successfully implemented.

# VM Provisioning & OS Configuration

KVM (Kernel-based Virtual Machine) with virt-manager was used for this task instead of VirtualBox, as it provides better integration and performance on Linux-based systems. It’s type 1 hypervisor which means it’s works on kernel level, unlike oracle VM which is type 2 hypervisor and works on top of the OS, so KVM will have better performance.

## 2.1 Virtual Hardware Allocation

The virtual machine was created using **virt-manager**. The **RHEL 10 boot ISO** was selected as the installation media during the virtual machine creation process, as shown in Figure 1.

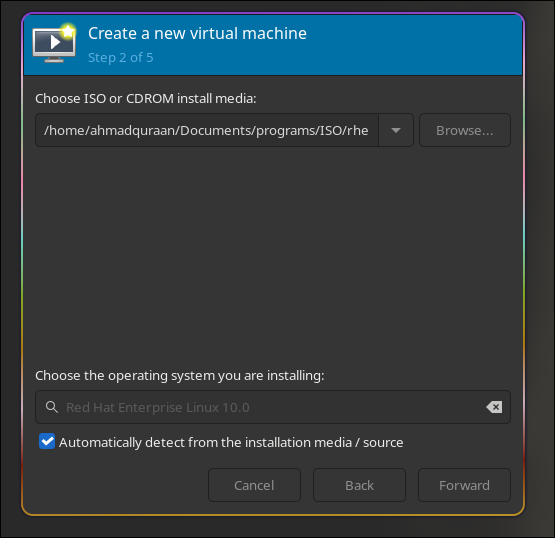


Figure 1: RHEL 10 iso selection in virt-manager.

The virtual machine was configured with 4 GB of RAM and 4 vCPUs, as shown in figure 2.

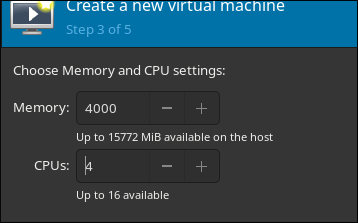


Figure 2: Adjust memory and Cpu.

Two virtual disks were configured for the virtual machine. A **20 GB disk** was allocated for the operating system installation, while an additional **40 GB disk** was added for data storage purposes, as shown in Figure 3.

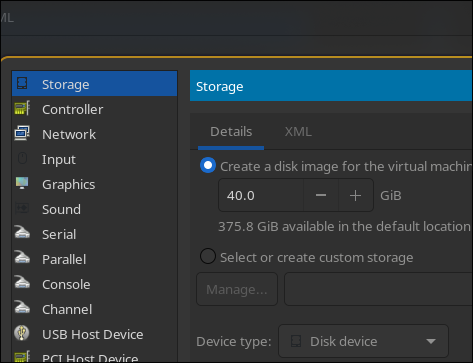


Figure 3: Choose a disk image of 20GB and another one of 40GB.

The VM is now ready, and RHEL opened for configuration as shown in figure 4.

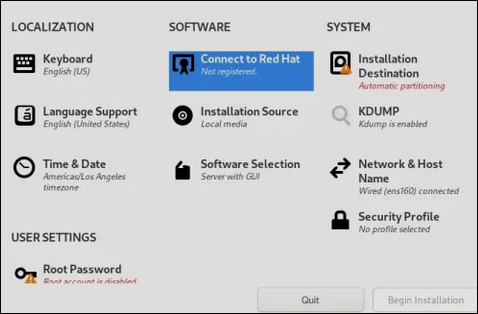
 

Figure 4: shows the VM starts to work.

## 2.2 Language & Localization

English was chosen as the main language, as shown in figure 5.

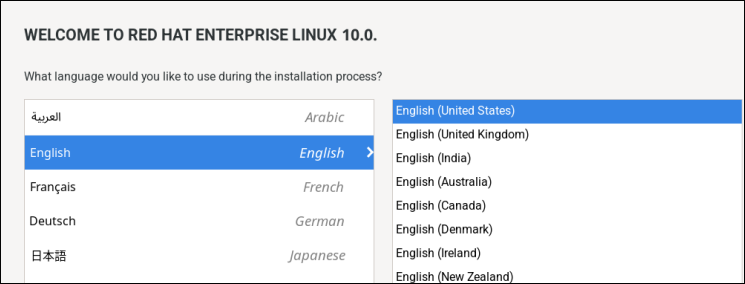


Figure 5: choose English lang.

**Asia/ Amman** time zone was chosen, as shown in figure 6.

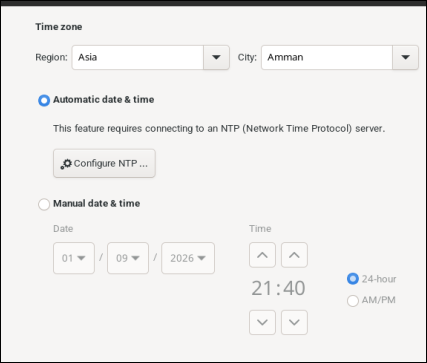


Figure 6: set the time zone Asia/ Amman

## 

## 2.3 Disk & Partitioning

The 20GB disk image we previously made was chosen to setup the OS on, as shown

in Figure 7.

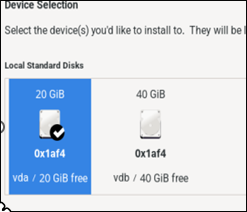


Figure 7: Set the 20GB disk as the main OS disk.

## 2.4 User Configuration

**Disabling** remote SSH access for the **root** account (shown in figure 8.) is a **critical security best practice**, because:

1. Root is a universal target, so attackers **don’t need to guess the username**, only the password or key.
2. Direct root access = total system compromise -> root has **unrestricted privileges.**

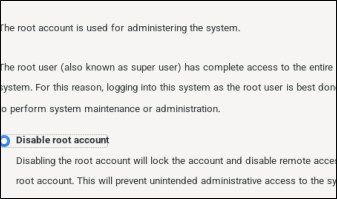


Figure 8: Disabling root account in OS configuration.

An administrative user account was created during the installation process to allow secure system administration without enabling direct root login, as shown in figure 9.

The user **sysadmin** was created and added to the **wheel** group (shown in figure 9), granting administrative privileges through the sudo command.

- Username: **sysadmin**

- Added to **wheel group** (Allow root privilege using sudo).

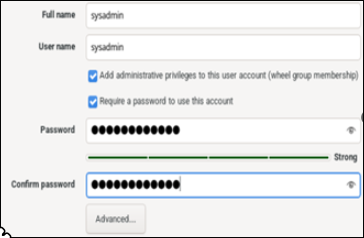


Figure 9: Set a user account with administrative access.

Red Hat Enterprise Linux **requires registering the system with Red Hat Subscription Management account**, in order to access its official software repositories and receive updates via DNF/YUM, you can do this by visiting official red hat website and [register there. I used my User name and password to do that .](https://sso.redhat.com/auth/realms/redhat-external/login-actions/registration?client_id=rhcom&tab_id=Or-E2j8bEoo&client_data=eyJydSI6Imh0dHBzOi8vd3d3LnJlZGhhdC5jb20vZW4vZGFzaGJvYXJkIiwicnQiOiJjb2RlIiwic3QiOiJjMDQ5MjBiYTRmNWI0YTcyODc5MDQ1MTY5NmNjNWM4ZiJ9)

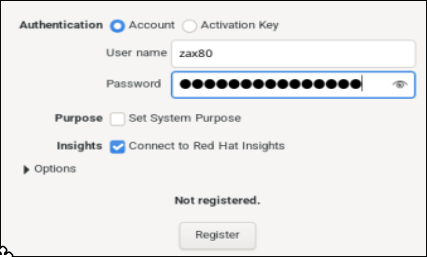


Figure 10: Register with Red hat subscription account.

## 2.5 Software selection

Minimal installation was chosen because we don’t need any fancy GUI to work with. A simple terminal will be more than enough.

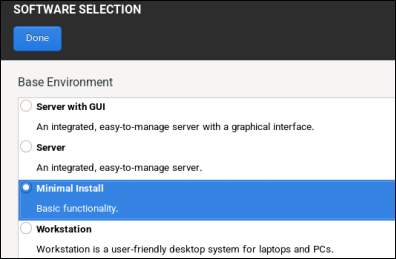


Figure 11: Choose minimal installation.

Our system started to work with it’s terminal as shown in figure 12.

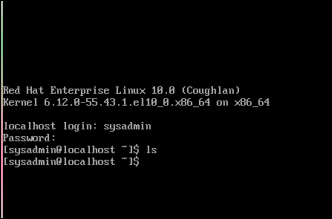


Figure 12: RHEL terminal

## 2.6 Automated Installation (Kickstart)

A **Kickstart configuration file** was used to automate the operating system installation process. This file was generated during the installation and contains predefined settings for system language, keyboard layout, disk partitioning, package selection, and timezone configuration.

Using Kickstart enables **consistent and repeatable deployments**, which is essential in enterprise environments where multiple systems may need to be installed with identical configurations.

**Figure 13 shows a portion of the generated Kickstart file, illustrating key configuration parameters used during the automated installation.**

The Kickstart configuration file was generated automatically during the installation process and saved as (**/root/anaconda-ks.cfg)**. This file contains all installation parameters required to reproduce the system setup, including language settings, disk partitioning, package selection, and timezone configuration. The code could be found in the [Appendix C](#_Appendix_C) and folder structure under **configuration-and-script\_files/** .

# Initial System Setup

## 3.1 System Update

After completing the operating system installation, the system was fully updated to ensure that all installed packages were running the latest available versions. This step is important for maintaining system security, stability, and performance.

The following command was used to update all packages: sudo dnf update –y

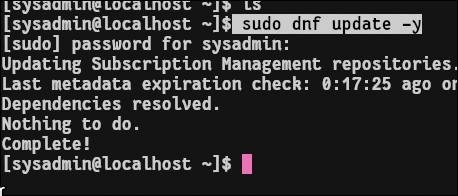


Figure 14: Update all packages in the system

## 3.2 Required Software Installation

After updating the system, the required software packages were installed to support storage management, performance tuning, file synchronization, networking, and web services.

The following command was used to install the required packages: sudo dnf install nano vdo kmod-kvdo rsync tuned httpd nginx net-tools shown in figure 15.

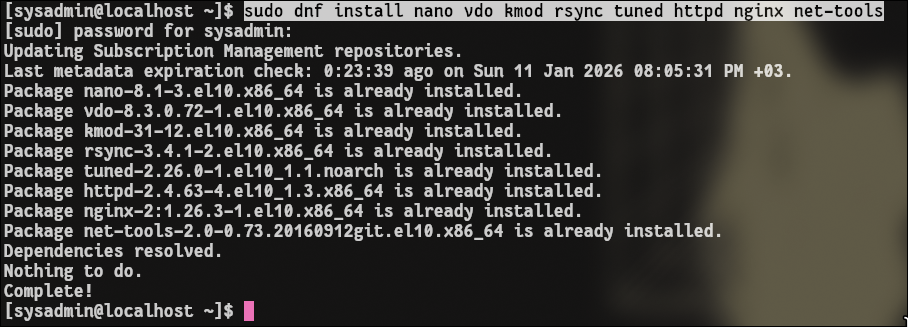


Figure 15: Install the required software.

# Storage Configuration (40GB Disk)

The additional **40 GB disk** was configured to provide dedicated storage for user data and departmental directories. This disk was initialized using a **GPT partition table** and configured with **Logical Volume Management (LVM)** to allow flexible and scalable storage management.

A single LVM physical volume was created on the disk and added to a volume group. Multiple logical volumes were then created on top of this volume group to separate data according to system requirements. All logical volumes were formatted using the **XFS filesystem**, which is the default and recommended filesystem for RHEL systems.

The existing /home directory was migrated to a new logical volume with a size of **15 GB**, and the original directory was preserved by renaming it to /**home.bak** to prevent data loss. In addition, a **10 GB logical volume** was created and mounted at /company to store departmental data. An additional **swap logical volume** was also created based on the system’s RAM size to improve memory management.

All filesystems and swap configurations were made persistent by adding the appropriate entries to the **/etc/fstab** file, ensuring that they are automatically mounted during system startup.

**Steps**:

1) Partition (GPT / MBR)

2) LVM

3) File System

4) Mount

5) Files

## 4.1 Disk Partitioning and Layout

The lsblk command was used to display the system’s block devices and verify the disk layout after storage configuration, as shown in Figure 16.

As shown in the output, the device **vdb**, which corresponds to the additional **40 GB storage disk**, was initially unpartitioned and did not contain any existing data or filesystems prior to configuration.

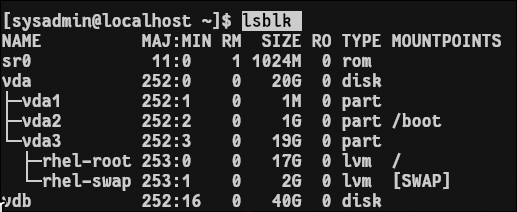


Figure 16: Block device layout showing system disks and logical volume mappings.

Linux provides multiple disk partitioning editors, including fdisk, parted, cfdisk, and sgdisk. For this setup, **fdisk** was selected due to its support for GPT and MBR partition tables and its suitability for partitioning a 40 GB disk, as shown in figure 17

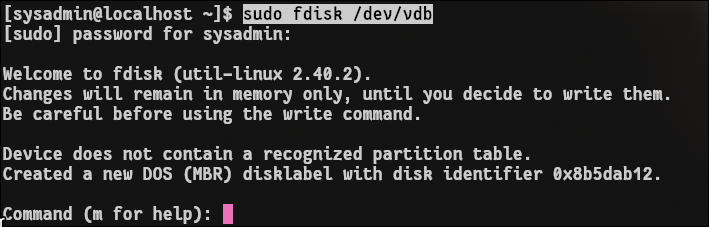


figure 17: Managing **/dev/vdb** disk using **fdisk** utility.

A new **GPT partition table** was created on the disk to support modern partitioning standards, as illustrated in Figure 18.



Figure 18: Making GPT partition table on our disk.

Following this, a single partition was created using the default values provided by fdisk, allowing the partition to take the entire disk capacity, as shown in Figure 19.

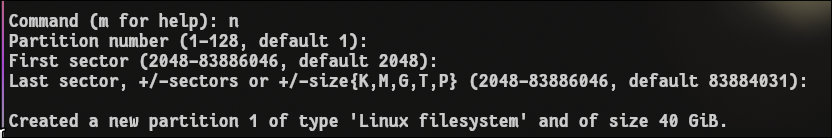


Figure 19: Making new partition on the hard disk.

The partition was made successfully, as showin in figure 20

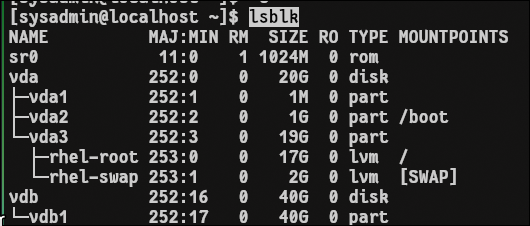


Figure 20: New partition was made name **vdb1**

## 4.2 LVM Configuration

After partitioning the storage disk, the newly created partition was initialized as an LVM physical volume using the **pvcreate** command, as shown in Figure 21.



Figure 21: Create physical volume for /dev/vdb1 to make physical extents (chunks of bytes)

Following this, a volume group named **vg0** was created using the physical volume /dev/vdb1, as illustrated in Figure 22. This volume group was used as the base for creating all required logical volumes.

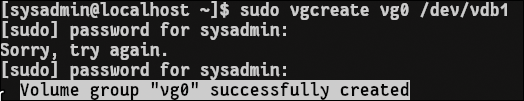
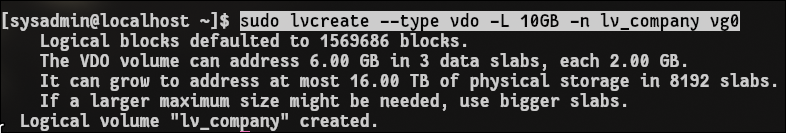
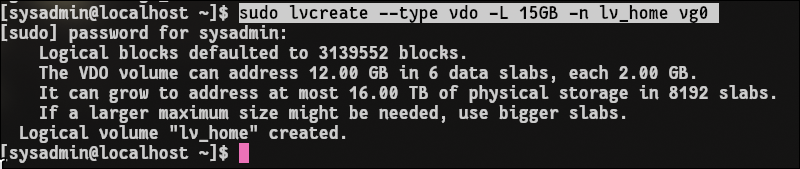


Figure 22: Create volume group consists of /dev/vdb1

A VDO-enabled logical volume named lv\_company with a size of 10 GB was created within the vg0 volume group using LVM-integrated VDO, as shown in Figure 23. This logical volume was configured to provide optimized storage for shared departmental data.

Figure 23: Create logical volume of type vdo for lv\_company.

The same has been done for **lv\_home,** as shown in figure 24

Figure 24: Create logical volume of type vdo for lv\_home

A swap logical volume was created without using VDO, as swap space requires direct and efficient access to storage.



Figure 25: create logical volume for swap space.

## 4.3 Filesystem Creation

Following creating logical volumes, a file system was created for each LV.

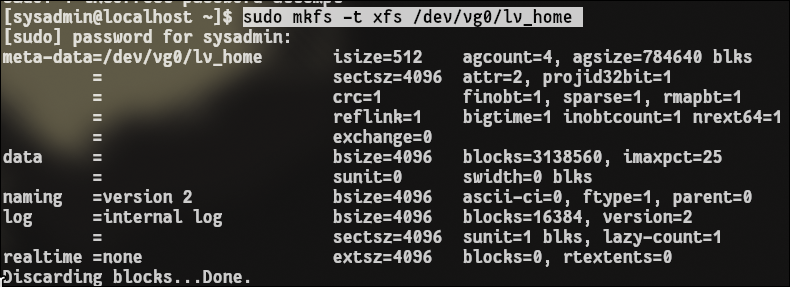
A file system was created for lv\_home and lv\_company.  

Figure 26: Creating file system for compnay and home logical volume.

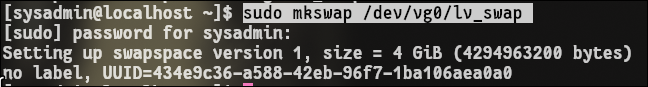
Lv\_swap was initialized as swap area using mkswap command, shown in figure 27. 

Figure 27: make swap area for lv\_swap.

Activate the swap area using swapon command.



Figure 28: Activate the swap area.

4.4 Mount Points and Directory Migration

After creating and formatting the logical volumes, mount points were prepared to make the filesystems accessible to the system. The /company directory was created and used as the mount point for the lv\_company logical volume, as shown in Figure 29.



Figure 29: mounting lv\_company into /company/

To migrate the existing home directories, the original /home directory was preserved by renaming it to /home.bak. A new /home directory was then created and used as the mount point for the lv\_home logical volume, as shown in Figure 30.



Figure 30: mounting lv\_home into /home/

4.5 Persistence Configuration

To ensure that the /home and /company filesystems are mounted automatically at system startup, their corresponding UUIDs were retrieved using the ***blkid*** command and added to the /etc/fstab file, as shown in figure 31.

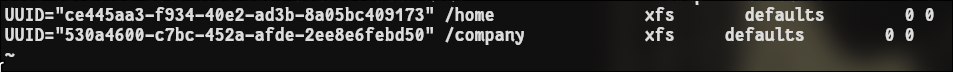


Figure 31: Persistent mount configuration for /home and /company

# 5. Departmental Directory Structure

A departmental directory structure was created to organize shared data in a clear and manageable manner. This structure is located under the `/company` directory, which is hosted on a dedicated logical volume to separate departmental data from user home directories.

Subdirectories were created for each department to ensure logical separation of files and improve data organization. This approach simplifies access control, backup management, and future expansion.

## 5.1 Department Directories

Department-specific directories were created under the /company directory to organize shared data according to organizational structure. Separate directories were established for each department, including **engineering**, **finance**, **hr**, and **management**, to ensure clear separation of departmental resources.

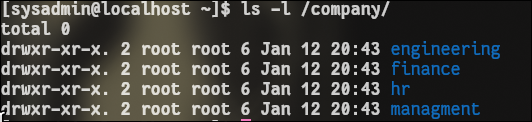


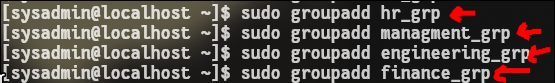
Figure 32: Deperatment directories in /company/ak

## 5.2 Access Control and Permissions

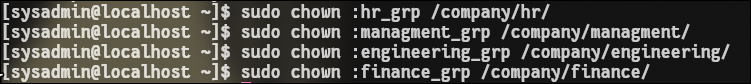
Access control was implemented using **Linux group permissions** to ensure departmental isolation and secure collaboration. Each departmental directory was assigned a dedicated group and configured to allow full access only to members of that group.

The ***setgid*** permission was applied to all departmental directories to ensure that newly created files inherit the directory’s group ownership, enabling seamless group collaboration. Additionally, the ***sticky bit*** was enabled to prevent users from deleting or modifying files created by other group members.

Create groups for each department.

Figure 33: making groups for each departmental directory.

Each departmental directory was assigned to a specific group. This ensures **department isolation.**

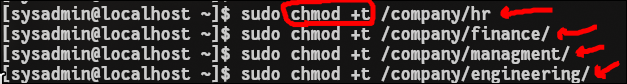
 Figure 34: changing the group for each directory

Directory permissions were configured to enforce **departmental isolation** while allowing **collaboration among users within the same department**. Each departmental directory was assigned group-based access permissions to **restrict access to authorized users only**.

Figure 35: Configuring group-based permissions for departmental directories

What 2770 means:

* 2 → **setgid** (inherit group ownership)
* 7 → owner: read/write/execute
* 7 → group: read/write/execute
* 0 → others: no access (isolation)

To prevent users from deleting or modifying files owned by other users within the same department, the sticky bit was enabled on all departmental directories. Figure 36: Enabling the sticky bit on departmental directories to prevent unauthorized file deletion

Final departmental directory structure and permission.

Figure 37: Verify /company directory structure and permission.

# 6. User Groups and Accounts

User groups were created to represent organizational departments and system roles, including hr, finance, engineering, management, and it. User accounts were then created and associated with the appropriate groups based on their responsibilities.

The **IT group** was designated for system administration purposes, and users belonging to this group were granted administrative privileges through sudo access. Non-IT users were restricted to standard user permissions in accordance with the access control policies defined earlier.

An initial password was assigned to user accounts for configuration and testing purposes.

To simplify and standardize user account creation, a Bash script was developed to automate the creation of users, assignment to departmental groups, configuration of administrative privileges, and initialization of passwords. The script ensures consistency with the user management requirements defined in Table 1 and reduces the risk of manual configuration errors.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Username** | **Department** | **Primary Group** | **Administrative Privileges** | **Password** |
| sara | HR | hr | No | Htu@123 |
| huda | HR | hr | No | Htu@123 |
| ahmed | Finance | finance | No | Htu@123 |
| rami | Finance | finance | No | Htu@123 |
| omar | Engineering | engineering | No | Htu@123 |
| ali | Engineering | engineering | No | Htu@123 |
| manager | Management | management | No | Htu@123 |
| admin1 | IT | it | Yes | Htu@123 |
| admin2 | IT | it | Yes | Htu@123 |

**Table1: User, Group, and Privilege Mapping**

The script creates user accounts according to the specifications shown in Table 1, including username, department, primary group, and administrative privileges.

The complete implementation of the automation script is provided in [Appendix A](#_Appendix_A) and is also available in the scenario document files.

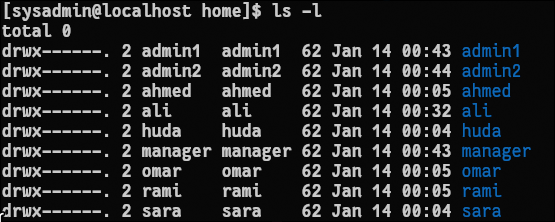


Figure 39: Users

The /etc/group file shows the defined system groups along with the users assigned to each group.

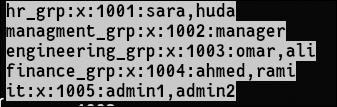


Figure 40: Contents of the /etc/group file showing group memberships.

# 7. Backup Mechanism Implementation

To ensure data availability and protect against accidental data loss, an automated backup mechanism was implemented for the system. The backup process was designed to run without manual intervention and to securely archive organizational data on a daily basis.

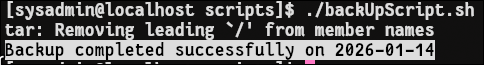
## 7.1 Backup Script

A Bash script was developed to perform **daily backups** of the /company directory, which contains all departmental data. The script creates a compressed archive of the directory using the tar utility and stores the resulting backup file in the /backup directory.  
 To prevent overwriting previous backups, each archive is named using the current date, allowing multiple backup versions to be retained.

This approach ensures that departmental data can be recovered in the event of system failure, accidental deletion, or data corruption.

The complete implementation of the backup script is provided in [**Appendix B**](#_Appendix_B)

**Using the script to backup /company into /backup directory.**

 Figure 41: Backup /comapny into /backup

## 7.2 Automation

To automate the backup process, the script was scheduled using the cron service. A cron job was configured to execute the backup script daily at **11:59 AM**, ensuring consistent and reliable backups without requiring user interaction.

This scheduling guarantees that backups are performed regularly and consistently, meeting the system’s data protection requirements.

The configured cron job was verified using the crontab -l command, confirming that the backup task is correctly scheduled.



Figure 42: crontab verification.

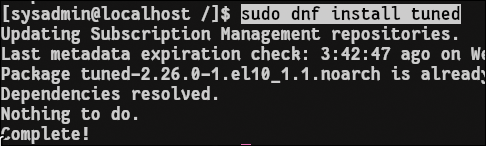
# 8. Server Optimization with TuneD

To ensure stable and responsive system performance in a virtualized environment, the TuneD performance tuning framework was deployed on the server. TuneD provides dynamic system tuning by applying predefined profiles optimized for different workloads.

## 8.1 Installation and Enablement of TuneD

The TuneD package was installed using the system package manager and configured to start automatically at system boot. Enabling TuneD ensures that performance optimizations are consistently applied without requiring manual intervention after reboots.

Tuning was installed using dnf package manager in Red Hat.

 Figure 43: installation of tuning

Tuned was enabled throw systemctl

 Figure 44: Enable tuning

## 8.2 Performance Profile Configuration

After enabling the TuneD service, the **virtual-guest** profile was applied to optimize the system for operation within a virtual machine. This profile adjusts CPU scheduling, memory management, and I/O behavior to improve performance and stability in virtualized environments.

The active TuneD profile was verified to confirm that the correct optimization settings were applied.

The **virtual-guest profile** was activated to optimize kernel and system parameters for improved performance and stability in a virtualized environment, rather than prioritizing power saving.

Figure 45: Activating the virtual-guest TuneD performance profile.



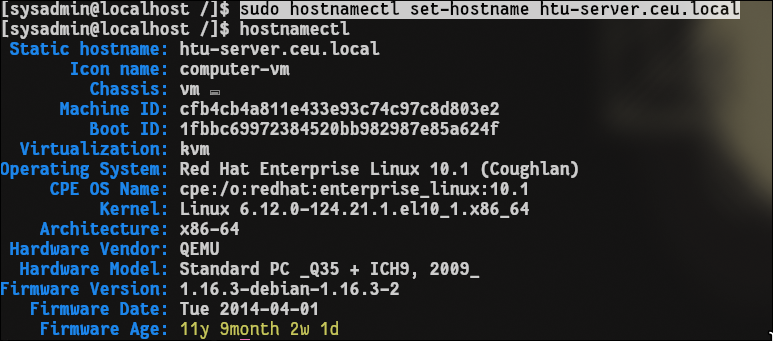
Figure 46: confirming active profile.

# 9. System Identity & Package Management

Proper system identity and package management are essential for reliable server operation and long-term maintainability. In this section, the server’s hostname and local name resolution were configured to ensure consistent identification and independence from external DNS services. In addition, stable package repository configurations were verified to guarantee continued access to software packages and updates.

## 9.1 Hostname Configuration

The system hostname was configured as **htu-server.ceu.local** to provide a clear and consistent identity for the server within the HTU environment. Using a fully qualified domain name improves system manageability, simplifies administration, and ensures compatibility with services that rely on hostname-based identification.

 Figure 47: Verifying the system hostname configuration using hostnamectl.

## 9.2 Local Hostname Resolution

Local hostname resolution was configured to ensure that the server can resolve its own hostname without relying on external DNS services. This was achieved by updating the **/etc/hosts file** to map the server’s hostname to its local IP address.

By configuring local name resolution, the system is able to reliably resolve htu-server.ceu.local during startup and normal operation, even when network connectivity or DNS services are unavailable. This prevents service startup delays and improves overall system stability.

An entry was added to the **/etc/hosts file** to map the server’s hostname to its local IP address:

 Figure 48: adding entry to /etc/hosts for IP resolution

To verify that local hostname resolution is functioning correctly, the ping command was used to test connectivity to the configured hostname.

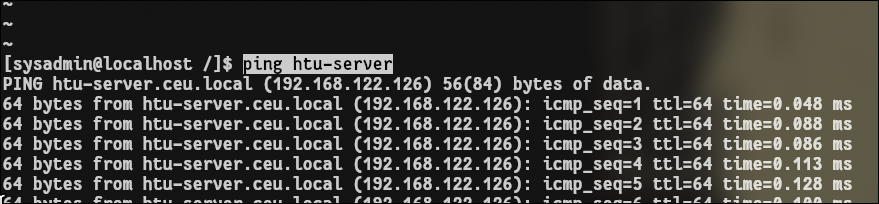


Figure 49: Verify connection to local hostname.

# 10. YUM Repository Configuration

Reliable package management is essential for system maintenance and software installation. To ensure consistent access to system packages and updates, YUM repositories were configured to use internally provided repository sources rather than external public repositories.

## 10.1 BaseOS Repository

The BaseOS repository was configured to provide access to core operating system packages, including essential system components and libraries required for stable server operation. The repository was defined using the provided internal URL and enabled to allow package installations and updates through the system package manager.

The BaseOS repository configuration was defined in the /etc/yum.repos.d/htu.repo file, linking the repository to the internal BaseOS URL at <http://content.example.com/rhel8.0/x86_64/dvd/BaseOS>.

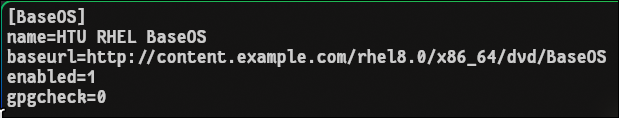


Figure 50: BaseOS repository configuration defined in /etc/yum.repos.d/htu.repo.

## 10.2 AppStream Repository

The AppStream repository was configured to provide access to application-level packages and additional software components required by the server. This includes services, development tools, and runtime environments that extend the functionality of the base operating system.

The AppStream repository configuration was defined in the /etc/yum.repos.d/htu.repo file, linking the repository to the internal AppStream URL at  
 <http://content.example.com/rhel8.0/x86_64/dvd/AppStream>.  
 The repository was enabled to ensure that application packages can be installed and managed using the system package manager.



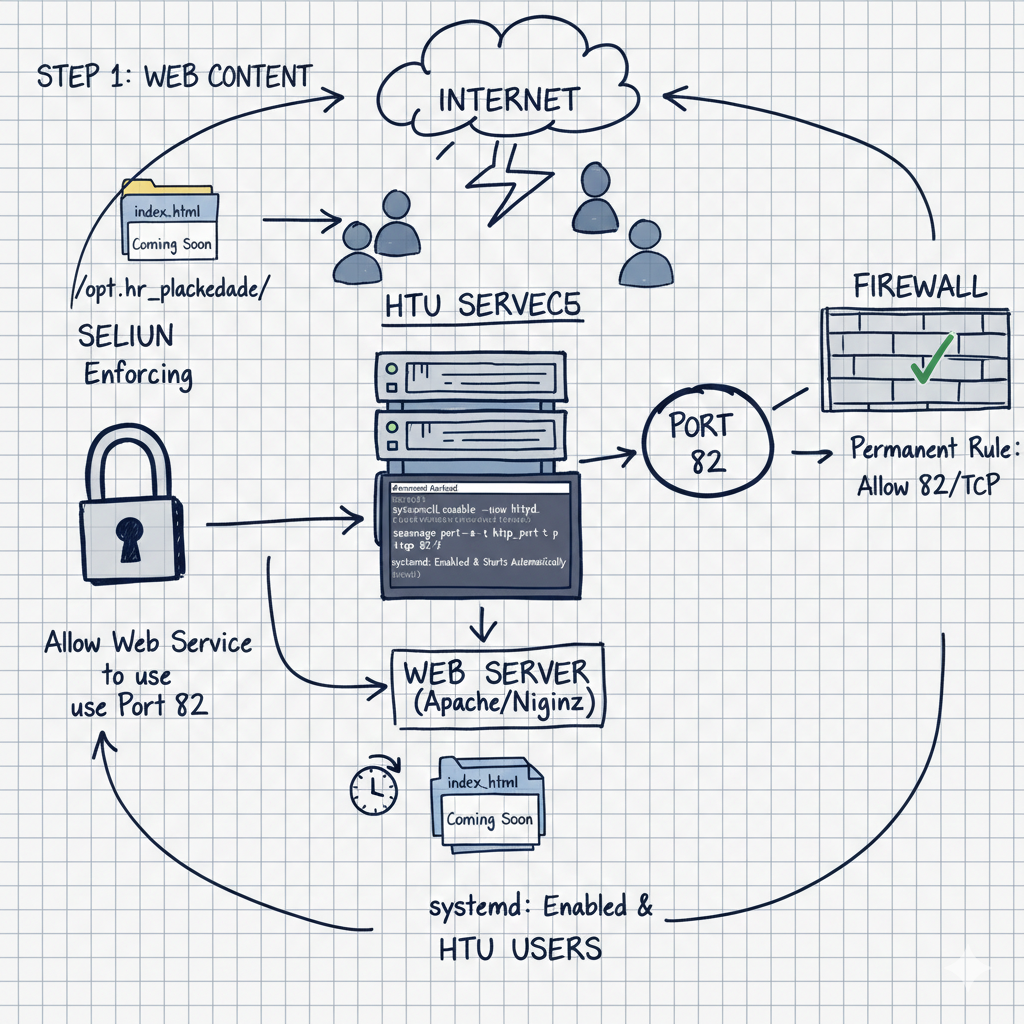
Figure 51: AppStream repository configuration defined in /etc/yum.repos.d/htu.repo.

When executing **dnf install commands**, the system queries the enabled BaseOS and AppStream repositories defined in the repository configuration files and installs packages if they are available.

This approach is particularly useful for local or isolated servers that are not connected to the internet, as it allows software packages to be installed from internally provided repository sources instead of external Red Hat repositories, and offcourse they suppose to support .rpm (redhat files) and cannot support any OS files like .deb (debian).

# 11. Temporary HR Web Application Placeholder

To ensure the availability of an interim HR service while the final application is under development, a temporary web placeholder was deployed on the server. This placeholder provides a simple “Coming Soon” page and demonstrates the server’s ability to host and manage web services securely. The deployment includes proper service management using systemd, secure access control enforced by SELinux, and controlled network access through firewall configuration.



High-level structure of this section

## 11.1 Web Content

A custom directory was created at **/opt/hr\_placeholder** to host the HR placeholder content. This location was chosen to clearly separate custom application data from system-managed directories. A simple HTML page displaying a “Coming Soon” message was added to verify correct content delivery by the web service.

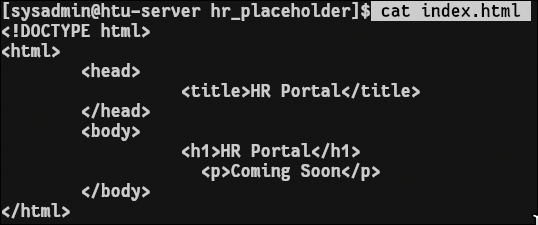


Figure 52: simple html page.

## 11.2 Web Service Configuration

The NGINX web server was selected to host the temporary HR placeholder due to its lightweight design and reliability. NGINX was configured to serve content from /opt/hr\_placeholder and operate on a non-standard port (82) to avoid conflicts with default services. The service was managed using systemd to ensure that it runs continuously in the background and starts automatically after every system reboot.

A custom NGINX configuration file named hr\_placeholder.conf was created in **/etc/nginx/conf.d/**. The following configuration was added to allow NGINX to listen for HTTP requests on port 82 and serve content from the HR placeholder directory, shown in figure 53.

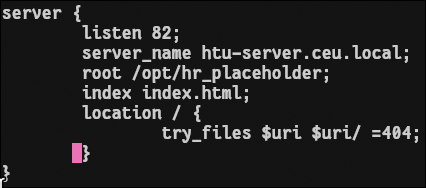


Figure 53: NGINX server configuration for the HR placeholder service listening on port 82.

The NGINX service was enabled and managed using systemd to ensure that it runs continuously in the background and starts automatically at system boot.

The service status was verified using the **systemctl status nginx** command, confirming that the NGINX service is active and running.

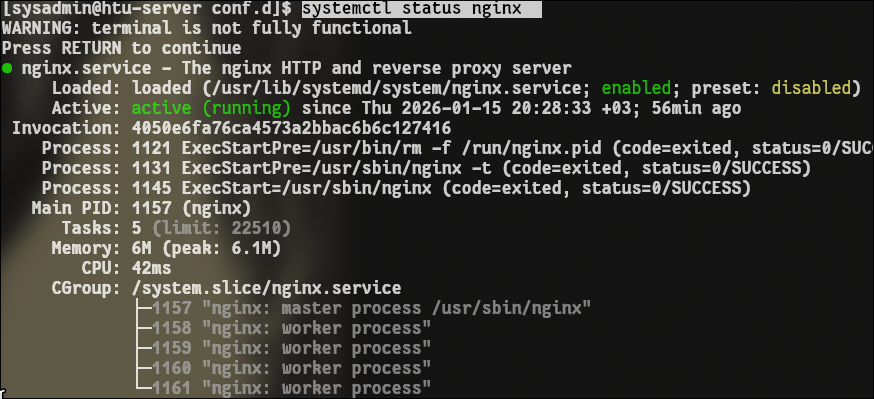


Figure 54: NGINX service status verified using systemd.

## 11.3 SELinux

SELinux was kept in **enforcing mode** to maintain mandatory access control and ensure that security policies remain active during service operation. Rather than disabling SELinux, the required permissions were configured to allow the NGINX web server to operate securely within the enforced policy.

The correct SELinux file context was applied to the custom web directory (/opt/hr\_placeholder) to allow NGINX to read and serve the placeholder content. Additionally, SELinux was configured to permit the web service to bind to the non-standard port (82), enabling HTTP access without compromising system security.

These configurations allow the web service to function correctly while preserving the system’s security posture.

SELinux was verified to be operating in **enforcing mode**, ensuring that mandatory access control policies remain active on the system.



Figure 55: Verifying SELinux enforcing mode using the getenforce command.

This command configures SELinux to allow web servers (like Apache or Nginx) to use TCP port 82.



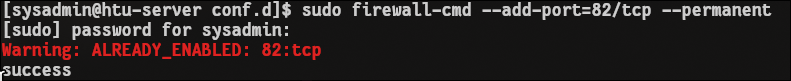
Figure 56: Allowing Nginx web server to listen to port 82.

## 11.4 Firewall Configuration

The firewall controls who is allowed to reach the server over the network.

No one can reach the service unless the firewall allows the port.

Allowing traffic to port 82 using firewall.

 Figure 56: Open traffic on port 82.

# 12. SSH Service Hardening

To enhance the security of remote access to the server, the SSH service was hardened in accordance with HTU’s security requirements. The applied hardening measures focus on eliminating password-based authentication, enforcing the use of secure cryptographic keys, and restricting SSH access to authorized administrative users only. These configurations significantly reduce the risk of unauthorized access and brute-force attacks.

## 12.1 SSH Key Authentication

SSH key-based authentication was implemented for the sysadmin account to replace traditional password-based logins. A secure SSH key pair was generated, and the public key was installed on the server to allow authenticated access.

Key-based authentication ensures that only users in possession of the corresponding private key can log in, providing a stronger and more secure authentication mechanism.

A public/private SSH key pair was generated using the ssh-keygen utility, as shown in Figure 57.

 Figure 57: Generating an SSH key pair on the client system.

The generated public key was then added to the authorized\_keys file located in **/home/sysadmin/.ssh/**, allowing the sysadmin user to authenticate using 12.2 Disable Password Login for sysadmin

SSH keys.

 Figure 58: nstalling the public SSH key in the authorized\_keys file for the sysadmin user.

## 12.2 Disable Password Login for sysadmin

Password authentication was explicitly disabled for the sysadmin account, enforcing exclusive use of SSH key-based authentication. This prevents administrative access through passwords, even if credentials are compromised, and ensures that only authorized key holders can log in as the system administrator.

The SSH daemon configuration file (sshd\_config) was modified by adding this code to disable password-based authentication for the sysadmin account, enforcing exclusive use of SSH key-based authentication.



Figure 59: Disabling password-based SSH authentication for the sysadmin user in sshd\_config.

## 12.3 Disable Password Authentication System-Wide

Password-based SSH authentication was disabled for all system users. This system-wide restriction eliminates the risk of password guessing and brute-force attacks against user accounts, ensuring that all remote SSH access relies solely on secure key-based authentication.

The SSH daemon configuration file (/etc/ssh/sshd\_config) was modified to disable password-based authentication for all users. By setting PasswordAuthentication no, the system enforces exclusive use of SSH key-based authentication for all remote SSH access, effectively preventing password guessing and brute-force attacks.



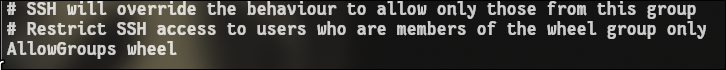
Figure 60: Disabling password-based SSH authentication system-wide in sshd\_config.

## 12.4 Restrict SSH Access to Administrative Users

SSH access was restricted to authorized administrative users only. Non-administrative accounts were denied remote SSH access by limiting login permissions to the administrative group. This approach reduces the system’s attack surface and ensures that remote access is used strictly for administrative purposes.

The configuration file (**/etc/ssh/sshd\_config)** was edited to apply only the **wheel group** (have the authorized administrative) into the allowed groups.

This configuration will override the default configuration/ behavior ssh.

 Figure 61: Restricting SSH access to administrative users using the AllowGroups directive.

The service was restarted to apply the new configuration.

 Figure 62: Restarting the sshd service.

# Appendix A

User Account Automation Script

#!/bin/bash

#The purpose of this script is to automate the creation of a user on the system and assign it to a group

echo -e "\nPlease note that you can't enter names having the following\n

1) Starting with a number, Ex: 12Ahmad, 12\n2) Starting with uppercase letters, Ex: Ahmed, USER1

3) Special characters, EX: @ ! $ % ... "

# check if it the correct foramt

while true; do

read -p "Enter username:" username

if [[ ! "$username" =~ ^[a-z][a-z0-9\_]\*$ ]]; then

echo "Error: only lowercase letters, numbers and underscores are allowed"

echo "Please re-enter a name with the correct format"

else

break

fi

done

put\_password() {

while true; do

sudo passwd "$username"

if [ "$?" == 0 ]; then

break

fi

echo "Password doesn't match"

done

}

# Make the username and password

sudo adduser "$username"

if [ "$?" == 0 ]; then

put\_password

else

echo "Passed"

fi

while true; do

read -p "Do you want to delete the user ? yes/no " take

case "$take" in

yes)

sudo userdel -r "$username"

echo "User deleted succefully"

exit 0

;;

no)

break

;;

\*)

echo "Enter yes or no please"

;;

esac

done

# Assign the user to a departmental group or adminstartive group

groups=("hr\_grp" "managment\_grp" "engineering\_grp" "finance\_grp" "it")

#Enter group function and Check if group num is correct

choose\_group() {

echo "Which group do you want to assign the user into ?"

echo "Choose a number 1-5 ."

echo -e "1) hr\_grp\n2) managment\_grp\n3) engineering\_grp\n4) finance\_grp\n5) it"

while true; do

read -p "Enter group: " grpNum

if [[ ! "$grpNum" =~ ^[1-5]$ ]]; then

echo "Invalid group number"

else

((grpNum -= 1))

sudo usermod -aG "${groups[$grpNum]}" "$username"

echo "$username added successfully to group ${groups[$grpNum]}"

break

fi

done

}

#start the function

choose\_group

#Add the username to the group

#Change group if choose one by mistake

while true; do

read -p "Do you want to remove the user from this group? yes/no " remove

case "$remove" in

yes)

sudo gpasswd -d "$username" "${groups[$grpNum]}"

echo "User deleted from the group succesfully"

choose\_group

;;

no)

echo -e "Done\n"

break

;;

\*)

echo "Enter yes or no please"

;;

esac

done

#If the group is it then add it to the wheel group for admin access

if [[ "$grpNum" == 4 ]]; then

sudo usermod -aG wheel "$username"

fi

echo -e "Username in /etc/passwd file:"

sudo cat /etc/passwd | grep "$username"

echo -e "\n"

echo -e "Username in /etc/group file:"

sudo cat /etc/group | grep "$username"

# Appendix B

#!/bin/bash

#Backup directory

SOURCE\_DIR="/company" BACKUP\_DIR="/backup"

#Create backup directory if it does not exist

sudo mkdir -p "$BACKUP\_DIR"

#Generate date for filename

DATE=$(date +%F)

#Backup file name

BACKUP\_FILE="$BACKUP\_DIR/company\_backup\_$DATE.tar.gz"

#Create compressed archive

sudo tar -czf "$BACKUP\_FILE" "$SOURCE\_DIR"

#log result

if [[ $? -eq 0 ]]; then

echo "Backup completed successfully on $DATE"

else echo "Backup failed on $DATE"

fi

# Appendix C

#Generated by Anaconda 40.22.3.26

#Generated by pykickstart v3.52.8

#version=RHEL10

#Use graphical install

Graphical

%addon com\_redhat\_kdump --enable --reserve-mb='auto'

%end

#Keyboard layouts

keyboard --vckeymap=us --xlayouts='us'

#System language

lang en\_US.UTF-8

%packages

@^minimal-environment

%end

#Run the Setup Agent on first boot

firstboot --enable

#Generated using Blivet version 3.10.0

ignoredisk --only-use=vda

autopart

#Partition clearing information

clearpart --none --initlabel

#System timezone

timezone Asia/Amman --utc

#Root password

rootpw --lock

# References

* Red Hat, Red Hat Enterprise Linux 8 Documentation, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8>
* Red Hat, Logical Volume Manager (LVM) Administration Guide, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_and_managing_logical_volumes/>
* Red Hat, SELinux User and Administrator Guide, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/using_selinux/>
* Red Hat, Managing Firewalls and Packet Filters, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/security_hardening/using_firewalls>
* OpenSSH Project, OpenSSH Manual Pages, OpenBSD Project.  
   <https://www.openssh.com/manual.html>
* Red Hat, Securing OpenSSH, Red Hat Inc.  
   <https://access.redhat.com/solutions/1246933>
* NGINX Inc., NGINX Documentation.  
   <https://nginx.org/en/docs/>
* Red Hat, Using TuneD to Optimize Performance, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/monitoring_and_managing_system_status_and_performance/using-tuned>
* Red Hat, Systemd Service Management, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/assembly_managing-services-with-systemd>
* Red Hat, DNF Package Manager Documentation, Red Hat Inc.  
   <https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/installing_and_managing_software_packages/>
* GNU Project, Cron and Scheduled Tasks Documentation.  
   <https://www.gnu.org/software/mcron/manual/>
* Linux Foundation, Filesystem Hierarchy Standard (FHS).  
   <https://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html>