~~~~
### :star:  RHCSA
  ---
![image](image/redhat.png)<br>
&nbsp;&nbsp;[![License: MIT](https://img.shields.io/badge/License-MIT-blue.svg)]
(https://opensource.org/licenses/MIT "MIT License")
&nbsp;&nbsp;[![Active](http://img.shields.io/badge/Status-Active-green.svg)]
(https://github.com/soficx/)
 [![Maintenance](https://img.shields.io/badge/last%20updated-20--02--2022-pink)]
(https://github.com/soficx)

  ---
**Description:** This is a collection of tasks,labs  that i used to practice and prepare for The
Red Hat Certified System Administrator (RHCSA)
The information below is gatehered  from different [resources](#resources) books (Asghar Ghori,
Sander van Vugt) github repos, telegram and slack channels

---
## Talble of Contents
- [Understand and use essential tools](#understand-and-use-essential-tools)
- [Create simple shell scripts](#create-simple-shell-scripts)
- [Operate running systems](#operate-running-systems)
- [Configure local storage](#configure-local-storage)
- [Create and configure file systems](#create-and-configure-file-systems)
- [Automount](#automount)
- [Layered Storage](#layered-storage)
- [Deploy configure and maintain systems](#deploy-configure-and-maintain-systems)
- [Manage basic networking](#manage-basic-networking)
- [Manage users and groups](#manage-users-and-groups)
- [Manage security](#manage-security)
- [Manage containers](#manage-containers)
- [Resources](#resources)

### Understand and use essential tools

- ***Task 1 :***<br>
        Find all setuid files on the system and save the list
```
 # find / -type f -perm -4000 > setuid_list
```


-  ***Task 2 :*** <br>
        Find all log messages in /var/log/messages that contain "ACPI", and export them to a file
called /root/logs.
        Then archive all of /var/log and save it to /tmp/log_archive.tgz

```
# sudo grep ACPI /var/log/messages >> /root/logs
# tar cvf /tmp/log_archive.tgz /var/log/
```
- ***Task 3 :*** <br>
        Create tar files compressed with gzip and bzip2 of /home and extract them

```
    Gzip:
  # tar cvfz home.tar.gz /home
  # tar -tf home.tar.gz
  # tar xvfz home.tar.gz

   Bzip2:
  # tar cvfj home.tar.bz2 /home
  # tar -t home.tar.dz2
  # tar xvfj home.tar.bz2
```

-  ***Task 4 :***<br>
        Create an empty file hard1 under /tmp and display its attributes.
        Create hard links hard2 and hard3. Edit hard2 and observe the attributes. Remove hard1 and
hard3 and list the attributes again
~~~~

```
  # cd /tmp
  # touch hard1
  # ls -li hard1

  # ln hard1 hard2
  # ln hard1 hard3

  # echo "Redhat" > hard2
  # ls -li                        ==> They have the same inode
  # rm -f hard{1,3}
  # ls -li
```

-  ***Task 5 :***<br>
        Create an empty file soft1 under /root pointing to /tmp/hard2. Edit soft1 and list the attributes after editing. Remove hard2 and then list soft1

```
  # ln -s /tmp/hard2 /root/soft1
  # ls -li /root/soft1   also they've the same inode
  # echo "from soft1" >> /root/soft1
  # ls -li soft1 /tmp/hard2
  # rm -f /tmp/hard2            ==> The link is broken
```

-  ***Task 6 :***<br>
         Create a file perm_file1 with read permissions for owner, group and other.
         Add an execute bit for the owner and a write bit for group and public.
         Revoke the write bit from public and assign read, write, and execute bits to the three user categories at the same time.
         Revoke write from the owning group and revoke write and execute bits from public
```
  # touch perm_file1
  # chmod 444 permu_file1
  # chmod u+x,g+w,o+w permu_file1
  # chmod o-w,a=rwx permu_file1
  # chmod g-w o-wx permu_file1
```
## Create simple shell scripts

-  ***Task 1 :***<br>
        Create a bash script that display the hostname ,system information and the users that  are currently logged in
```
# vim /usr/local/bin/sysinfo.sh
        #!/bin/bash
        echo "================="
        echo " Display sys info "
        echo
        /usr/bin/hostnamectl
        echo "================"
        echo " The users that are currently logged in: "
        /usr/bin/who
# chmod +x $_
```
-  ***Task 2 :***<br>
        Create a bash script that shows total count of the supplied arguments , value of the first argument, PID of the script and all the supplied arguments
```
# vim /usr/local/bin/argts.sh
        #! /bin/bash
        echo "========================="
        echo "The total count of the supp argts: $# "
        echo "The value of the first arg is: $1"
        echo "The PID of the script is : $$"
        echo "All the argts: $*"
# chmod +x $_
```

~~~
-  ***Task 3 :***<br>
        Create a bash script that can create user10, user20 and user30 accounts with each account
is create a message saying " The account is created successfuly " will be displayed otherwise the
script will terminate .
        In case of a successful account creation assign the user account a password as their
username

```
There are multiple methods to do that
# vim user_add.sh
        #!/bin/bash

        PASS=$(which passwd)
        ADD=$(which useradd)

        for user in user{10..30..10};do
                echo "Creating user : $user"
                $ADD $user > /dev/null 2>&1                     # to ignore the output that
generates from the useradd cmd

                if [ $? = 0 ]
                then
                        echo "==============="
                        echo " The account is create successfuly"
                        echo $user | $PASS --stdin $user
                else
                        echo "Failed to create account $user"
                fi
# chmod +x $_
```
-  ***Task 4:***<br>
        Write a script that finds all the files owned by new_user and have size greater than 30KB
and less than 50KB  and store them in /tmp/

```
# vim find_files.sh
        #!/bin/bash
        find / -type f -user new_user -size +30k -size -50k -exec cp {} /tmp/ \;
# chmod +x $_
```
-  ***Task 5 :***<br>
        Write a script named backup.sh under /root which will search files less than 2M from /usr
and store it in /root/backup
```
# vim /root/backup.sh
        find /usr/ -type f -size -2M -exec cp '{}' /root/backup \;
# chmod +x $_
```
-  ***Task 6 :***<br>
        As a System Administrator you are responsible to take a backup of your /etc directory
every night. Build a shell script to take a backup of the /etc/directory using the tar command.
The backup script should be named as /root/backup.sh. Schedule this script to run at 11:00 PM
every night – except Sundays.
```
# vim /root/backup.sh
        #!/bin/bash
        # you can add the file name with the date to keep all the changes
        tar -cvf etc_backup.tar /etc/
# chmod +x /root/backup.sh
#/root/backup.sh
# crontab -e
* 23 * * 1-6 /root/backup.sh
```

### Operate running systems

-  ***Task 1 :***<br>
        Modify the GRUB timeout and make it 1 second instead of 5 seconds
~~~

~~~
```
 Any change to /etc/default/grub require rebuilding the grub.cfg
# vim /etc/default/grub
   GRUB_TIMEOUT=1
- On BIOS-based machine :
# grub2-mkconfig -o /boot/grub2/grub.cfg
- On UEFI-based machine :
# grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
# reboot
        To determine if the system is booted on BIOS or UEFI mode
         # dmesg | egrep -i "efi|bios"
```

-  ***Task 2 :***<br>
        Terminate the boot process at an early stage to access a debug shell to reset the root
password
```
 == after you reboot the machine | press e to the specific kernel
 and add rd.break before rgb quiet
 # mount -o remount,rw /sysroot
 # passwd               ==> changing the pass
 # touch /.autorelable
 # exit
 # exit

  If you follow this approach and you have a large data then the process of relabling by SELinux
takes more time
  So to save some time you can add the option :
  rd.break enforcing=0 (without create the /.autrelable file)
  affter you change the password and exit ,restore the lable of /etc/shadow
 # restorecon -v /etc/shadow
 # setenforce 1
```
-  ***Task 3 :***<br>
         Download the latest available kernel packages from the Red Hat Customer Portal and
install them
```
To check out the kernel package information you can use dnf :
# dnf info kernel
To update to the latest  kernel
# dnf install kernel --best
# reboot

```
-  ***Task 4 :***<br>
         Install the tuned service, start it and enable it for auto-restart upon reboot. Display
all available profiles and the current active profile. Switch to one of the available profiles and
confirm. Determine the recommended profile for the system and switch to it. Deactive tuning and
reactivate it
```
# systemctl status tuned.service        => Check if the service is installed
# dnf install tuned
# tuned-adm list
# tuned-adm profile powersave
# tuned-adm active                      => to confirm which of the profile is active
# tuned-adm profile $(tuned-adm recommned)
```
-  ***Task 5:***<br>
        Launch the command dd if=/dev/zero of=/dev/null three times as a background job.
        Increase the priority of one of these ps
        Change the priority of the same process again, but this time use the value -15. Observe
the difference.
        Kill all the dd processes you just started
```
# dd if=/dev/zero of=/dev/null &
# dd if=/dev/zero of=/dev/null &
# dd if=/dev/zero of=/dev/null &
# ps aux | grep "dd "
~~~

~~~~
# renice -n 5 -p PID
# sudo renice -n -15 -p PID
# killall dd
# jobs
```
-  ***Task 6 :***<br>
        Configure the journal to be persistent across system reboots.
        Make a configuration file that writes all messages with an info priority to the file
/var/log/messages.info.
        Configure logrotate to keep ten old versions of log files
```
# vim /etc/systemd/journald.conf
        #uncomment storage line to persistent, a dir in /var/log/journal/ will be created
        Storage=persistent
# systemctl restart systemd-journald.service
# ls /var/log/journal
 " by restarting the systemd-journald , you 'll loose all the logging of the current session
 so it's recommended to do the following cmd :
# killall -USR1 systemd-journald

# vim /etc/rsyslog.d/info.conf
        *.info                  /var/log/messages.info
# systemctl restart rsyslogd
# logger -p daemon.debug "this message from Task6"
# tail /var/log/messages.info
 Configure logrotate:
# vim /etc/logrotate.conf
        rotate 10
```
-  ***Task 7:***<br>
        Copy the /etc/hosts file to the /tmp direcotry on server2 using scp command.
        Try to connect to server2 as user root and copy the /etc/passwd file to you home direcotry
```
# scp /etc/hosts server2:/tmp/
# scp server2:/etc/passwd ~
```

-  ***Task 8 :***<br>
        Donwload and install the apache web service. try to configure apache to log error messages
through syslog using the facility local1
        Create a rule that send all messages that it receives from local1 (That used above)
facility to /var/log/httpd-error.log
        verify the last changed by accessing a page that does not exist
```
# dnf install -y httpd
# vim /etc/httpd/conf/http.conf
        # adding the line
        ErrorLog        syslog:local1
# systemctl restart httpd
# vim /etc/rsyslog.conf
        # adding the following line in rule section
        local1.error            /var/log/httpd-error.log
# systemctl restart rsyslogd
# curl localhost

```
### Configure local storage

-  ***Task 1 :***<br>
        Create a 2 GB gpt partition and format the partition with xfs and mount the device
persistently

```
# lsblk         => to list all the block device
# fdisk /dev/sdb
# press m   => list all the opts
# enter g   => to create a new gpt part
# enter n   => to add new part
# accept the default part
~~~~

~~~~
# accept the default starting sector
# For the ending sector , Enter +2G
# enter w   => to write table to disk and exit
# partprobe => to inform the os part table change

# mkfs.xfs /dev/sdb1
# blkid   => to grep the uuid
# mkdir /mnt/task1
# echo 'UUID=xxxx /mnt/task1 xfs defaults 0 0' >> /etc/fstab
# mount -a
```

-  ***Task 2 :***<br>
         Assign partition type "msdos" to /dev/sdc for using it as an MBR disk. Create and confirm
a 100MB primary partition on the disk

```
same steps created in task 1 expet the size and the type "msdos"
# enter o   => to create an MBR disk
```

-  ***Task 3 :***<br>
        Delete the sdb1 partition that was created in Task1 above
```
# lsblk         => to see the moint point
# umount /mnt/task1             && comment the line in /etc/fstab
# mount -a
# fdisk /dev/sdb
# enter d       => to remove the part
# press w   => to save
# partprobe
```

-  ***Task 4 :***<br>

        Initialise one partition sdb1 (90MB) and one disk sdc (250MB) for use in LVM. Create a
volume group called vgbook and add both physical volumes to it. Use the PE size of 16MB and list
and display the volume group and the physical volumes
```
# fdisk /dev/sdb
# enter n   => to add new part
# accept the default part
# accept the default starting sector
# For the ending sector , Enter The size of the part +90M
# press t       => to change the type of part to Linux lvm
# 8e
# enter w   => to write table to disk and exit
# partprobe => to inform the os part table change

# pvcreate /dev/sdc
# vgcreate vgbook /dev/sdb1 /dev/sdc -s 16M
# pvs
# pvdisplay
# vgs
# vgdisplay vgbook
```
-  ***Task 5:***<br>
        Create two logical volumes, lvol1 and lvbook1, in the vgbook volume group. Use 120MB for
lvol0 and 192MB for lvbook1. Display the details of the volume group and the logical volumes
```
# lvcreate --name lvol0 -L 120M vgbook
# lvcreate --name lvbook1 -L 192 vgbook
# lvs
# lvdisplay /dev/vgbook/lvm_name
```
-  ***Task 6 :***<br>
        Add another partition sdb2 of size 158MB to vgbook to increase the pool of allocatable
space. Initialise the new partition prior to adding it to the volume group. Increase the size of
lvbook1 to 336MB. Display the basic information for the physical volumes, volume group, and
~~~~

logical volume
```
# fdisk /dev/sdb
# enter n
# accept the default part
# accept the default starting sector
# For the ending sector , Enter The size of the part +158M
# enter w
# partprobe
# pvcreate /dev/sdb2
# vgextend vgbook /dev/sdb2
# lvextend -L +144M /dev/vgbook/lvbook1
# lvs
```

- ***Task 7 :***<br>
        Rename lvol0 to lvbook2. Decrease the size of lvbook2 to 50MB using the lvreduce command and then add 32MB with the lvresize command. Remove both logical volumes. Display the summary for the volume groups, logical volumes, and physical volumes
```
# lvrename /dev/vgbook/lvol0 lvbook2
# lvreduce -L -50M /dev/vgbook/lvbook2
# lvresize -L +32M /dev/vgbook/lvbook2
# lvremove /dev/vgbook/lvbook*
lvs, vgs, pvs
```
- ***Task 8 :***<br>
        Uninitialise all three physical volumes - sdb1, sdb2, and sdb - by deleting the LVM structural information from them. Use the pvs command for confirmation. Remove the partitions from the sdd disk and verify that all disks are now in their original raw state
```
# vgremove vgbook
# pvremove /dev/sdb1 /dev/sdb2 /dev/sdc
# pvs
# wipefs -af /dev/sdb /dev/sdc
# lsblk
```
- ***Task 9 :***<br>
        Create a new logical volume (LV-A) with a size of 30 extents that belongs to the volume group VG-A (with a PE size of 32M). After creating the volume, configure the server to mount it persistently on /mnt

```
# pvcreate /dev/sdb
# vgcreate vg-a /dev/sdb -s 32M
# lvcreate --name lv-a vg-a -l 30M
# mkfs.xfs /dev/vg-a/lv-a
# blkid
# echo "UUID=XXXX /mnt xfs defaults 0 0 ">> /etc/fstab
```
- ***Task 10 :***<br>
        Create 1 swap area in a new 40MB partition called sdc3 using the mkswap command. Create another swap area in a 140MB logical volume called swapvol in vgfs. Add their entries to the /etc/fstab file for persistence. Use the UUID and priority 1 for the partition swap and the device file and priority 2 for the logical volume swap. Activate them and use appropriate tools to validate the activation
```
  After adding an sdc part
# mkswap --label sdc3 /dev/sdb3
# echo "UUID=XXXXX swap swap defaults,pri=1 0 0 " >> /etc/fstab
# mount -a
# free -m
# swapon /dev/sdc3
# free -m
# lvcreate --name swapvol -L 140M vgfs                          after you create: vgfs
# mkswap --lable swpvol /dev/vgfs/swapvol
# echo "swpvol swap swap defaults,pri=2 0 0" >> /etc/fstab
# mount -a
# swapon -s
```

~~~~
```

### Create and configure file systems <br>

-  ***Task 1 :***<br>
        Create 2x100MB partitions on the /dev/sdb disk, initialise them separately with the Ext4
and XFS file system types
        - create mount points called /ext4fs and /xfs1
        - attach them to the directory structure, verify their availability and usage
        - mount them persistantly using their UUIDS

```
# fdisk /dev/sdb
# press m   => list all the opts
# enter n   => to add new part
# accept the default part
# accept the default starting sector
# For the ending sector , Enter The size of the part +100M
# press n       => to add another part as above
# enter w   => to write table to disk and exit
# partprobe => to inform the os part table change

File sys type :
# mkfs.ext4 /dev/sdb1
# mkfs.xfs /dev/sdb2
Create the moint point
# mkdir /ext4fs
# mkdir /xfs1
# mount /dev/sdb1 /ext4fs
# mount /dev/sdb2 /xfs1
# blkid         => to grep the UUID
# echo "UUID=xxxx /ext4fs ext4 defaults 0 0 " >> /etc/fstab
# echo "UUID=xxxx /xfs    xfs  defaults 0 0 " >> /etc/fstab
# mount -a
```

-  ***Task 2 :***<br>
         Create a volume group called vgfs comprised of a 160MB physical volume created in a
partition on the /dev/sdb disk. The PE size for the volume group should be set at 16MB. Create 2
logical volumes called ext4vol and xfsvol of size 80MB each and initialise them with the Ext4 and
XFS file system types. Ensure that both file systems are persistently defined using their logical
volume device filenames. Create mount points /ext4fs2 and /xfsfs2, mount the file systems, and
verify their availability and usage
```
Initialize the physical volume for use by LVM
# pvcreate /dev/sdb
# pvs
# vgcreate vgfs /dev/sdb -s 16MB
# lvcreate --name ext4vol -L 80MB vgfs
# lvcreate --name xfsvol -L 80MB vgfs
# vgs
# mkfs.ext4 /dev/vgfs/ext4vol
# mkfs.xfs /dev/vgfs/xfsvol
# mount /dev/vgfs/ext4vol /ext4fs2
# mount /dev/vgfs/xfsvol /xfsfs2
# blkid         => grep the UUID and add the appropriate lines in /etc/fstab
# df -hT
```

-  ***Task 3 :***<br>
        Grow the size of the vgfs volume group that was created above by adding another  disk to
it. Extend the ext4vol logical volume along with the file system it contains by 40MB
```
# vgextend vgfs /dev/sdc
# lvextend -L +40 -r /dev/gvfs/ext4fs   => -r : to extend the file system at the same time
# lvs
```

-  ***Task 4 :***<br>
~~~~

~~~
         Create a directory called /common and export it to server in read/write mode. Ensure that
NFS traffic is allowed through the firewall. Confirm the export
```
In server1:
# mkdir /common
# dnf -y install nfs-utils
# systemctl enable --now nfs-server.service
# firewall-cmd --add-service=nfs --permanent
# firewall-cmd --add-service=rpc-bind --permanent
# firewall-cmd --add-service=mountd --permanent
# firewall-cmd --reload
# echo "/common  *(rw)" >> /etc/exports
# exportfs -av          => to confirm the export
```
-  ***Task 5 :***<br>
         Mount the /common that exported in task 4 . Create a mount point called /local. Add the
remote share to the file system table for persistence. Create a test file in the mount point and
confirm the file creation on the NFS server

```
In server2:
# mkdir /local
# showmount -e server1                  => or you can use the ip @ of the server
# mount server1:/ /local
# echo "server1:/common  /local  nfs _netdev 0 0" >> /etc/fstab
# mount -a
# echo "This message from server2 " >> /local/file
In server1:
# cat /common/file
```
-  ***Task 6 :***<br>
         Create users user100, user200 and group sgrp with GID 9999. add user100 and user200 to
this group.
         Create a directory /sdir with ownership and owning groups belong to root and sgrp, and
set the setgid bit on /sdir
```
# groupadd sgrp -g 9999
# useradd user100 -G sgrp
# useradd user200 -G sgrp
# mkdir /sdir
# chown root:sgrp /sdir
# chmod g+s,g+w /sdir
# cd /sdir
# touch file            => all the files created in dir get the same grp owner
#
```
-  ***Task 7:***<br>
         Create a file under /tmp as user100 and try to delete it as user200. Unset the sticky bit
on /tmp and try to erase the file again. Restore the sticky bit on /tmp
```bash
# su - user100
# touch /tmp/file100
# su - user200
# rm /tmp/file200               => rm: cannot remove 'file': Operation not permitted
# chmod o-t /tmp                => with root priv
# rm /tmp/file
# chmod o+t /tmp
```
#### Automount
-  ***Task 1 :***<br>
        Configure a direct map to automount the NFS share /common that is available from server2.
Install the relevant software, create a local mount point /autodir, and set up AutoFS maps to
support the automatic mounting. Note that /common is already mounted on the /local mount point on
server1 via fstab. Ensure there is no conflict in configuration or functionality between the 2
```bash
# dnf install -y autofs
# mkdir /autodir
# vim /etc/auto.master
        /-       /etc/auto.master.d/auto.dir
~~~

~~~
# vim /etc/auto.master.d/auto.dir
        /autodir        -rw   server1:/common
```
-  ***Task 2:***<br>
        On server1 (NFS server), create a user account called user30 with UID 3000. Add the /home
directory to the list of NFS shares so that it becomes available for remote mount. On server2 (NFS
client), create a user account called user30 with UID 3000, base directory /nfshome, and no user
home directory. establish an indirect map to automount the remote home directory of user30 under
/nfshome. Observe that the home directory of user30 is automounted under /nfshome when you sign in
as user30
```bash
# useradd -u 3000 user30
echo "/home/    client(rw,no_root_squash)" >> /etc/exports
On server2 :
 mkdir /nfshome
 useradd -u 3000 user30 -b /nfshome  -M
 echo "/nfshome /etc/auto.master.d/home" >> /etc/auto.master
 echo "*        -rw            server1:/home/&" >> /etc/auto.master.d/home
 systemctl restart autofs.service
 su - user30
 df -h
```
-  ***Task 3 :***<br>
    Configura Autofs
    - All Ldapuser2 home directory is exported via NFS, which is available on
classroom.example.com (172.25.254.254) and your NFS-exports directory is **/home/guests for
Ldapuser2**,
    - Ldapuser2's home directory is classroom.example.com:/home/guests/ldapuse2
    - Ldapuser2's home directory should be automount autofs service.
    - Home directories must be writable by their users.
   while you are able to log in as any of the user ldapuser1 through ldapuser20, the only home
directory that is accessible from your system is ldapsuser2

```bash
# yum install -y autofs
# vim /etc/auto.master.d/home.autofs
  /home/guests   /etc/auto.home
# vim /etc/auto.home
  * -rw classroom.example.com:/home/guests/&
# systemctl enable --now autofs.service
# ssh ldapuser5@localhost
# cd
# pwd  # /home/guests/ldapuser5
```
#### Layered Storage

-  ***Task 1 :***<br>
         Install the VDO software packages, start the VDO services, and mark it for autostart on
subsequent reboots
        Create a volume called vdo-vol1 of logical size 16GB on the /dev/sdc disk (the actual size
of /dev/sdc is 4GB). List the volume and display its status information. Show the activation
status of the compression and de-duplication features
```
# dnf install vdo kmod-kvdo -y
# systemctl enable --now vdo
# vdo create --name vdo-vol1 --vdoLogicalSize 16G --device /dev/sdc
# vdo list                      => vdo-vol1
# vdo status --name vdo-vol1 | egrep -i "compression|deduplication"
```


-  ***Task 2 :***<br>
         Delete the vdo-vol1 volume that was created above and confirm the removal.
         Create a VDO volume called vdo1 of logical size 16GB on the sdc disk. Initialise the
volume with the XFS file system type, define it for persistence using its device files, create a
mount point called /xfsvdo1, attach it to the directory structure, and verify its availability and
usage

```
# vdo remove --name vdo-vol1 --verbose
~~~

~~~
# vdo list
# wipefs -a /dev/sdc
# vdo create --name vdo1 --vdoLogicalSize 16G --device /dev/sdc
# vdo list
# mkfs.xfs -K /dev/mapper/vdo1
# mkdir /xfsvdo1
# blkid
# echo "UUID=XXXXX /xfsvdo1 xfs defaults,x-systemd.requires=vdo.service,discard 0 0">>/etc/fstab
=> if you forget the mount option : check the man vdo page
# mount -a
# cp -rf /etc/[a-f]* /xfsvdo1/
# vdostats --human-readable vdo1
```


-  ***Task 3 :***<br>
        Install the Stratis software packages, start the Stratis service, and mark it for
autostart on subsequent system reboots
```
# sudo dnf install stratis-cli stratisd
# sudo systemctl enable --now stratis
```
-  ***Task 4 :***<br>
        Create a Stratis pool called strpool and a file system strfs2 by reusing the 1GB sdb disk.
Display information about the pool, file system, and device used. Expand the pool to include
another 1GB disk sdc and confirm
```
# stratis pool create strpool /dev/sdb
# stratis filesystem create strpool strfs2
# stratis pool list
# stratis filesystem list
# stratis blockdev list
# stratis pool add-data strpool /dev/sdc
# stratis blockdev list strpool
```
-  ***Task 5 :***<br>
        Destroy the Stratis file system and the pool that was created, expanded in the above
tasks. Verify the deletion with appropriate commands
```
 Before you destory any fs ,check if the fs is mounted
# stratis filesystem destroy strpool strfs2
# stratis pool destroy strpool
# stratis pool list
```
-  ***Task 6 :***<br>
        Create a new STRATIS volume according to the following requirements:
        - The volume is named  'stratisfs' belongs to 'stratispool'
        - The volume must be mounted permanent under '/stratisvolume'
        - Copy all the files from '/var/log/audit/' and subdirectories to /stratisvolume
        - Take a  snapshot of  stratisfs named  stratissnap.
```
# systemctl status stratisd.service
# stratis pool create stratispool /dev/sdc
# stratis filesystem create stratisfs
# stratis filesystem list
# mkdir /stratisvolume
# stratis filesystem list
# echo "UUID=XXXX /stratisvolume xfs defaults,x-systemd.requires=stratisd.service 0 0" >>
/etc/fstab
# mount -a
# df -h
# cp -R /var/log/audit/ /stratisvolume
# stratis filesystem snapshot stratispool stratisfs stratissnap

```
### Deploy configure and maintain systems

-  ***Task 1 :***<br>
         As Bob, create a once-off job that creates a file called /testresults/Hello.sh containing
the text "Hello World. This is Admin." in 2 days later
~~~

~~~
```
# vim /testresult/Hello.sh
# chmod 755 /Hello.sh
        #!/bin/bash
        echo "Hello World. This is Admin"

# at now +2 days -f /testresult/Hello.sh
```


-  ***Task 2:***<br>
        Set the system time zone and configure the system to use NTP
```
Check if the chrony service is up and running
# systemctl status chronyd.service
# timedatectl list-timezones    => choose the apropriate timezone
# timedatectl set-timezone XX/XX
# timedatectl set-ntp true
```


-  ***Task 3 :***<br>
        Create a periodic job that appends the current date to the file ~/tracking every 5 minutes
every Sunday and Wednesday between the hours of 3am and 4am. Remove the ability of bob to create
cron jobs
```
# touch ~/tracking
# crontab -e
         */5 03-04 * * 0,3 echo $(date) >> ~/tracking
# crontab -l
# echo "bob" >> /etc/cron.deny

```
-  ***Task 4 :***<br>
         Create a daily cron job at 4:27PM for the Derek user that runs cat /etc/redhat-release
and redirects the output to /home/derek/release
```
# crontab -e -u Derek
        27 16 * * *  cat /etc/redhat-release >> /home/derek/release
# crontab -l -u Derek
```
-  ***Task 5 :***<br>
        Submit a job as user100 to run the date command at 11:30pm on March 31, 2022, and have the
output and any error messages generated redirected to /tmp/date.out. List the submitted job and
then remove it
```
# su - user100
# at 11:30pm 2022-03-31
 > date &> /tmp/date.out
# atq           => to remove the job in queue : atrm
```
-  ***Task 6 :***<br>
        Access the repositories that are available on the RHEL 8 image. Create a definition file
for the repositories and confirm
```
# mount /dev/sr0 /mnt/
# vim /etc/yum.repo.d/redhat-local.repo
        [BaseOs]
         name = BaseOs
         baseurl = file:///mnt/BaseOs
         enabled=1
         gpgcheck=0
        [AppStream]
         name = AppStream
         enabled=1
         baseurl = file:///mnt/AppStream
         gpgcheck = 0
# dnf repolist
```
-  ***Task 7 :***<br>
        Verify the integrity and authenticity of a package called dcraw located in the
~~~

/mnt/AppStream/Packages directory on the installation image and then install it. Display basic information about the package, show files it contains, list documentation files, verify the package attributes and remove the package
```
# rpmkey -K /mnt/AppStream/Packages/dcraw-9.27.0-9.el8.x86_64.rpm       => digests signature OK
# rpm -iv /mnt/AppStream/Packages/dcraw-9.27.0-9.el8.x86_64.rpm
# rpm -ql dcraw
# rpm -qd dcraw
# rpm -Vv dcraw
# rpm -ve dcraw
```
-  ***Task 8 :***<br>
        Determine if the cifs-utils package is installed and if it is available for installation. Display its information before installing it. Install the package and display its information again. Remove the package along with its dependencies and confirm the removal
```
# dnf list installed cifs-utils
# rpm -q cifs-utils
# dnf info cifs-utils
        ...
        Repository      : rhel-..

# dnf install cifs-utils -y
# dnf info cifs-utils
        ...
        Repository      : @System
```
-  ***Task 9 :***<br>
        Perform management operations on a package group called system tools. Determine if this group is already installed and if it is available for installation. List the packages it contains and install it. Remove the group along with its dependencies and confirm the removal
```
# dnf group list "System Tools"
# dnf group list installed "System Tools"
# dnf groupinfo "System Tools"
# dnf group install -y "System Tools"
# dnf group remove -y "System Tools" --all
```
-  ***Task 10 :***<br>
        Perform management operations on a module called postgresql. Determine if this module is already installed and if it is available for installation. Show its information and install the default profile for stream 10. Remove the module profile along with any dependencies and confirm its removal
```
# dnf module list postgresql
# dnf module info postgresql
# dnf module install -y postgresql:10   => as show in the output [d]
# postges --version
# dnf -y module romve postrgresql:10
# dnf module list postgresql
```

### Manage basic networking

-  ***Task 1 :***<br>
        Create a connection profile for the new network interface on server using a text editing tool. Assign the IP 172.10.10.110/24 with gateway 172.10.10.1 and set it to autoactivate at system reboots. Deactivate and reactive this interface at the command prompt
```
# vim /etc/sysconfig/network-scripts/ifcfg-eth1
## The config file is a KEY=value pair
TYPE=ethernet
BOOTPROTO=none
ONBOOT=yes
IPADDR=172.10.10.110
GATEWAY=172.10.10.1
PREFIX=24
DEVICE=eth1
```

~~~
# ifup eth1
# nmcli con sh
# ip a s
```
-  ***Task 2 :***<br>
        Create a connection profile using the nmcli command for the new network interface that was
added to server2. Assign the IP 172.10.10.120/24 with gateway 172.10.10.1, and set it to
autoactivate at system reboot. Deactivate and reactivate this interface at the command prompt
```
# nmcli con add type ethernet ifname eth1 ipv4.addresses 172.10.10.120 ipv4.gateway 172.10.10.1
autoconnect yes
# nmcli con down name_con
# nmcli con up name_con
```
-  ***Task 3 :***<br>
        Update the /etc/hosts file on both server1 and server2. Add the IP addresses assigned to
both connections and map them to hostnames . Test connectivity from server to client and from
client to server using their IP addresses and then their hostnames
```
On Server
# vim /etc/hosts
172.10.10.120 server.example.com server
172.10.10.121 client.example.com client

On client :
# vim /etc/hosts
172.10.10.121 client.example.com client
172.10.10.120 server.example.com server
```
-  ***Task 4 :***<br>
        Determine the current active zone. Add and activate a permanent rule to allow HTTP traffic
on port 80, and then add a runtime rule for traffic intended for TCP port 443. Add a permanent
rule to the internal zone for TCP port range 5901 to 5910. Confirm the changes and display the
contents of the affected zone files. Switch the default zone to the internal zone and activate it
```
# firewall-cmd --get-default-zone
# firewall-cmd --add-service=http --permanent           => by default http listen on : 80
# firewall-cmd --add-port=443/tcp
# firewall-cmd --add-port=5901-5910/tcp --zone=internal --permanent
# firewall-cmd --reload
# firewall-cmd --list-all --zone=internal       => zone=public
# vim /etc/firewalld/zones/name_zone.xml        => name_zone=public|internal
# firewall-cmd --set-default-zone=internal
# firewall-cmd --reload
# firewall-cmd --get-active-zones
```
-  ***Task 5 :***<br>
        Remove the 2 permanent rules added above. Switch back to the public zone as the default
zone, and confirm the changes
```
# firewall-cmd --remove-service=http --zone=public --permanent
# firewall-cmd --remove-port=5901-5910 --zone=internal --permanent
# firewall-cmd --set-default-zone=public --permanent
# firewall-cmd --list-all
# firewall-cmd --reload
```


-  ***Task 6 :***<br>
        Set the system hostname to server1.example.com and alias server1. Make sure that the new
hostname is reflected in the command prompt
```
# hostnamectl set-hostname server1.example.com
# systemctl restart systemd-hostnamed.service
# hostnamectl status
# logout
```
### Manage users and groups
-  ***Task 1 :***<br>
        Create three users (Derek, Tom, and Kenny) that belong to the instructors group. Prevent
~~~

```
Tom's user from accessing a shell, and make his account expire 10 day from now
```
```
# groupadd instructors
# useradd Derek -G instructors
# useradd Tom -G instructors -s /sbin/nologin -e +10days
# useradd Kenny -G instructors
```
```


-  ***Task 2 :***<br>
        Add 3 new users alice, bob and charles. Create a marketing group and add these users to
the group. Create a directory /marketing and change the owner to alice and group to marketing. Set
permissions so that members of the marketing group can share documents in the directory but nobody
else can see them. Give charles read-only permission. Create an empty file in the directory
```
```
# useradd alice -G marketing    => same thing as : bob|charles
# mkdir /marketing
# chown alice:marketing /marketing
# chmod 770 /marketing
# setfacl -m u:charles:r-- /marketing
# touch /marketing -l file
```
```
-  ***Task 3 :***<br>
        Create user300 with the default attributes in the useradd and login.defs files. Assign
this user a password and show the line entries from all 4 authentication files
```
```
# useradd usre300
# passwd user300
# grep user300 /etc/passwd /etc/group /etc/shadow /etc/gshadow
```
```
-  ***Task 4 :***<br>
        For user200 change the login name to user200new, UID to 2000, home directory to
/home/user200new, and login shell to /sbin/nologin. Display the line entry for user2new from the
passwd for validation. Remove this user and confirm the deletion
```
```
# usermod user300 -u 2000 -l user300new -s /sbin/nologin -m -d /home/user300new
# grep user300new /etc/passwd
# userdel user300new
!! -m : to move the content of the user's home dir to the new location
```
```
-  ***Task 5 :***<br>
        Configure password ageing for Derek using the chage command . Set the mindays to 7,
maxdays to 28, and warndays to 5. Verify the new settings. Rerun the command and set account
expiry to January 31, 2022
```
```
# chage -m 7 -M 28 -W 5 Derek
# chage -l
# chage -E 2022-01-31
```
```
-  ***Task 6 :***<br>
        Configure password aging for using the PASSWD command. Set the mindays to 10, maxdays to
90, and warndays to 14, and verify the new settings. Set the number of inactivity days to 5 and
ensure that the user is forced to change their password upon next login
```
```
# passwd -n 10 -x 90 -w 14 Derek
# passwd --status
# passwd -i 5 Derek
# passwd -e Derek
```
```
-  ***Task 7 :***<br>

        Create a group called linuxadm with GID 5000 and another group called dba sharing the GID
5000. Add Derek as a secondary member to group linuxadm
```
```
# groupadd -g 5000  linuxadm
# groupadd -g 5000 -o dba
# usermod -aG linuxadm Derek
# id Derek
```
```
-  ***Task 8 :***<br>
```

        Change the linuxadm group name to sysadm and the GID to 6000. Modify the primary group for Derek to sysadm. Remove the sysadm group and confirm
```
# groupmod -g 6000 -n sysadm linuxadm
# usermod -g sysadm Derek
# groupdel sysadm               => cannot remove the primary group
        !! we can add  the option -f, --force: to delete grp even if it is the primary group of a user
# groupdel -f sysadm
```

### Manage security
-  ***Task 1 :***<br>
        Create a file acluser as user100 in /tmp and check if there are any ACL settings on the file. Apply access ACLs on the file for user100 for read and write access. Add user200 to the file for full permissions. Remove all access ACLs from the file
```
as user100
# touch /tmp/acluser
# getfacl /tmp/acluser
# setfacl -m u:user200:rwx /tmp/acluser
# getfacl /tmp/acluser
# setfacl -b /tmp/acluser
```

-  ***Task 2 :***<br>
        Generate a password-less ssh key pair using RSA for user100 on server. Distribute the public key to client and attempt to log on to client from server. Show the log file message for the login attempt
```
as user100
# ssh-keygen
# ssh-copy-id server2.example.com       => or you can user the ip @ of the client
# ssh server2
# vim /var/log/secure
```
-  ***Task 3 :***<br>
        Remove the sshd service rule from the runtime configuration on server and try to access the server from the client using the ssh command
```
 On the server
# firewall-cmd --remove-service=ssh
# firewall-cmd --reload
# ssh server1
no route to host
```
-  ***Task 4:***<br>
        Create a directory sedir1 under /tmp and a file sefile1 under sedir1. Check the context on the directory and file. Change the SELinux user and type to user_u and public_content_t on both and verify
```
# mkdir /tmp/sedir1 && cd $_            => the variable $_ in bash is the last arg given to the previous cmd
# touch sefile1
# ls -lZd       => to check the context on the dir sedir1
# ls -lZ sefile1
# chcon -u user_u  -t public_content_t -R ./
# ls -lZ sefile1
```
-  ***Task 5 :***<br>
        Add the current context on sedir1 to the SELinux policy database to ensure a relabeling will not reset it to its previous value. Next, you will change the context on the directory to some random values. Restore the default context from the policy database back to the directory recursively
```
# semanage fcontext -a -t public_content_t -s user_u "/tmp/sedir1(/.*)?"
To see the policy that been added to the database recently :
# cat /etc/selinux/targeted/files/file_contexts.local
# restorecon -Rv sedir1                 => to apply the policy to the file sys
```

~~~~
```
        tip: chcon write the context to the file sys and not to the policy so everything is
overwritten when
                 the fs is relabeled where the original context is restored from the policy to the
file sys
                 so it's recommended to work with semanage
-  ***Task 6 :***<br>
        Add a non-standard port 8010 to the SELinux policy database for the httpd service and
confirm the addition. Remove the port from the policy and verify the deletion
```
# semanage port -a -t http_port_t -p tcp 8010
# semanage port -l | grep http
# semanage port -d -t http_port_t -p tcp 8010   => d : for delete
# semanage port -l | grep http
```
-  ***Task 7 :***<br>
        Create a file called sefile2 under /tmp and display its context. Copy this file to the
/etc/default directory, and observe the change in the context. Remove sefile2 from /etc/default,
and copy it again to the same destination, ensuring that the target file receives the source
file's context
```
# touch /tmp/sefile2
# ls -lZ sefile2
# cp /tmp/sefile2 /etc/default
# ls -lZ /etc/default/sefile2   => the context is :etc_t
# rm /etc/default/sefile2
# cp /tmp/sefile2 /etc/default/sefile2 ---preserve=context  => or -a to preserve all
# ls -lZ sefile2
```
-  ***Task 8 :***<br>
        Display the current state of the Boolean nfs_export_all_rw. Toggle its value temporarily,
and reboot the system. Flip its value persistently after the system has been back up
```
# semanage boolean -l | grep nfs_export_all_rw or # getsebool -a | grep nfs...
# setsebool nfs_export_all_rw off
# reboot
# setsebool -P nfs_export_all off
```


### Manage containers
-  ***Task 1 :***<br>
        Download the Apache web server container image (httpd 2.4) and inspect the container
image. Check the exposed ports in the container image configuration
```
# podman login registry.redhat.io
# podman search registry.redhat.io/httpd-24             => you can use docker registry
# podman pull registry.redhat.io/rhel8/httpd-24
# podman inspect httpd-24 | vim -                       => search the exposed port: ExposedPorts -
8080,8443/tcp ..
```
        UBI images "stands for universal base image" and it is used as the foundation for all of
the redhat product : registry.redhat.io/ubi8/httpd-24
        vim - : this is vim func that reads the STDIN
        The puprose of skopeo: you can inspect the image without the need of pulling the image
from the registry
-  ***Task 2 :***<br>
        Run the httpd container in the background. Assign the name myweb to the container, verify
that the container is running, stop the container and verify that it has stopped, and delete the
container and the container image
```
# podman run -d --name myweb httpd-24
# podman ps
# podman stop myweb
# podman ps -a                  => Check the status of the container
# podman rm -f myweb    => as the container in paused state we can use -f option
# podman rmi httpd-24
```
-  ***Task 3 :***<br>
        Pull the Apache web server container image (httpd 2.4) and run the container with the name
~~~~

~~~~text
webserver. Configure webserver to display content "Welcome to container-based web server". Use
port 3333 on the host machine to receive http requests. Start a bash shell in the container to
verify the configuration
```
- There are different methods to configure the web server
        - using the bind-mount : mount a dir (~/hostfiles/) that host the files to /var/www/html
        - create a containerfile to build a container image
        - create a new image based on the changed container


# podman pull registry.redhat.io/rhel8/httpd-24
        --- using the containerfile ---
        # echo "Welcome to container-based web server " > index.html
        # vim Dockerfile
                FROM httpd-24
                MAINTAINER      First_name
                COPY index.html /var/www/html/
        # podman build . -t httpd:v1
# podman run -dit --name webserver -p 3333:8080 httpd:v1
# firewall-cmd --add-port=3333/tcp --permanent
# firewall-cmd --reload
# curl localhost:3333                   => Welcome to container-based web serve
# podman exec -it webserver bash
# cat /var/www/html/index.html
# exit
```
-  ***Task 4 :***<br>
        Configure the system to start the webserver container at boot as a systemd service.
Start/enable the systemd service to make sure the container will start at boot, and reboot the
system to verify if the container is running as expected
```
- With root priv :
# podman generate systemd webserver > httpd-server.service
# systemctl daemon-reload
# systemctl enable --now cont-httpd-server.service
# reboot
# systemctl status httpd-container.service
# curl localhost:3333
- With rootless container :
# mkdir -p ~/.config/systemd/user && cd $_
# podman generate systemd webserver > httpd-srv.service

# loginctl enable-ligner user-name
# systemctl --user daemon-reload
# vim httpd-srv.service
        # Change WantedBy to default.target
        WantedBy=default.target
# systemctl --user status httpd-srv.service
# systemctl --user enable --now httpd-srv.service
By default this service will start when the user login and stop when user logged out, to change
this behaviour you've to enable user linger like we done above
# reboot
# podman ps
```
&nbsp;&nbsp;if you face this error: Failed to connect to bus [check this link]
(https://access.redhat.com/discussions/6029491)



-  ***Task 5 :***<br>
        Create a container logserver from an image rsyslog in server1 from registry.redhat.io<br>
        - Configure the container with systemd services by an existing user user10.<br>
        - Service name should be container-logserver, and configure it to start automatically
across reboot.<br>
        - Configure your host journal to store all journal across reboot, copy all *.journal from
/var/log/journal and all subdirectories to /home/user10/container_logserver <br>
        - Configure automount /var/log/journal from  logserver (container) to
/home/user10/container_logserver when container starts.
```bash
# podman search registry.redhat.io/rsyslog
~~~~

~~~~
# podman login registry.redhat.io
  Username:
  Password:
# podman pull registry.redhat.io/rhel8/rsyslog
        --- root priv ---
# vim /etc/systemd/journal.conf
        Storage=persistent
# systemctl restart systemd-journald
# ls /var/log/journal
# mkdir /home/user10/container_logserver
# cp -R /var/log/journal/ /home/user10/container_logserver
# chown -R user10:user10 ~/container_logserver
# su - user10
# podman run -d --name logserver -v ~/container_logserver/:/var/log/:Z  rsyslog
# podman ps
# mkdir -p ~/.config/systemd/user/
# cd $_
# podman generate systemd --name logserver --files --new
# podman exec -it logserverlogserver
# vim container-logserver.service
        WantedBy=default.target
# loginctl enable-linger $USER
# systemctl --user daemon-reload
# systemctl --user enable container-logserver.service
# systemctl --user status container-logserver
# reboot                => to make sure that the container is up and running

```

### resources

  - RHCSA Red Hat Enterprise Linux 8 (UPDATED): Training and Exam Preparation Guide (EX200),
Second Edition

  - Red Hat RHCSA 8 Cert Guide: EX200 (Certification Guide)
      by: Sander van Vugt

  - [Github repo](https://github.com/jrandj/redhat/blob/master/README.md#Exercises)
~~~~