

Red Hat Enterprise Linux Installation and Setup Lab

By Michael Ambeguia

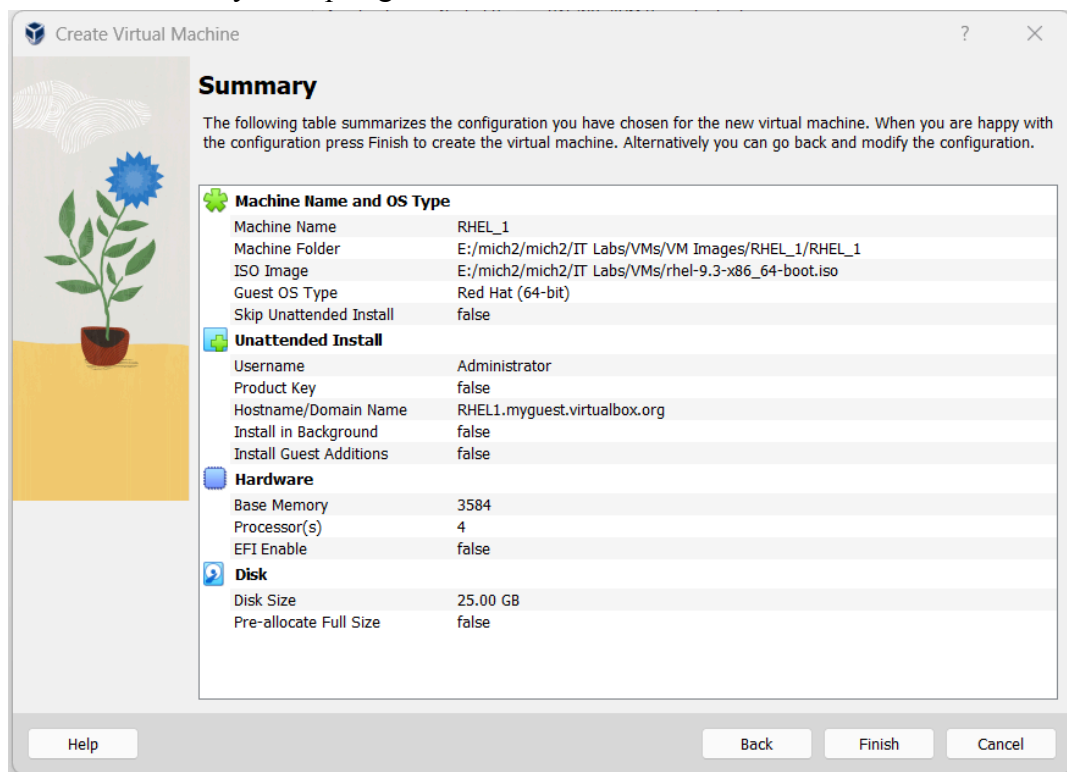
Lab Purpose: The purpose of this lab is to gain hands-on experience installing RHEL. I would also like to practice customizing my RHEL VM, conducting user management, system hardening, and various other post-installation tasks. RHEL is one of the most successful Linux distributions available. It is also one of the most used Linux OSs in an enterprise setting due to its comprehensive support, updates, and security and utility features.

Sections:

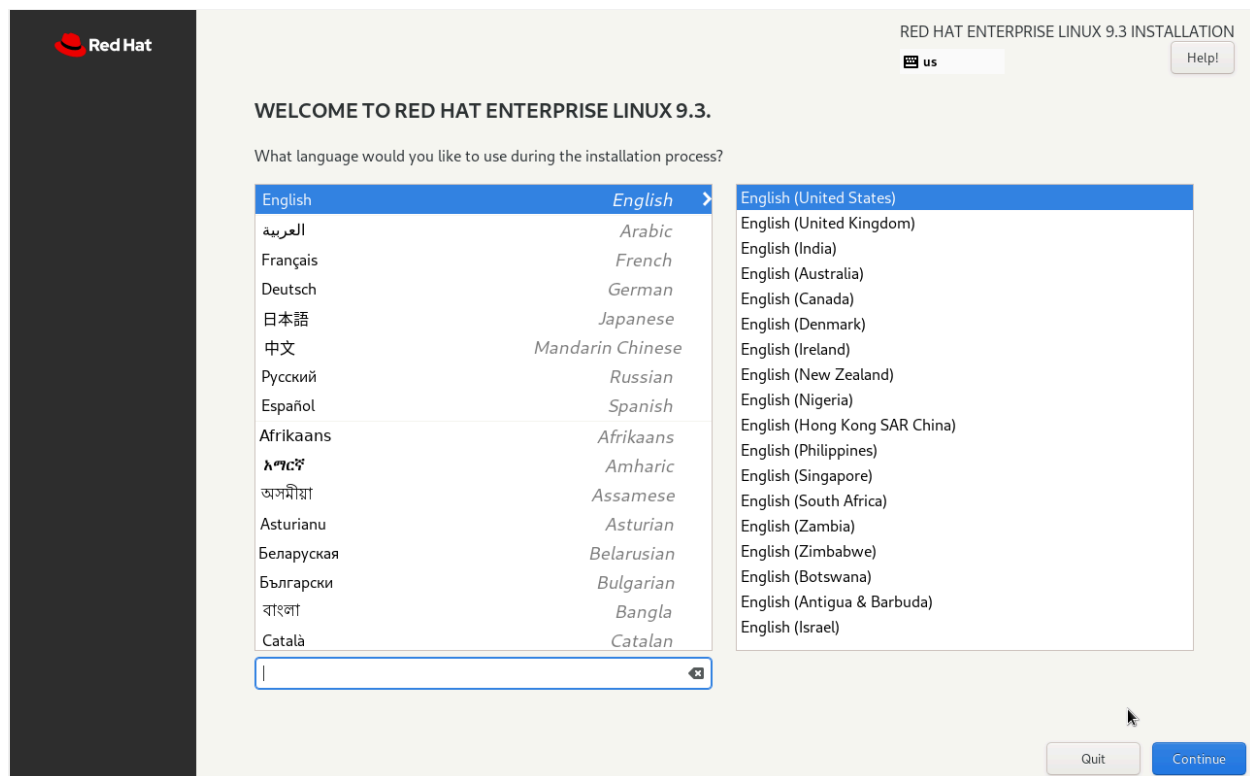
1. Installation
2. User Management
3. System Hardening
4. System customization
5. Installing key services and applications.

Section #1 Installation:

Step #1: Set up a VM in VirtualBox. Configure your VM settings such as storage size and the amount of memory and cpus given to the VM.




Step #2. Start the installation process once the VM is running. The following screenshots will depict what you will see in order.



Choose your language and country. You will then see the following screen. You will want to do the following tasks:

1. Set the root password.
2. Connect your VM to your Redhat Account. ***This is very important since without registering your vm you will not receive any updates and the rpm/ yum package managers will not work***
3. Next choose your system type and the software you want. I configured my VM to be a RHEL server with a GUI. I chose to skip the installation of software since I will do so during the post installation phase.




INSTALLATION SUMMARY


RED HAT ENTERPRISE LINUX 9.3 INSTALLATION


us

Help!


LOCALIZATION


 **Keyboard**
English (US)


 **Language Support**
English (United States)

 **Time & Date**
Americas/Los Angeles timezone


SOFTWARE


 **Connect to Red Hat**
Not registered.


 **Installation Source**
Red Hat CDN


 **Software Selection**
Red Hat CDN requires registration.

SYSTEM


 **Installation Destination**
Automatic partitioning selected


 **KDUMP**
Kdump is enabled

 **Network & Host Name**
Connected: enp0s3

 **Security Profile**
No profile selected

USER SETTINGS

 **Root Password**
Root password is set

 **User Creation**
No user will be created

ROOT PASSWORD


Done

RED HAT ENTERPRISE LINUX 9.3 INSTALLATION


us

Help!

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

Root Password: 

Strong

Confirm: 

☐ Lock root account

☐ Allow root SSH login with password

Set the root password.

CONNECT TO RED HAT

RED HAT ENTERPRISE LINUX 9.3 INSTALLATION

Done

us

Help!

Authentication

☒ Account ☐ Activation Key

User name

Password

Purpose

☒ Set System Purpose

Role

Red Hat Enterprise Linux Workstation

SLA

Self-Support

Usage

Development/Test

Insights

☐ Connect to Red Hat Insights

Options

Not registered.

Register

Connect the VM to your Red Hat Account. You will enter your username and password for the account and set the purpose of your VM. Or you can use an activation key like I did. The activation key can be created from your Red Hat account online.

CONNECT TO RED HAT

RED HAT ENTERPRISE LINUX 9.3 INSTALLATION

Done

us

Help!

The system is registered.

Unregister

Method

Registered with organization 17849446

System Purpose

Role: Red Hat Enterprise Linux Workstation

SLA: Self-Support

Usage: Development/Test

Insights

Not connected to Red Hat Insights

Subscribed in Simple Content Access mode.

Now my VM is connected to my Red Hat Account.

SOFTWARE SELECTION

Done

RED HAT ENTERPRISE LINUX 9.3 INSTALLATION

us

Help!

Base Environment

☒ **Server with GUI**
An integrated, easy-to-manage server with a graphical interface.

☐ **Server**
An integrated, easy-to-manage server.

☐ **Minimal Install**
Basic functionality.

☐ **Workstation**
Workstation is a user-friendly desktop system for laptops and PCs.

☐ **Virtualization Host**
Minimal virtualization host.

☐ **Custom Operating System**
Basic building block for a custom RHEL system.

Additional software for Selected Environment

☐ **DNS Name Server**
This package group allows you to run a DNS name server (BIND) on the system.

☐ **FTP Server**
These tools allow you to run an FTP server on the system.

☐ **Virtualization Tools**
Tools for offline virtual image management.

☐ **Virtualization Hypervisor**
Smallest possible virtualization host installation.

☐ **Basic Web Server**
These tools allow you to run a Web server on the system.

☐ **Virtualization Client**
Clients for installing and managing virtualization instances.

☐ **Debugging Tools**
Tools for debugging misbehaving applications and diagnosing performance problems.

☐ **Remote Desktop Clients**

☐ **Guest Agents**
Agents used when running under a hypervisor.

☐ **Network Servers**
These packages include network-based servers such as DHCP, Kerberos and NIS.

☐ **Network File System Client**
Enables the system to attach to network storage.

☐ **Mail Server**

Now I will choose my install type which is a server with a GUI.

Step 4. Install the OS.

Red Hat

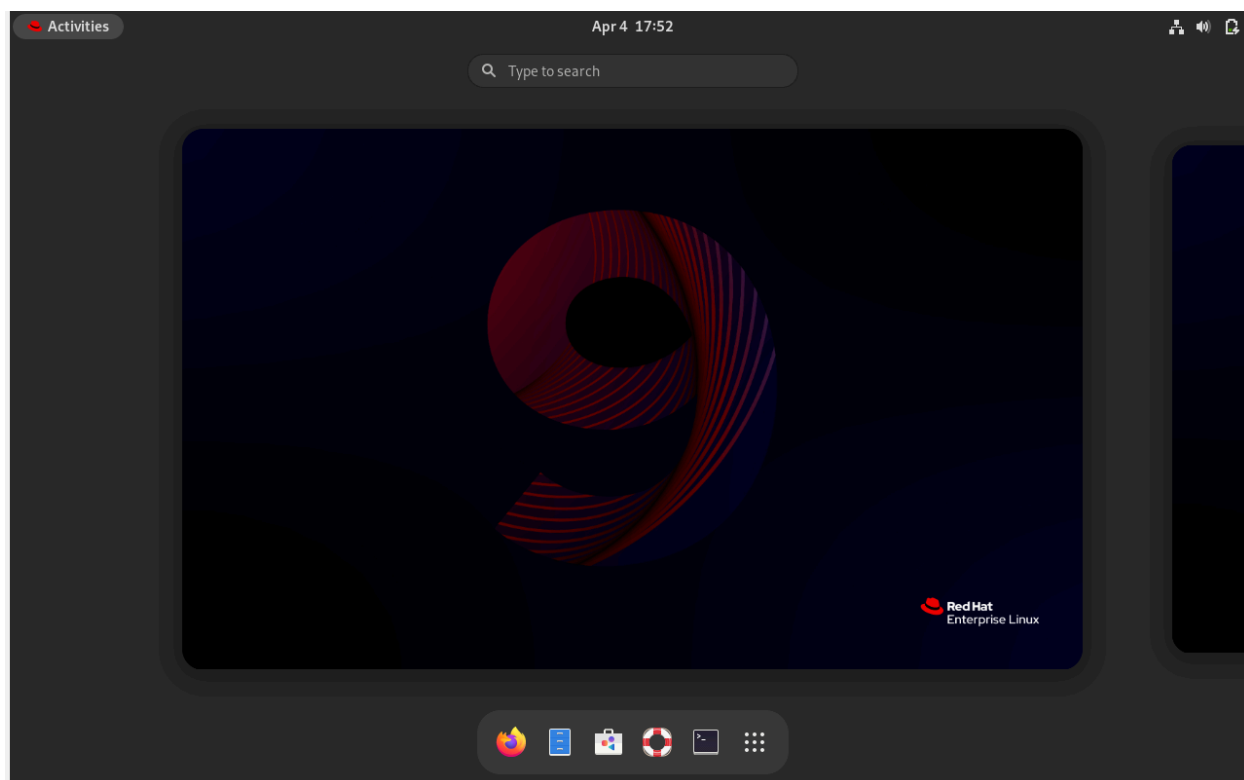
INSTALLATION PROGRESS

RED HAT ENTERPRISE LINUX 9.3 INSTALLATION

us

Creating xfs on /dev/sda1

Step #5. The installation is done. Sign into the new RHEL VM.



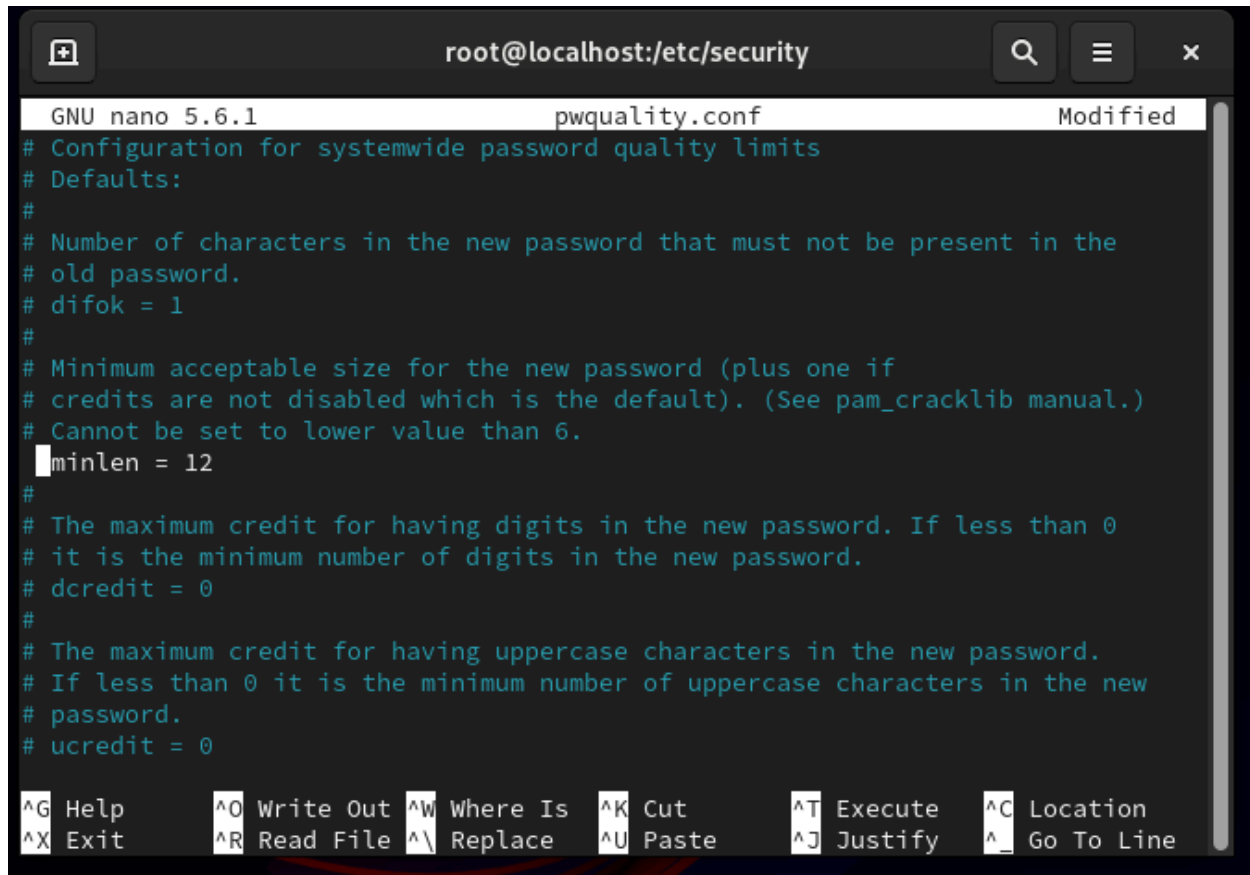
Section #2 User Management:

Step 1. Set the root password. Since this is a new system root will not have a password assigned, so one must be given for it so that privileged tasks can be accomplished on the RHEL server.

Even though you created the root password during installation, the password is simply for the first account you created and that user will have sudo privileges, but they are not root.

Step 2. Configure strong password policies for all subsequent users on the system after the root user is created. This can be done by configuring the libpam-pwquality service. This must be done before more users are created since the password policies will not be enforced on users that were created prior to setting them.

Configuring the libpam-pwquality configuration file. To make sure it is enforced get rid of the # to uncomment the setting parameter.



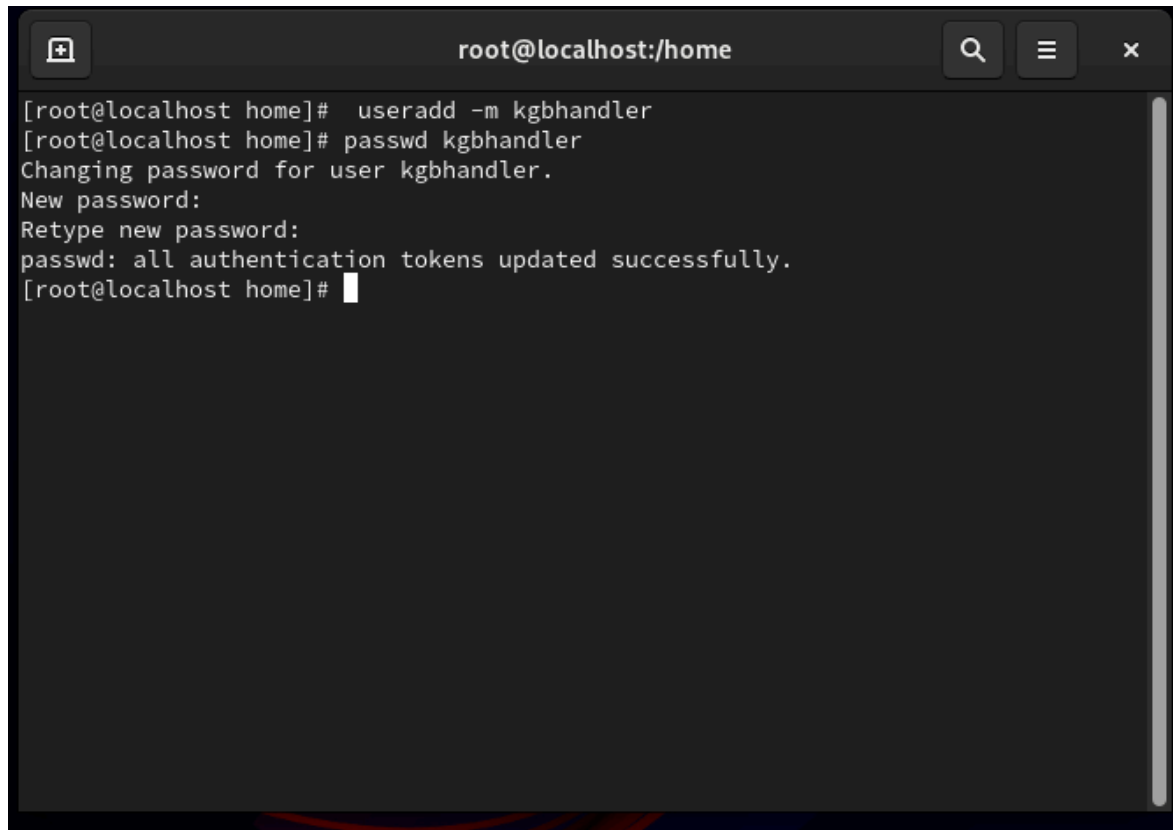
```
root@localhost:/etc/security
GNU nano 5.6.1 pwquality.conf Modified
# Configuration for systemwide password quality limits
# Defaults:
#
# Number of characters in the new password that must not be present in the
# old password.
# difok = 1
#
# Minimum acceptable size for the new password (plus one if
# credits are not disabled which is the default). (See pam_cracklib manual.)
# Cannot be set to lower value than 6.
minlen = 12
#
# The maximum credit for having digits in the new password. If less than 0
# it is the minimum number of digits in the new password.
# dcredit = 0
#
# The maximum credit for having uppercase characters in the new password.
# If less than 0 it is the minimum number of uppercase characters in the new
# password.
# ucrcedit = 0
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Configurations Made:

1. Minlen = 12 (so passwords must be 12 characters or longer)
2. Minclass = 4 (so upper, lowercase, numbers, and special required)
3. Dictcheck =1 (Compare passwords to ones found in password dictionary)
4. Usercheck =1 (Make sure that username is not used in password)
5. Retry = 5 (Five retries allowed for password authentication)
6. Enforce_for_root (When root password is changed, the settings apply)

Step 3. I will create a sudo user named KGBHandler and add the user to the wheel (sudo) group.

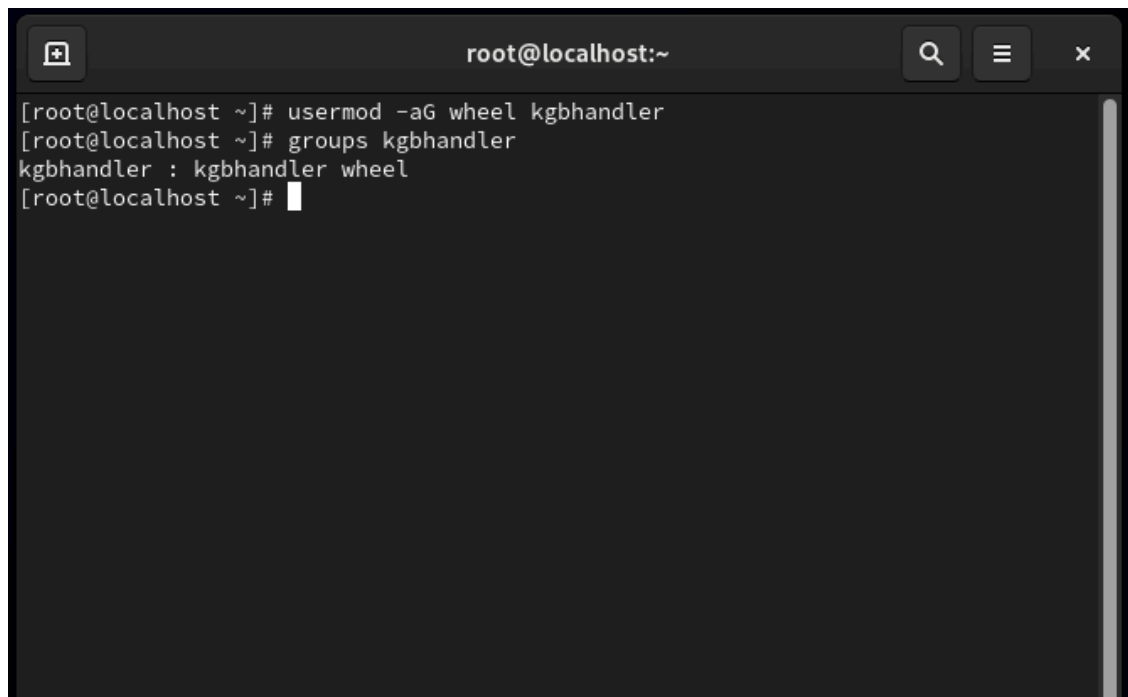
Create KGBHandler user using `useradd -m` and set their password using `passwd`



A terminal window titled "root@localhost:/home" with search, menu, and close buttons. The terminal shows the following commands and output:

```
[root@localhost home]# useradd -m kgbhandler
[root@localhost home]# passwd kgbhandler
Changing password for user kgbhandler.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
[root@localhost home]#
```

Add KGBHandler to wheel group (sudo group) using usermod -aG wheel

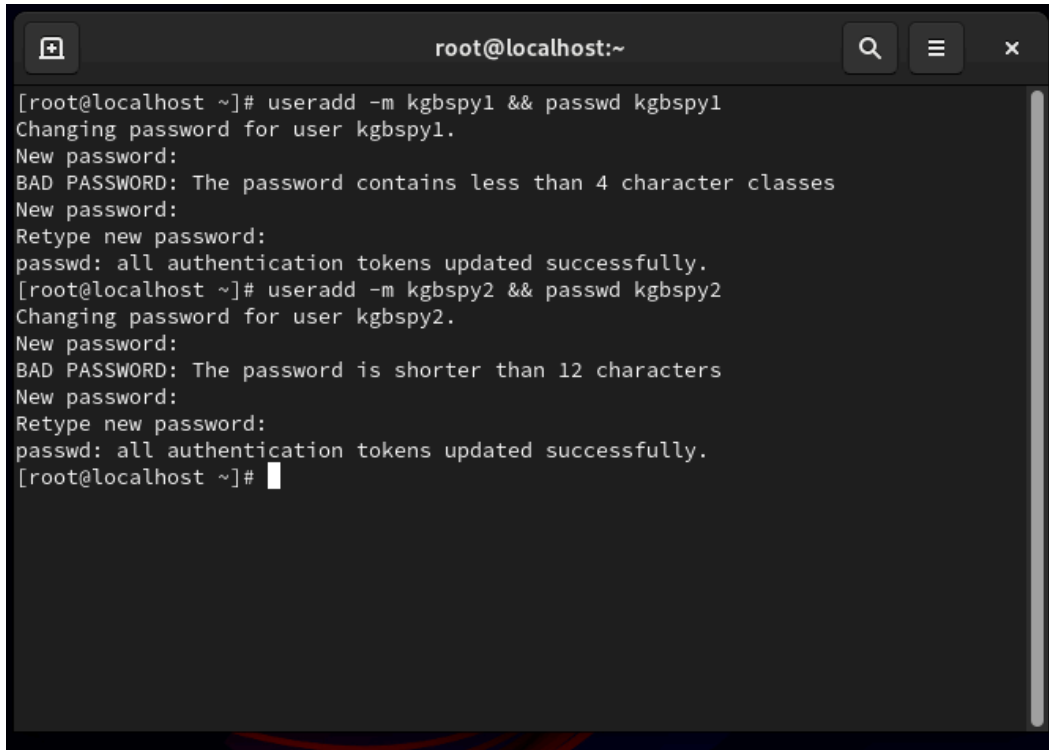


A terminal window titled "root@localhost:~" with search, menu, and close buttons. The terminal shows the following commands and output:

```
[root@localhost ~]# usermod -aG wheel kgbhandler
[root@localhost ~]# groups kgbhandler
kgbhandler : kgbhandler wheel
[root@localhost ~]#
```


Step 4. I will create two non root or sudo users named KGBSpy1 and KGBSpy2. They will have non administrative permissions.

Create the two users using `useradd -m` and `passwd`

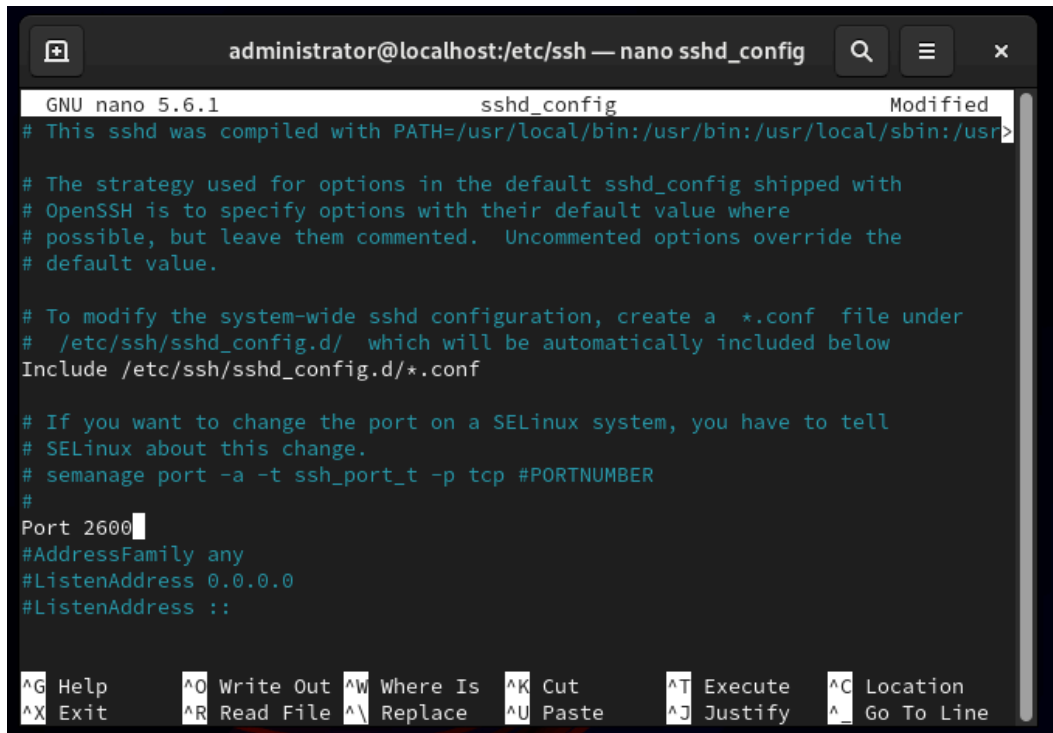


```
root@localhost:~  
[root@localhost ~]# useradd -m kgbspy1 && passwd kgbspy1  
Changing password for user kgbspy1.  
New password:  
BAD PASSWORD: The password contains less than 4 character classes  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@localhost ~]# useradd -m kgbspy2 && passwd kgbspy2  
Changing password for user kgbspy2.  
New password:  
BAD PASSWORD: The password is shorter than 12 characters  
New password:  
Retype new password:  
passwd: all authentication tokens updated successfully.  
[root@localhost ~]#
```

Section #3 System Hardening:

Note: You want to change the setting on the `/etc/ssh/sshd_config` file since you want to enforce settings for the server SSH software, not the client SSH software.

1. Change the ssh port by editing the `/etc/ssh/sshd_config` file. You can choose any port that is not being used, I will choose 2600 since it is random. You simply uncomment Port and then change it from 22 to whatever port number you want to use.



The image shows a terminal window with the nano text editor open to the file `/etc/ssh/sshd_config`. The window title is `administrator@localhost:/etc/ssh — nano sshd_config`. The editor shows the following content:

```
GNU nano 5.6.1 sshd_config Modified
# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/usr/local/sbin:/usr>

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options override the
# default value.

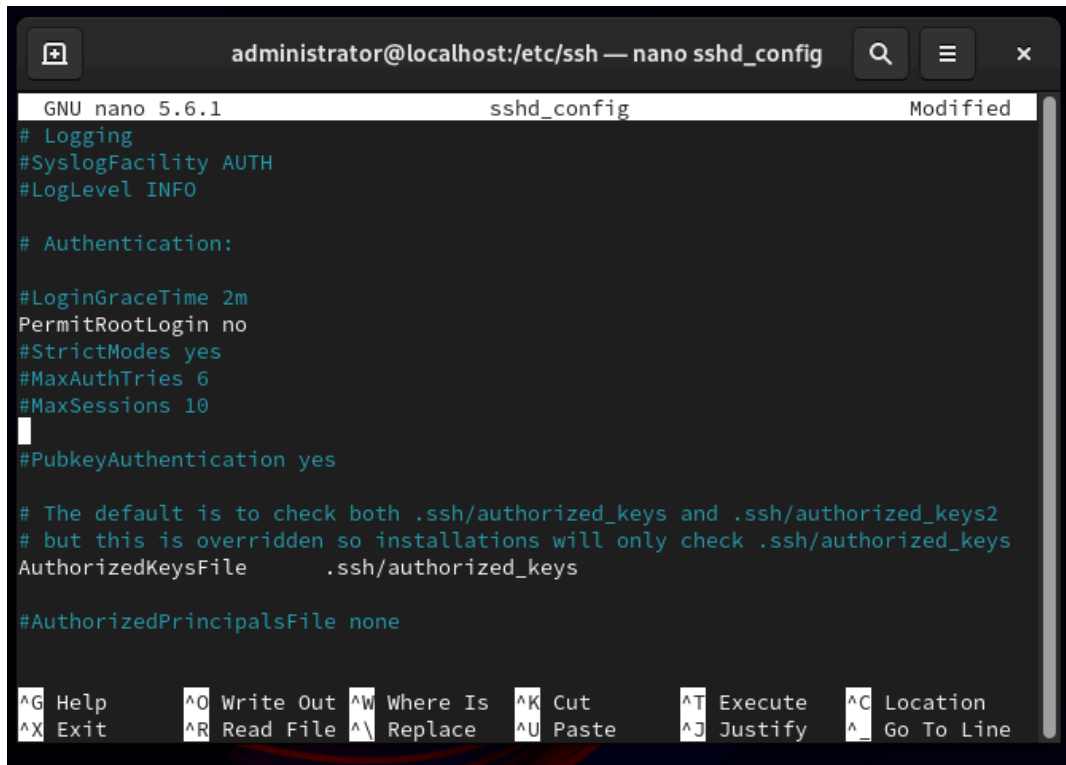
# To modify the system-wide sshd configuration, create a *.conf file under
# /etc/ssh/sshd_config.d/ which will be automatically included below
Include /etc/ssh/sshd_config.d/*.conf

# If you want to change the port on a SELinux system, you have to tell
# SELinux about this change.
# semanage port -a -t ssh_port_t -p tcp #PORTNUMBER
#
Port 2600
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

The bottom of the screen displays the nano editor's command shortcuts:

<code>^G</code> Help	<code>^O</code> Write Out	<code>^W</code> Where Is	<code>^K</code> Cut	<code>^T</code> Execute	<code>^C</code> Location
<code>^X</code> Exit	<code>^R</code> Read File	<code>^_\</code> Replace	<code>^U</code> Paste	<code>^J</code> Justify	<code>^_</code> Go To Line

2. Disable root login via ssh. This can be accomplished by editing the `/etc/ssh/sshd_config` file. You will uncomment the following line and type no (to block root from authenticating via SSH).



```
GNU nano 5.6.1 sshd_config Modified
# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
#PubkeyAuthentication yes

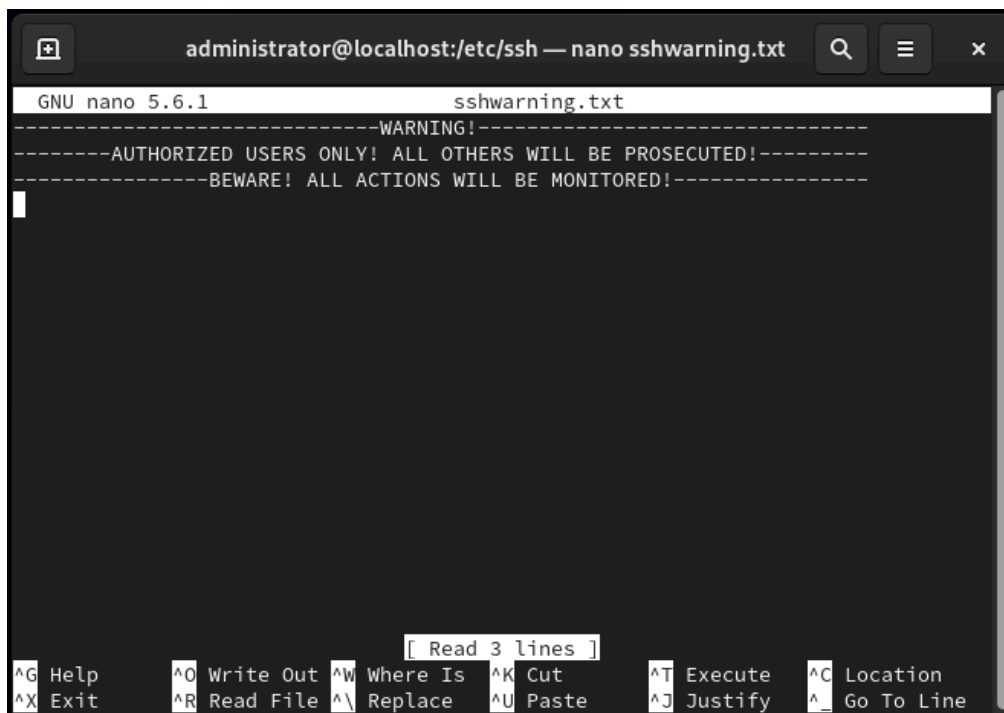
# The default is to check both .ssh/authorized_keys and .ssh/authorized_keys2
# but this is overridden so installations will only check .ssh/authorized_keys
AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

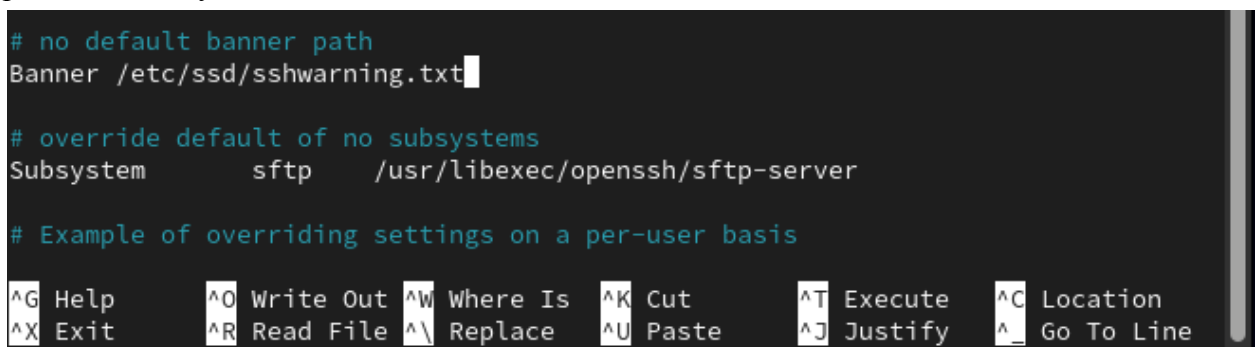
3. Create an ssh warning banner. This banner can serve as a warning to wannabe hackers trying to breach your system informing them that they will be prosecuted if they can be identified and caught in the act.

-First create a warning txt file in the /etc/ssh directory.



```
administrator@localhost:/etc/ssh — nano sshwarning.txt
GNU nano 5.6.1 sshwarning.txt
-----WARNING!-----
-----AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!-----
-----BEWARE! ALL ACTIONS WILL BE MONITORED!-----
[ Read 3 lines ]
^G Help  ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit  ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

- Next, add the SSH banner to the /etc/ssh/sshd_config file. You will have to put the full path of the file you want to use as the banner.



```
# no default banner path
Banner /etc/ssd/sshwarning.txt

# override default of no subsystems
Subsystem sftp /usr/libexec/openssh/sftp-server

# Example of overriding settings on a per-user basis
[ Read 3 lines ]
^G Help  ^O Write Out  ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit  ^R Read File  ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

- When testing to see if the SSH banner appears, I ran into an error. Here is what the error entails:

```
root@localhost:/var/log
ort 2500. For complete SELinux messages run: sealert -l bb5a03e2-7c35-41f7-8b82-00c8fe479507
Apr 14 16:07:03 localhost setroubleshoot[3776]: SELinux is preventing /usr/sbin/ssh from name_bind access on the tcp_socket p
ort 2500.#012#012***** Plugin bind_ports (92.2 confidence) suggests *****#012#012If you want to allow /u
sr/sbin/ssh to bind to network port 2500#012Then you need to modify the port type.#012Do#012# semanage port -a -t PORT_TYPE -
p tcp 2500#012 where PORT_TYPE is one of the following: ssh_port_t, vnc_port_t, xserver_port_t.#012#012***** Plugin catcha
ll_boolean (7.83 confidence) suggests *****#012#012If you want to allow nis to be enabled#012Then you must tell SE
Linux about this by enabling the 'nis_enabled' boolean.#012#012Do#012setsebool -P nis_enabled 1#012#012***** Plugin catchall
(1.41 confidence) suggests *****#012#012If you believe that sshd should be allowed name_bind access on
the port 2500 tcp_socket by default.#012Then you should report this as a bug.#012You can generate a local policy module to all
ow this access.#012Do#012allow this access for now by executing:#012# ausearch -c 'sshd' --raw | audit2allow -M my-sshd#012# s
emodule -X 300 -i my-sshd.pp#012
Apr 14 16:07:05 localhost setroubleshoot[3776]: SELinux is preventing /usr/sbin/ssh from name_bind access on the tcp_socket p
ort 2500. For complete SELinux messages run: sealert -l bb5a03e2-7c35-41f7-8b82-00c8fe479507
Apr 14 16:07:05 localhost setroubleshoot[3776]: SELinux is preventing /usr/sbin/ssh from name_bind access on the tcp_socket p
ort 2500.#012#012***** Plugin bind_ports (92.2 confidence) suggests *****#012#012If you want to allow /u
sr/sbin/ssh to bind to network port 2500#012Then you need to modify the port type.#012Do#012# semanage port -a -t PORT_TYPE -
p tcp 2500#012 where PORT_TYPE is one of the following: ssh_port_t, vnc_port_t, xserver_port_t.#012#012***** Plugin catcha
ll_boolean (7.83 confidence) suggests *****#012#012If you want to allow nis to be enabled#012Then you must tell SE
Linux about this by enabling the 'nis_enabled' boolean.#012#012Do#012setsebool -P nis_enabled 1#012#012***** Plugin catchall
(1.41 confidence) suggests *****#012#012If you believe that sshd should be allowed name_bind access on
the port 2500 tcp_socket by default.#012Then you should report this as a bug.#012You can generate a local policy module to all
ow this access.#012Do#012allow this access for now by executing:#012# ausearch -c 'sshd' --raw | audit2allow -M my-sshd#012# s
emodule -X 300 -i my-sshd.pp#012
[root@localhost log]#
```

Essentially, I was not aware that SELinux is enabled on my system. SELinux is a strict security enforcement mechanism for Unix/Linux systems that locks down the system using mandatory access control. Organizations that enforce SELinux are typically in the defense, government, or health sectors or any other sector that requires a Linux system to be locked down as much as possible. SELinux was preventing me from changing the default SSH port for the SSH daemon. For the purposes of this lab, I will not enforce SELinux since I am not in need of a highly secure Linux environment.

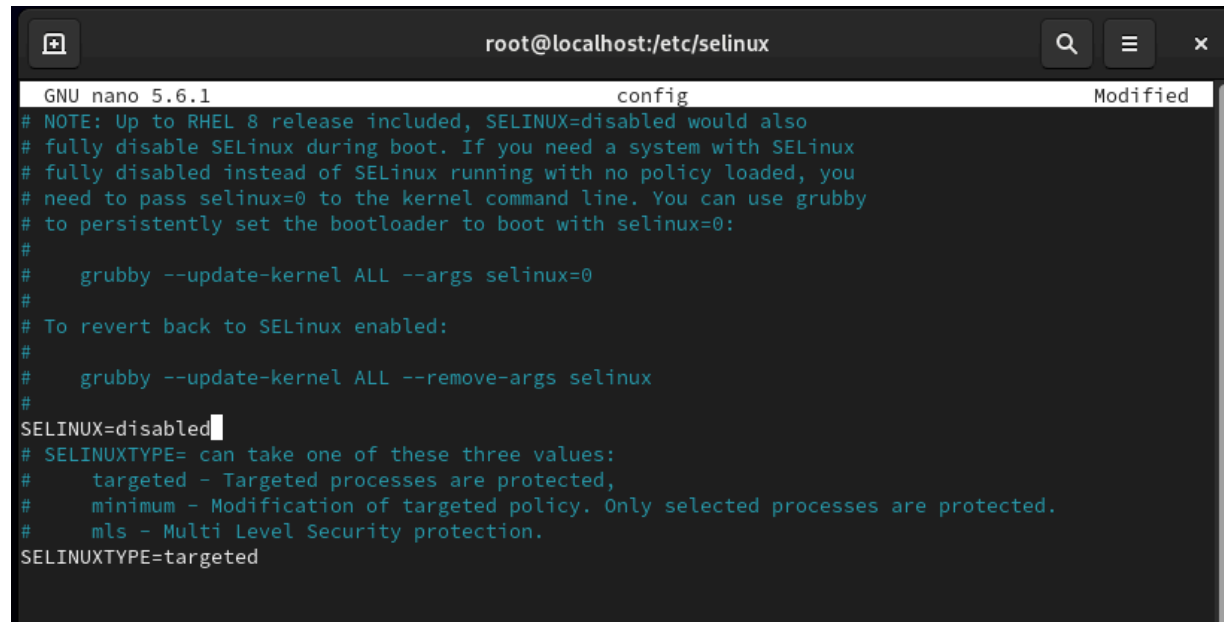
Solution to this issue:

1. I checked the status of SELinux

```
root@localhost:/var/log
[root@localhost log]# sestatus
SELinux status:                enabled
SELinuxfs mount:              /sys/fs/selinux
SELinux root directory:       /etc/selinux
Loaded policy name:            targeted
Current mode:                  enforcing
Mode from config file:         enforcing
Policy MLS status:             enabled
Policy deny_unknown status:    allowed
Memory protection checking:    actual (secure)
Max kernel policy version:     33
[root@localhost log]#
```

Well, it is an SELinux issue since it is enabled and enforced. I do not want it enforced though.

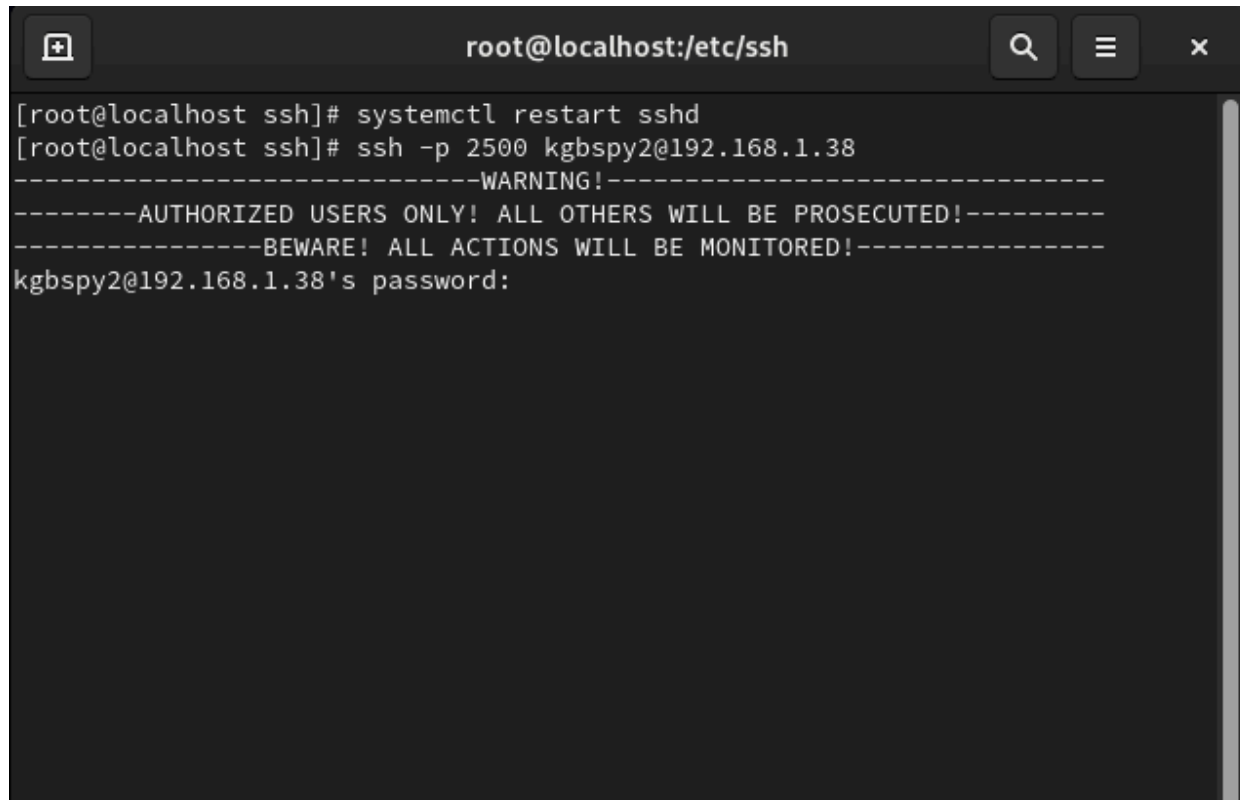
2. I disabled the enforcement of SELinux by editing the /etc/selinux/config configuration file. I set SELINUX=enforcing to SELINUX=disabled.



The screenshot shows a terminal window with the title 'root@localhost:/etc/selinux'. The window contains the contents of the /etc/selinux/config file, which is being edited with GNU nano 5.6.1. The file's title bar shows 'config' and 'Modified'. The content of the file includes instructions on how to disable SELinux during boot using grubby, and the current state where SELINUX is set to 'disabled' and SELINUXTYPE is set to 'targeted'.

```
GNU nano 5.6.1 config Modified
# NOTE: Up to RHEL 8 release included, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#   grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#   grubby --update-kernel ALL --remove-args selinux
#
SELINUX=disabled
# SELINUXTYPE= can take one of these three values:
#   targeted - Targeted processes are protected,
#   minimum - Modification of targeted policy. Only selected processes are protected.
#   mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

3. I then rebooted my RHEL vm and retried using SSH to test the banner. At first it didn't appear, but then I checked the path, and if you look at it it was /etc/ssd/sshwarning.txt when it should have been /etc/ssh/sshwarning.txt. Now it works since I corrected the spelling of the path.

A terminal window titled 'root@localhost:/etc/ssh' with search, menu, and close icons. The terminal shows the following commands and output:

```
[root@localhost ssh]# systemctl restart sshd
[root@localhost ssh]# ssh -p 2500 kgbspy2@192.168.1.38
-----WARNING!-----
-----AUTHORIZED USERS ONLY! ALL OTHERS WILL BE PROSECUTED!-----
-----BEWARE! ALL ACTIONS WILL BE MONITORED!-----
kgbspy2@192.168.1.38's password:
```

Section #4 System Customization:

1. Customize KGBHandler's bash shell

-Since KGBHandler is a sudo privileged user I will be using this user for a lot of tasks. Thus, having a customized terminal and bash shell will be beneficial. Also it would be nice to have a custom aesthetic for the terminal.

Step 1. Use an online bash prompt generator. I will use <https://ezprompt.net/> to customize my PS1 prompt. As you can see the PS1 prompt can be quite difficult to create manually so using a free prompt generator is more effective. I want KGBHandler's Bash prompt to display his username, hostname of the device, time in 12 hour format, and path of current directory.

EZPROMPT

Easy Bash PS1 Generator

1.) Pick the elements you want to use in your prompt.

Basic Elements

Status Elements

Date & Time Elements

Extra Characters

Username

Hostname

FQDN

Shell

Shell Version

Shell Release

Path To Current Directory

Current Directory

\$

2.) Select colors and rearrange elements here.

Username

@

Hostname

_

12hr Time

_

Path To Current Directory

Delete

Reset

FG



BG



Reset All

Delete All

3.) Preview the output.

user@host_01:08 PM_~/dir

4.) Copy and paste the code into your bashrc.

```
export PS1="\[\e[31m\]\u\[\e[m\]\[\e[31m\]@\[\e[m\]\[\e[31m\]\h\[\e[m\]\[\e[31m\]_\[\e[m\]\[\e[33m\]\@\[\e[m\]\[\e[31m\]_\[\e[m\]\[\e[33m\]\w\[\e[m\] "
```

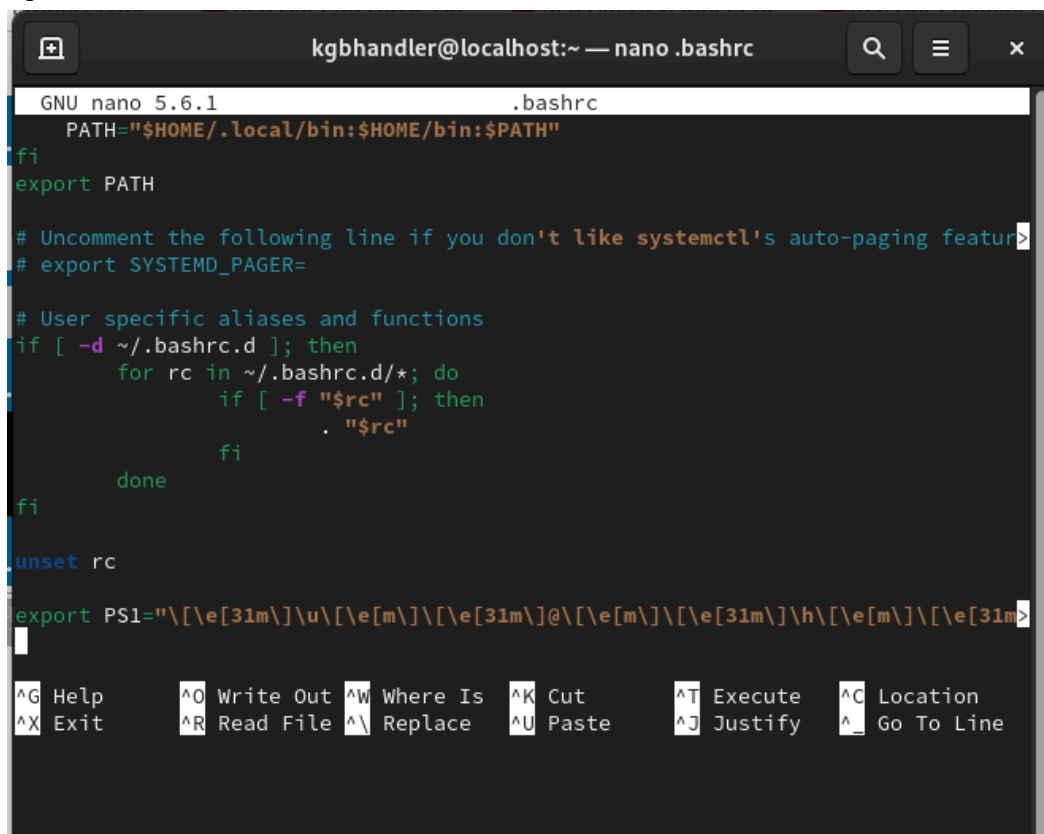
Step 2. On KGBHandler's bashrc file I will change the PS1 value provided by the generator I used. The bashrc file is a hidden file so I have to use ls -a to find it in KGBHandler's home directory.



A terminal window titled "kgbhandler@localhost:~" showing the output of the command `ls -a`. The output lists hidden files and directories in the user's home directory: `., .., .bash_logout, .bash_profile, .bashrc, .mozilla, .xauthcZ9FWZ`. The prompt is `[kgbhandler@localhost ~]$`.

```
[kgbhandler@localhost ~]$ ls -a
.  ..  .bash_logout  .bash_profile  .bashrc  .mozilla  .xauthcZ9FWZ
[kgbhandler@localhost ~]$
```

I will use nano to edit the `.bashrc` file to add the new prompt. I simply pasted the new PS1 prompt to the bottom of the `.bashrc` file.



A terminal window titled "kgbhandler@localhost:~ — nano .bashrc" showing the nano editor editing the `.bashrc` file. The editor shows the following content:

```
GNU nano 5.6.1 .bashrc
PATH="$HOME/.local/bin:$HOME/bin:$PATH"
fi
export PATH

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

# User specific aliases and functions
if [ -d ~/.bashrc.d ]; then
  for rc in ~/.bashrc.d/*; do
    if [ -f "$rc" ]; then
      . "$rc"
    fi
  done
fi

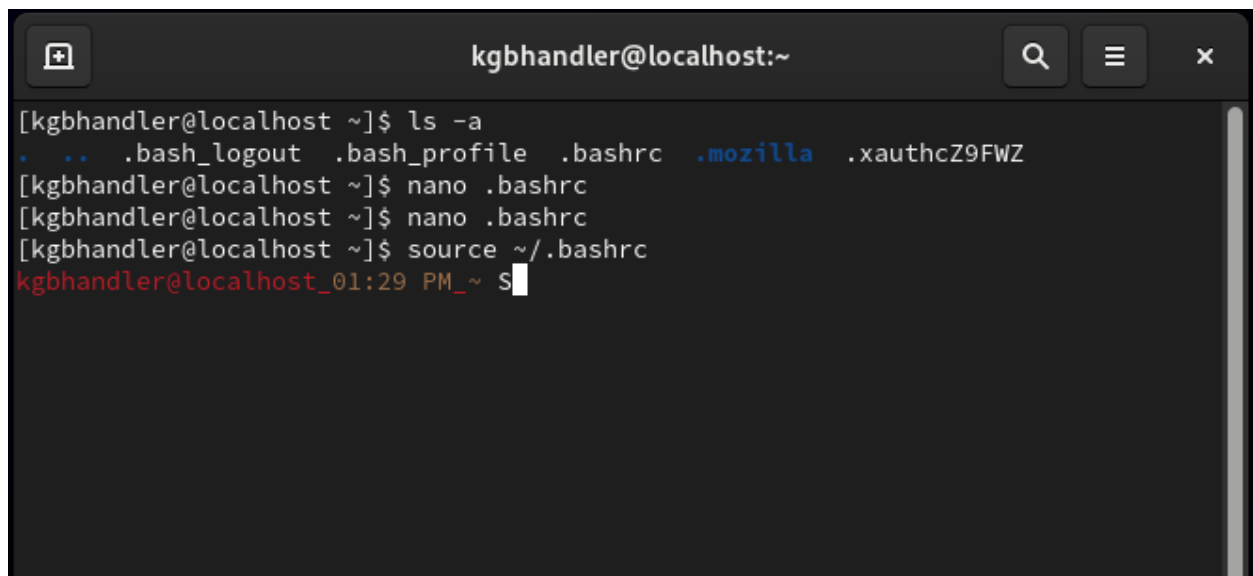
unset rc

export PS1="\[\e[31m\]\u\[\e[m\]\[\e[31m\]@\[\e[m\]\[\e[31m\]\h\[\e[m\]\[\e[31m\]>
```

The bottom of the window shows nano editor shortcuts:

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location
^X Exit	^R Read File	^_ Replace	^U Paste	^J Justify	^_ Go To Line

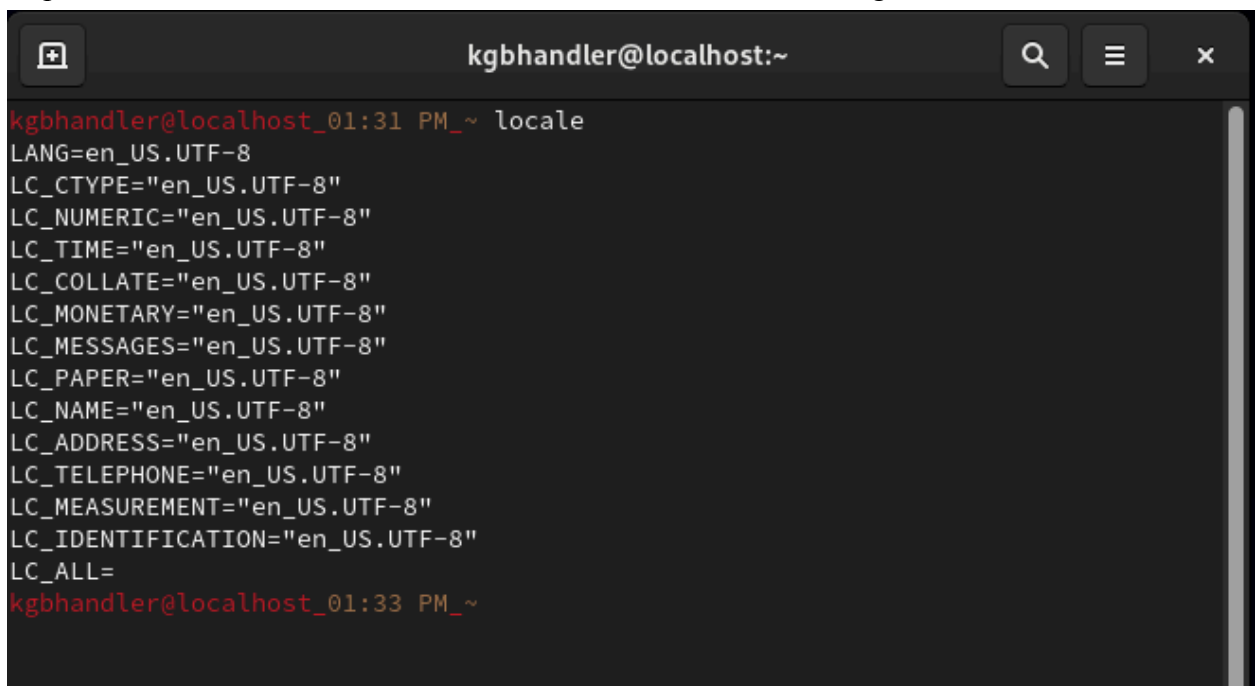
To make the changes apply right away I used `source ~/.bashrc`.



```
kgbhandler@localhost:~  
[kgbhandler@localhost ~]$ ls -a  
.  ..  .bash_logout  .bash_profile  .bashrc  .mozilla  .xauthcZ9FWZ  
[kgbhandler@localhost ~]$ nano .bashrc  
[kgbhandler@localhost ~]$ nano .bashrc  
[kgbhandler@localhost ~]$ source ~/.bashrc  
kgbhandler@localhost_01:29 PM_~ $
```

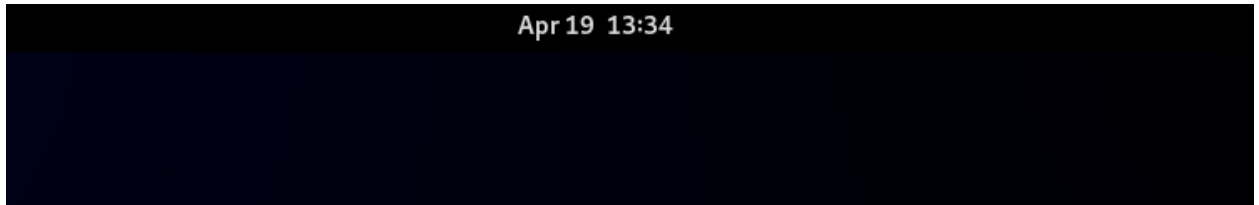
2. Change the time format on the system

Step 1. Use the locale command to determine the current time settings.

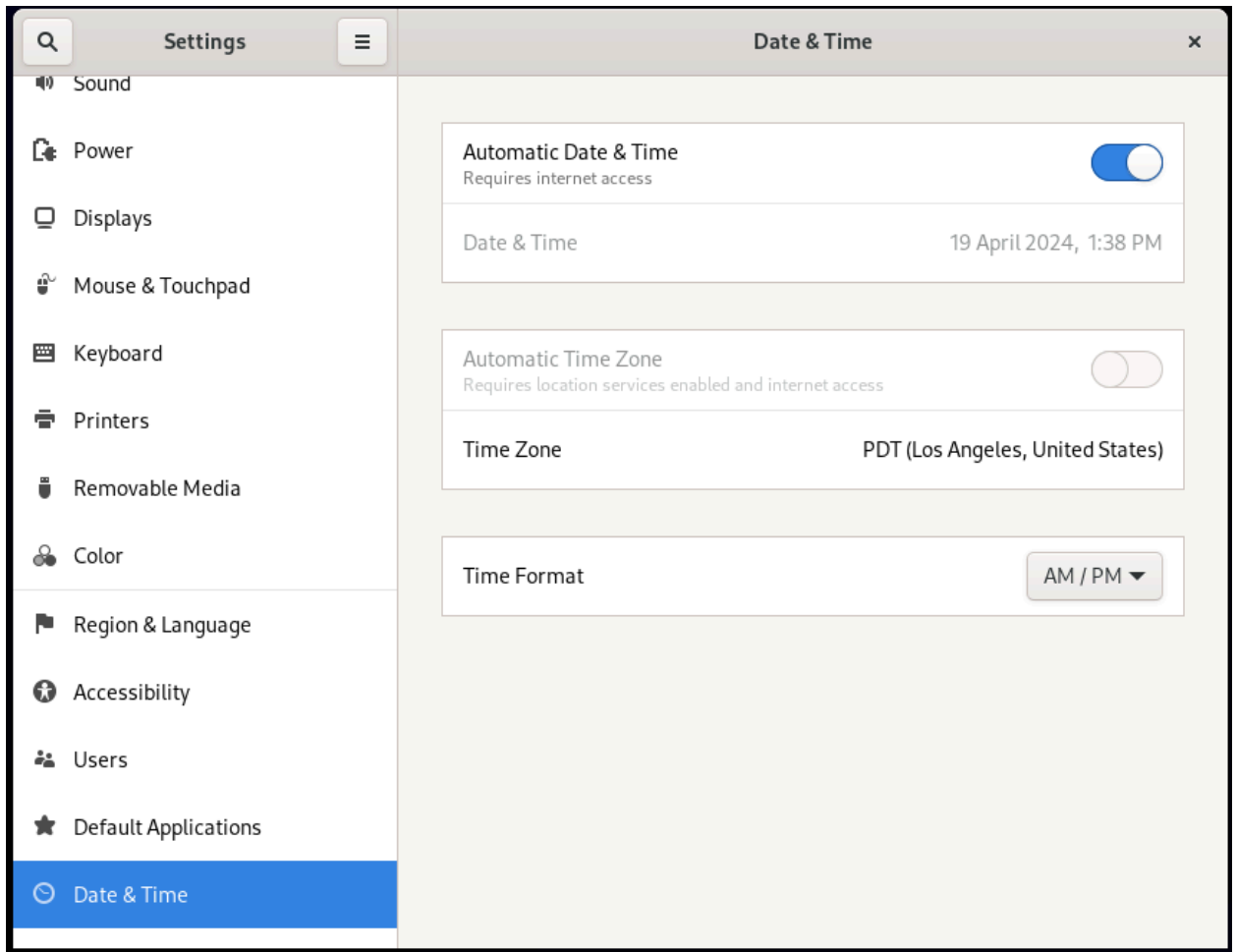


```
kgbhandler@localhost_01:31 PM_~ locale  
LANG=en_US.UTF-8  
LC_CTYPE="en_US.UTF-8"  
LC_NUMERIC="en_US.UTF-8"  
LC_TIME="en_US.UTF-8"  
LC_COLLATE="en_US.UTF-8"  
LC_MONETARY="en_US.UTF-8"  
LC_MESSAGES="en_US.UTF-8"  
LC_PAPER="en_US.UTF-8"  
LC_NAME="en_US.UTF-8"  
LC_ADDRESS="en_US.UTF-8"  
LC_TELEPHONE="en_US.UTF-8"  
LC_MEASUREMENT="en_US.UTF-8"  
LC_IDENTIFICATION="en_US.UTF-8"  
LC_ALL=  
kgbhandler@localhost_01:33 PM_~
```

Step 2. The time is correct on the bash terminal, but on my GNOME desktop it is in military time, but I want it to be in 12 hour time format.

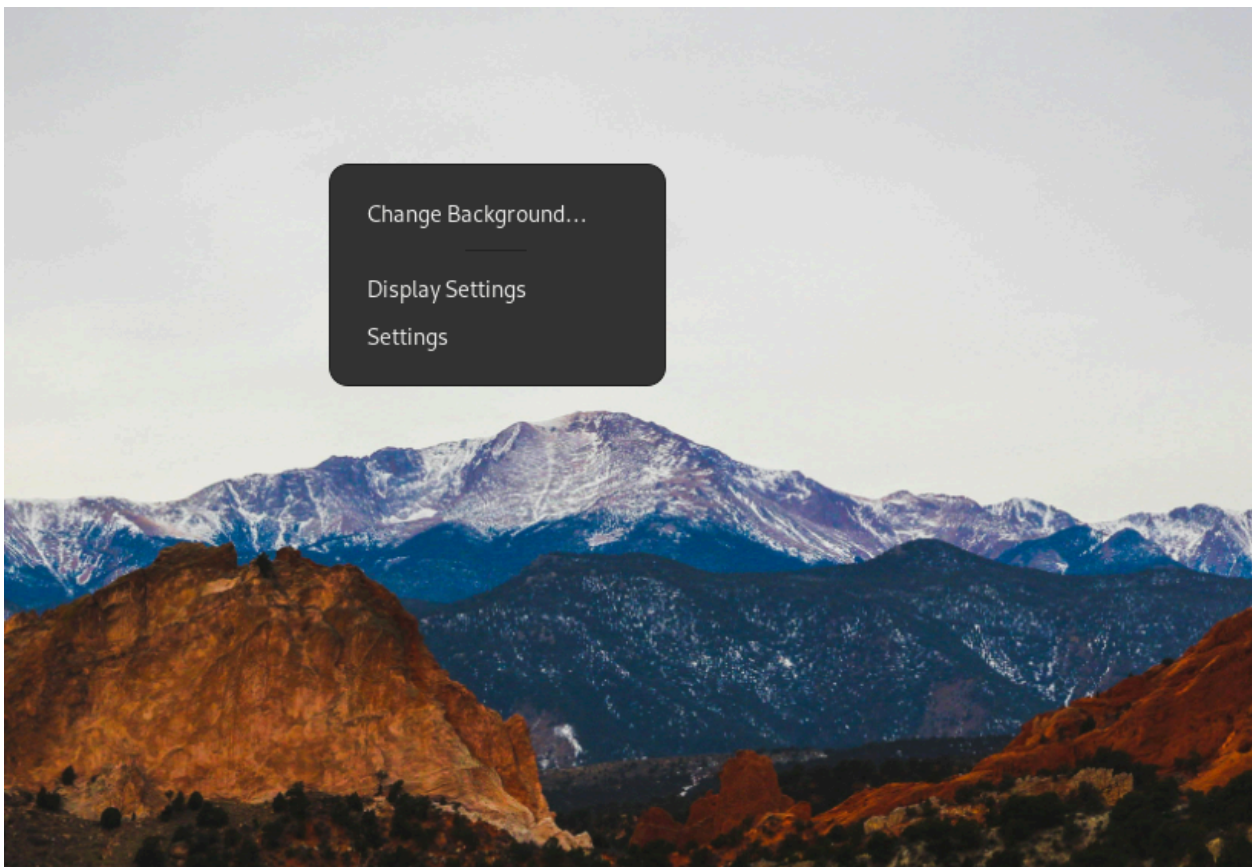


Step 3. To change this setting I will go to the settings for my GNOME desktop. I went to settings, Date & Time, and then changed the Time Format from 24 hour to AM/PM.



3. Change wallpaper

I will change my wallpaper for my RHEL system. Simply right click on the background in use and click on change background. I changed my background already in the following screenshot.



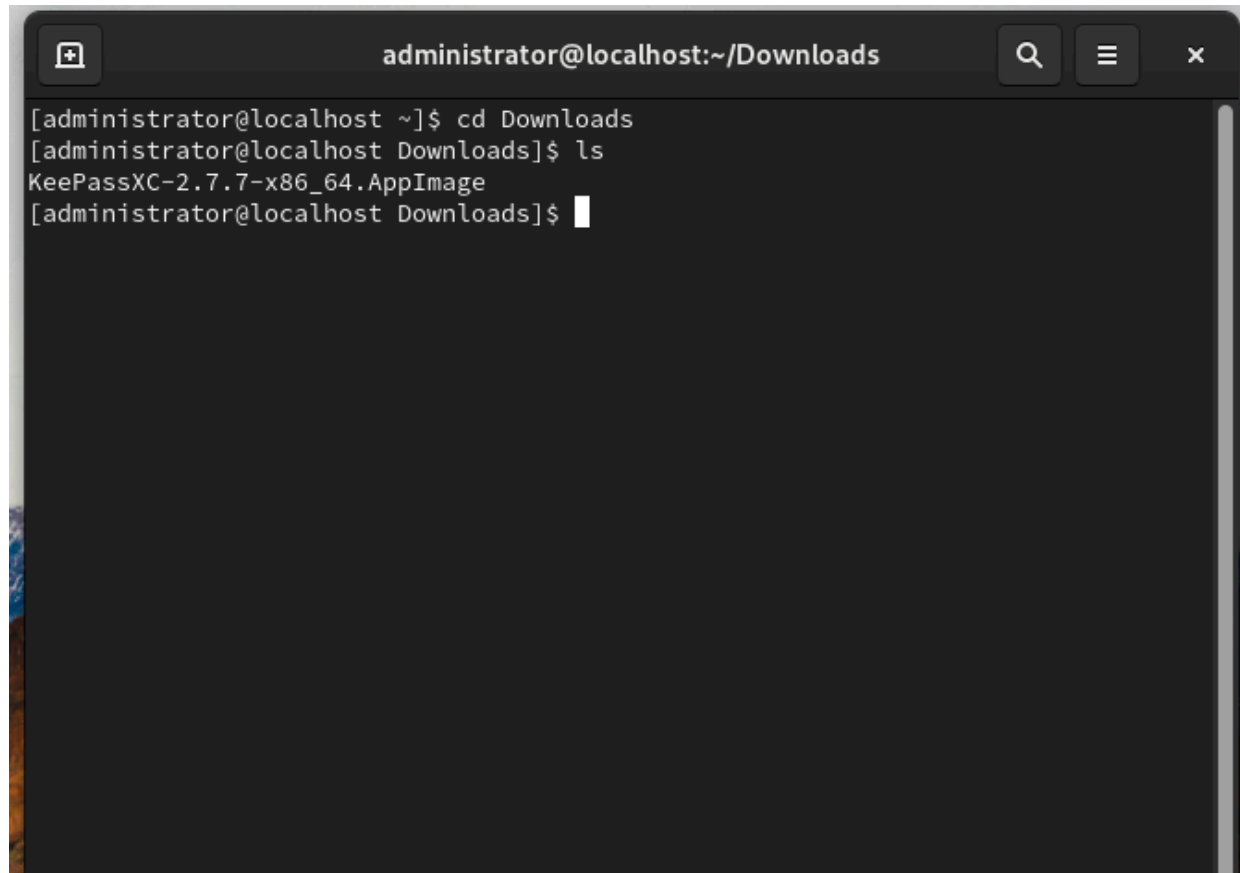
Section #5 Installing key services and applications

1. Install a password manager

Reasoning: Having a password manager will ensure that I can safely and efficiently manage passwords on my Linux VM. I will be installing KeePassxc.

Step 1. Install KeePassxc on my Linux system. I will be installing KeePassxc by downloading the latest version's software onto my VM, then installing it manually.

- Here is the downloaded app image file:

A terminal window titled 'administrator@localhost:~/Downloads' with search, menu, and close buttons. The terminal shows the following commands and output:

```
[administrator@localhost ~]$ cd Downloads
[administrator@localhost Downloads]$ ls
KeePassXC-2.7.7-x86_64.AppImage
[administrator@localhost Downloads]$
```

- Now I will move the app image file to /usr/share so that any user can access this executable and use it.

```
administrator@localhost:~/Downloads
[administrator@localhost Downloads]$ ls
KeePassXC-2.7.7-x86_64.AppImage
[administrator@localhost Downloads]$ sudo mv ~/Downloads/KeePassXC-2.7.7-x86_64.AppImage /usr/share/
[sudo] password for administrator:
[administrator@localhost Downloads]$
```

- Since it is now located in /usr/share/ I will use chmod +x so that all users can execute the app image and use KeePassxc.

```
administrator@localhost:/usr/share — keepassxc
grilo-0.3          trackers
grilo-plugins      tracker3-miners
groff              tuned
grub              udica
gststreamer-1.0    usb_modeswitch
gtk-3.0            vim
gtk-4.0            vulkan
gtksourceview-4     wayland-sessions
gutenprint         wireplumber
gvfs               X11
help               xdg-desktop-portal
hplip              xfsprogs
hwdata             xml
hyphen             xsessions
i18n               yelp
ibus               yelp-tools
icons              yelp-xsl
idl                zenity
info               zoneinfo
iso-codes           zsh
itstool
[administrator@localhost share]$ sudo chmod +x KeePassXC-2.7.7-x86_64.AppImage
[administrator@localhost share]$ ls
accountsservice    kdump
aclocal             KeePassXC-2.7.7-x86_64.AppImage
adobe               libdrm
alsa                libgnumekbd
alsa-card-profile   libgpg-error
```

- The app can be execute by typing in ./name of the app

```
administrator@localhost:/usr/share — keepassxc
groff          tuned
grub           udica
gstreamer-1.0  usb_modeswitch
gtk-3.0        vim
gtk-4.0        vulkan
gtksourceview-4 wayland-sessions
gutenprint     wireplumber
gvfs           X11
help           xdg-desktop-portal
hplip          xfsprogs
hwdata         xml
hyphen         xsessions
i18n           yelp
ibus           yelp-tools
icons          yelp-xsl
idl            zenity
info           zoneinfo
iso-codes      zsh
itstool

[administrator@localhost share]$ ./KeePassXC-2.7.7-x86_64.AppImage
Warning: Ignoring XDG_SESSION_TYPE=wayland on Gnome. Use QT_QPA_PLATFORM=wayland
to run on Wayland anyway.
QObject::startTimer: Timers cannot have negative intervals
qt.network.ssl: QSocket: cannot resolve EVP_PKEY_base_id
qt.network.ssl: QSocket: cannot resolve SSL_get_peer_certificate
qt.network.ssl: QSocket: cannot call unresolved function SSL_get_peer_certificate

```