RHL LAB ALL SOLUTIONS

## Lab-1: Improving Command-line Productivity

1. Create the /home/student/bin/bash-lab script file on workstation.
   1. On workstation, create the /home/student/bin/ folder if needed.

**[student@workstation ~]$ mkdir -p /home/student/bin**

* 1. Use **vim** to create and edit the /home/student/bin/bash-lab script file.

**[student@workstation ~]$ vim ~/bin/bash-lab**

* 1. Insert the following text and save the file.

#!/bin/bash

* 1. Make your script file executable.

**[student@workstation ~]$ chmod a+x ~/bin/bash-lab**

You can use **sudo** without requiring a password on the servera and serverb hosts. Remember to use a loop to simplify your script. You can also use multiple **grep** commands concatenated with the use of the pipe character (|).

1. Use **vim** to open and edit the /home/student/bin/bash-lab script file.

**[student@workstation ~]$ vim ~/bin/bash-lab**

1. Append the following lines in bold to the /home/student/bin/bash-lab script file.

### Note

The following is an example of how you can achieve the requested script. In Bash scripting, you can take different approaches and obtain the same result.

#!/bin/bash

**#**

**USR='student'**

**OUT='/home/student/output'**

**#**

**for SRV in servera serverb**

**do**

**ssh ${USR}@${SRV} "hostname -f" > ${OUT}-${SRV}**

**echo "#####" >> ${OUT}-${SRV}**

**ssh ${USR}@${SRV} "lscpu | grep '^CPU'" >> ${OUT}-${SRV}**

**echo "#####" >> ${OUT}-${SRV}**

**ssh ${USR}@${SRV} "grep -v '^$' /etc/selinux/config|grep -v '^#'" >> ${OUT}-${SRV}**

**echo "#####" >> ${OUT}-${SRV}**

**ssh ${USR}@${SRV} "sudo grep 'Failed password' /var/log/secure" >> ${OUT}-${SRV}**

**echo "#####" >> ${OUT}-${SRV}**

**done**

1. Execute the **/home/student/bin/bash-lab** script, and review the output content on workstation.
   1. On workstation, execute the **/home/student/bin/bash-lab** script.

**[student@workstation ~]$ bash-lab**

* 1. Review the content of /home/student/output-servera and /home/student/output-serverb.
  2. **[student@workstation ~]$ cat /home/student/output-servera**
  3. servera.lab.example.com
  4. #####
  5. CPU op-mode(s): 32-bit, 64-bit
  6. CPU(s): 2
  7. CPU family: 21
  8. CPU MHz: 2294.670
  9. #####
  10. SELINUX=enforcing
  11. SELINUXTYPE=targeted
  12. #####
  13. Mar 21 22:30:28 servera sshd[3939]: Failed password for invalid user operator1 from 172.25.250.9 port 58382 ssh2
  14. Mar 21 22:30:31 servera sshd[3951]: Failed password for invalid user sysadmin1 from 172.25.250.9 port 58384 ssh2
  15. Mar 21 22:30:34 servera sshd[3953]: Failed password for invalid user manager1 from 172.25.250.9 port 58386 ssh2

#####

**[student@workstation ~]$ cat /home/student/output-serverb**

serverb.lab.example.com

#####

CPU op-mode(s): 32-bit, 64-bit

CPU(s): 2

CPU family: 6

CPU MHz: 2294.664

#####

SELINUX=enforcing

SELINUXTYPE=targeted

#####

Mar 21 22:30:37 serverb sshd[3883]: Failed password for invalid user operator1 from 172.25.250.9 port 39008 ssh2

Mar 21 22:30:39 serverb sshd[3891]: Failed password for invalid user sysadmin1 from 172.25.250.9 port 39010 ssh2

Mar 21 22:30:43 serverb sshd[3893]: Failed password for invalid user manager1 from 172.25.250.9 port 39012 ssh2

#####

## Lab2: Tuning System Performance

1. Change the current tuning profile for serverb to balanced, a general non-specialized tuned profile.
   1. From workstation, open an SSH session to serverb as student user. The systems are configured to use SSH keys for authentication, so a password is not required.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*
   4. **[student@serverb ~]$**
   5. Use **yum** to confirm that the tuned package is installed.
   6. **[student@serverb ~]$ yum list tuned**
   7. *...output omitted...*
   8. Installed Packages
   9. **tuned.noarch** 2.10.0-15.el8 @anaconda
   10. Use the **systemctl is-active tuned** command to display the tuned service state.
   11. **[student@serverb ~]$ systemctl is-active tuned**
   12. **active**
   13. List all available tuning profiles and their descriptions. Note that the current active profile is virtual-guest.
   14. **[student@serverb ~]$ sudo tuned-adm list**
   15. **[sudo] password for student: student**
   16. Available profiles:
   17. - balanced - General non-specialized tuned profile
   18. - desktop - Optimize for the desktop use-case
   19. - latency-performance - Optimize for deterministic performance at the cost of
   20. increased power consumption
   21. - network-latency - Optimize for deterministic performance at the cost of
   22. increased power consumption, focused on low latency
   23. network performance
   24. - network-throughput - Optimize for streaming network throughput, generally
   25. only necessary on older CPUs or 40G+ networks
   26. - powersave - Optimize for low power consumption
   27. - throughput-performance - Broadly applicable tuning that provides excellent
   28. performance across a variety of common server workloads
   29. - virtual-guest - Optimize for running inside a virtual guest
   30. - virtual-host - Optimize for running KVM guests
   31. Current active profile: **virtual-guest**
   32. Change the current active tuning profile to the balanced profile.

**[student@serverb ~]$ sudo tuned-adm profile balanced**

* 1. List summary information of the current active tuned profile.

Use the **tuned-adm profile\_info** command to confirm that the active profile is the balanced profile.

**[student@serverb ~]$ sudo tuned-adm profile\_info**

Profile name:

**balanced**

Profile summary:

**General non-specialized tuned profile**

*...output omitted...*

1. Two processes on serverb are consuming a high percentage of CPU usage. Adjust each process's nice level to 10 to allow more CPU time for other processes.
   1. Determine the top two CPU consumers on serverb. The top CPU consumers are listed last in the command output. CPU percentage values will vary.
   2. **[student@serverb ~]$ ps aux --sort=pcpu**
   3. USER PID **%CPU** %MEM VSZ RSS TTY STAT START TIME COMMAND
   4. *...output omitted...*
   5. root 2983 **100** 0.0 228360 1744 ? R< 21:08 0:23 **md5sum /dev/zero**
   6. root 2967 **101** 0.0 228360 1732 ? RN 21:08 0:23 **sha1sum /dev/zero**
   7. [student@serverb ~]$
   8. Identify the current nice level for each of the top two CPU consumers.
   9. **[student@serverb ~]$ ps -o pid,pcpu,nice,comm \**
   10. **$(pgrep sha1sum;pgrep md5sum)**
   11. PID %CPU **NI** COMMAND
   12. 2967 99.6 **2** sha1sum
   13. 2983 99.7 **-2** md5sum
   14. Use the **sudo renice -n 10 *2967* *2983*** command to adjust the nice level for each process to 10. Use PID values identified in the previous command output.
   15. **[student@serverb ~]$ sudo renice -n 10 *2967* *2983***
   16. **[sudo] password for student: student**
   17. 2967 (process ID) old priority 2, new priority 10
   18. 2983 (process ID) old priority -2, new priority 10
   19. Verify that the current nice level for each process is 10.
   20. **[student@serverb ~]$ ps -o pid,pcpu,nice,comm \**
   21. **$(pgrep sha1sum;pgrep md5sum)**
   22. PID %CPU **NI** COMMAND
   23. 2967 99.6 **10** sha1sum
   24. 2983 99.7 **10** md5sum
   25. Exit from serverb.
   26. **[student@serverb ~]$ exit**
   27. logout
   28. Connection to serverb closed.

[student@workstation ~]$

## Lab3: Controlling Access to Files with ACLs

1. The cases directory and its contents should belong to the managers group. New files added to the cases directory should automatically belong to the managers group. The user and group owners for the existing files should have read and write permission, and other users should have no permission at all.

### Note

Hint: Do not use **setfacl**.

* 1. Log in to serverb as the student user.
  2. **[student@workstation ~]$ ssh student@serverb**
  3. *...output omitted...*

**[student@serverb ~]$**

* 1. Use the **sudo -i** command to switch to the root user. The password for the student user is student.
  2. **[student@serverb ~]$ sudo -i**
  3. [sudo] password for student: **student**

**[root@serverb ~]# Flab**

* 1. Use the **chgrp** command to recursively update group ownership on the directory and its contents.

**[root@serverb ~]# chgrp -R managers /shares/cases**

* 1. Use the **chmod** command to update the set-GID flag on the directory.

**[root@serverb ~]# chmod g+s /shares/cases**

* 1. Use **chmod** to update all existing file permissions to rw for owner and group.

**[root@serverb ~]# chmod 660 /shares/cases/\***

1. Add ACL entries to the cases directory (and its contents) that allow members of the contractors group to have read/write access on the files and execute permission on the directory. Restrict the contractor3 user to read access on the files and execute permission on the directory.
   1. Use **setfacl** to recursively update the existing cases directory and its contents. Grant the contractors group read, write, and conditional execute permissions.

**[root@serverb ~]# setfacl -Rm g:contractors:rwX /shares/cases**

* 1. Use **setfacl** to recursively update the existing cases directory and its contents. Grant the contractor3 user read and conditional execute permissions.

**[root@serverb ~]# setfacl -Rm u:contractor3:rX /shares/cases**

1. Add ACL entries that ensure any new files or directories in the cases directory have the correct permissions applied for *all* authorized users and groups.
   1. Use **setfacl** to update the *default* permissions for members of the contractors group. Default permissions are read, write, and execute (needed for proper subdirectory creation and access).

**[root@serverb ~]# setfacl -m d:g:contractors:rwx /shares/cases**

* 1. Use **setfacl** to update the *default* permissions for the contractor3 user. Default permissions are read and execute (needed for proper subdirectory access).

**[root@serverb ~]# setfacl -m d:u:contractor3:rx /shares/cases**

1. As the root user, use **ls** to check the cases directory and its content. Look for group ownership, directory and file permissions. The "s" in the group file permissions indicates the set-GID flag is set, and the "+" indicates that ACL entries exist. At the end exit from the root user session.
2. **[root@serverb ~]# ls -ld /shares/cases**
3. drwxrw**s**---**+** 2 root managers 46 Mar 29 00:40 /shares/cases
4. **[root@serverb ~]# ls -l /shares/cases**
5. total 8
6. -rw-rw----+ 1 root managers 44 Mar 29 00:33 backlog.txt

-rw-rw----+ 1 root managers 46 Mar 29 00:33 shortlist.txt

1. Use **getfacl** and review its output. Look for the named user and named group entries in both the standard and default ACL.
2. **[root@serverb ~]# getfacl /shares/cases**
3. # file: shares/cases
4. # owner: root
5. # group: managers
6. # flags: -s-
7. user::rwx
8. user:contractor3:r-x
9. group::rwx
10. group:contractors:rwx
11. mask::rwx
12. other::---
13. default:user::rwx
14. default:user:contractor3:r-x
15. default:group::rwx
16. default:group:contractors:rwx
17. default:mask::rwx
18. default:other::---
19. **[root@serverb ~]# exit**

logout

1. Switch to the manager1 user and perform the following operations. Check that you get the expected access behavior.
2. **[student@serverb ~]$ su - manager1**
3. Password: **redhat**
4. **[manager1@serverb ~]$ cd /shares/cases**
5. **[manager1@serverb cases]$ echo hello > manager1.txt**
6. **[manager1@serverb cases]$ cat shortlist.txt**
7. ###Shortlist of Clients to call###TEMPLATE###
8. **[manager1@serverb cases]$ mkdir manager1.dir**
9. **[manager1@serverb cases]$ echo hello > manager1.dir/test.txt**
10. **[manager1@serverb cases]$ ls -ld manager1.dir**
11. drwxrws---+ 2 manager1 managers 22 Mar 29 00:59 manager1.dir
12. **[manager1@serverb cases]$ ls -l manager1.dir**
13. total 4
14. -rw-rw----+ 1 manager1 managers 6 Mar 29 00:59 test.txt
15. **[manager1@serverb cases]$ getfacl manager1.dir**
16. # file: manager1.dir/
17. # owner: manager1
18. # group: managers
19. # flags: -s-
20. user::rwx
21. user:contractor3:r-x
22. group::rwx
23. group:contractors:rwx
24. mask::rwx
25. other::---
26. default:user::rwx
27. default:user:contractor3:r-x
28. default:group::rwx
29. default:group:contractors:rwx
30. default:mask::rwx
31. default:other::---
32. **[manager1@serverb cases]$ exit**

logout

1. Switch to the contractor1 user and perform the following operations. Check that you get the expected access behavior.
2. **[student@serverb ~]$ su - contractor1**
3. Password: **redhat**
4. **[contractor1@serverb ~]$ cd /shares/cases**
5. **[contractor1@serverb cases]$ echo hello > manager1.txt**
6. **[contractor1@serverb cases]$ cat shortlist.txt**
7. ###Shortlist of Clients to call###TEMPLATE###
8. **[contractor1@serverb cases]$ mkdir contractor1.dir**
9. **[contractor1@serverb cases]$ echo hello > contractor1.dir/test.txt**
10. **[contractor1@serverb cases]$ ls -ld contractor1.dir**
11. drwxrws---+ 2 contractor1 managers 22 Mar 29 01:05 contractor1.dir
12. **[contractor1@serverb cases]$ ls -l contractor1.dir**
13. total 4
14. -rw-rw----+ 1 contractor1 managers 6 Mar 29 01:07 test.txt
15. **[manager1@serverb cases]$ getfacl contractor1.dir**
16. # file: contractor1.dir/
17. # owner: contractor1
18. # group: managers
19. # flags: -s-
20. user::rwx
21. user:contractor3:r-x
22. group::rwx
23. group:contractors:rwx
24. mask::rwx
25. other::---
26. default:user::rwx
27. default:user:contractor3:r-x
28. default:group::rwx
29. default:group:contractors:rwx
30. default:mask::rwx
31. default:other::---
32. **[contractor1@serverb cases]$ exit**

logout

1. Switch to the contractor3 user, and perform the following operations. Check that you get the expected access behavior.
2. **[student@serverb ~]# su - contractor3**
3. Password: **redhat**
4. **[contractor3@serverb ~]# cd /shares/cases**
5. **[contractor3@serverb cases]# echo hello > contractor3.txt**
6. -bash: contractor3.txt: Permission denied
7. **[contractor3@serverb cases]# cat shortlist.txt**
8. ###Shortlist of Clients to call###TEMPLATE###
9. **[contractor3@serverb cases]# mkdir contractor3.dir**
10. mkdir: cannot create directory ‘contractor3.dir’: Permission denied
11. **[contractor3@serverb cases]# cat manager1.dir/test.txt**
12. hello
13. **[contractor3@serverb cases]# cat contractor1.dir/test.txt**
14. hello
15. **[contractor3@serverb cases]# exit**
16. logout

**[student@serverb ~]#**

1. Log off from serverb
2. **[student@serverb ~]# exit**
3. logout
4. Connection to serverb closed.

**[student@workstation ~]$**

## Lab4: Managing SELinux Security

1. Log in to serverb as the root user.
   1. Use the ssh command to log in to serverb as the student user. The systems are configured to use SSH keys for authentication, so a password is not required.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*

[student@serverb ~]$

* 1. Use the sudo -i command to switch to the root user. The password for the student user is student.
  2. **[student@serverb ~]$ sudo -i**
  3. [sudo] password for student: **student**

[root@serverb ~]#

1. Launch a web browser on workstation and browse to http://serverb/lab.html. You will see the error message: You do not have permission to access /lab.html on this server.
2. Research and identify the SELinux issue that is preventing Apache from serving web content.
   1. Using the less command, view the contents of /var/log/messages. Use the **/** key and search for sealert. Use the **q** key to quit the less command.
   2. **[root@serverb ~]# less /var/log/messages**
   3. Mar 28 10:19:51 **serverb setroubleshoot**[27387]: **SELinux is preventing /usr/sbin/httpd from getattr access on the file /lab-content/lab.html**. For complete SELinux messages run: **sealert -l 8824e73d-3ab0-4caf-8258-86e8792fee2d**

Mar 28 10:19:51 serverb platform-python[27387]: SELinux is preventing /usr/sbin/httpd from getattr access on the file /lab-content/lab.html.#012#012\*\*\*\*\* Plugin catchall (100. confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*#012#012If you believe that httpd should be allowed getattr access on the lab.html file by default.#012Then you should report this as a bug.#012You can generate a local policy module to allow this access.#012Do#012allow this access for now by executing:#012# ausearch -c 'httpd' --raw | audit2allow -M my-httpd#012# semodule -X 300 -i my-httpd.pp#012

* 1. Run the suggested sealert command. Note the source context, the target objects, the policy, and the enforcing mode.
  2. **[root@serverb ~]# sealert -l 8824e73d-3ab0-4caf-8258-86e8792fee2d**
  3. **SELinux is preventing /usr/sbin/httpd from getattr access on the file /lab-content/lab.html.**
  4. \*\*\*\*\* Plugin catchall (100. confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  5. If you believe that httpd should be allowed getattr access on the lab.html file by default.
  6. Then you should report this as a bug.
  7. You can generate a local policy module to allow this access.
  8. Do
  9. allow this access for now by executing:
  10. # ausearch -c 'httpd' --raw | audit2allow -M my-httpd
  11. # semodule -X 300 -i my-httpd.pp
  12. Additional Information:
  13. Source Context **system\_u:system\_r:httpd\_t:s0**
  14. Target Context **unconfined\_u:object\_r:default\_t:s0**
  15. Target Objects **/lab-content/lab.html [ file ]**
  16. Source httpd
  17. Source Path /usr/sbin/httpd
  18. Port <Unknown>
  19. Host **serverb.lab.example.com**
  20. Source RPM Packages
  21. Target RPM Packages
  22. Policy RPM selinux-policy-3.14.1-59.el8.noarch
  23. Selinux Enabled True
  24. Policy Type targeted
  25. Enforcing Mode **Enforcing**
  26. Host Name serverb.lab.example.com
  27. Platform Linux serverb.lab.example.com
  28. 4.18.0-67.el8.x86\_64 #1 SMP Sat Feb 9 12:44:00
  29. UTC 2019 x86\_64 x86\_64
  30. Alert Count 2
  31. First Seen 2019-03-28 15:19:46 CET
  32. Last Seen 2019-03-28 15:19:46 CET
  33. Local ID 8824e73d-3ab0-4caf-8258-86e8792fee2d
  34. Raw Audit Messages
  35. type=AVC msg=audit(1553782786.213:864): avc: denied { getattr } for pid=15606 comm="httpd" path="/lab-content/lab.html" dev="vda1" ino=8763212 scontext=system\_u:system\_r:httpd\_t:s0 tcontext=unconfined\_u:object\_r:default\_t:s0 tclass=file permissive=0

Hash: httpd,httpd\_t,default\_t,file,getattr

* 1. The Raw Audit Messages section of the sealert command contains information from the /var/log/audit/audit.log. Use the ausearch command to search the /var/log/audit/audit.log file. The -m option searches on the message type. The ts option searches based on time. This entry identifies the relevant process and file causing the alert. The process is the httpd Apache web server, the file is /lab-content/lab.html, and the context is system\_r:httpd\_t.
  2. **[root@serverb ~]# ausearch -m AVC -ts recent**
  3. time->Thu Mar 28 15:19:46 2019
  4. type=PROCTITLE msg=audit(1553782786.213:864): proctitle=2F7573722F7362696E2F6874747064002D44464F524547524F554E44
  5. type=SYSCALL msg=audit(1553782786.213:864): arch=c000003e syscall=6 success=no exit=-13 a0=7fb900004930 a1=7fb92dfca8e0 a2=7fb92dfca8e0 a3=1 items=0 ppid=15491 pid=15606 auid=4294967295 uid=48 gid=48 euid=48 suid=48 fsuid=48 egid=48 sgid=48 fsgid=48 tty=(none) ses=4294967295 comm="httpd" exe="/usr/sbin/httpd" subj=system\_u:system\_r:httpd\_t:s0 key=(null)

**type=AVC** msg=audit(1553782786.213:864): avc: **denied** { getattr } for pid=15606 **comm="httpd"** **path="/lab-content/lab.html"** dev="vda1" ino=8763212 scontext=system\_u:**system\_r:httpd\_t:s0** tcontext=unconfined\_u:object\_r:default\_t:s0 tclass=file permissive=0

1. Display the SELinux context of the new HTTP document root and the original HTTP document root. Resolve the SELinux issue preventing Apache from serving web content.
   1. Use the ls -dZ to compare the document root of /lab-content and /var/www/html.
   2. **[root@serverb ~]# ls -dZ /lab-content /var/www/html**

unconfined\_u:object\_r:**default\_t**:s0 **/lab-content/** system\_u:object\_r:**httpd\_sys\_content\_t**:s0 **/var/www/html/**

* 1. Create a file context rule that sets the default type to httpd\_sys\_content\_ for /lab-content and all the files below it.
  2. **[root@serverb ~]# semanage fcontext -a \**

**-t httpd\_sys\_content\_t '/lab-content(/.\*)?'**

* 1. Use the restorecon command to set the SELinux context for the files in /lab-content.

**[root@serverb ~]# restorecon -R /lab-content/**

erify that the SELinux issue has been resolved and Apache is able to serve web content.

Use your web browser to refresh the http://serverb/lab.html link. Now you should see some web content.

This is the html file for the SELinux final lab on SERVERB.

1. Exit from serverb.
2. **[root@serverb ~]# exit**
3. logout
4. **[student@serverb ~]$ exit**
5. logout
6. Connection to serverb closed.

[student@workstation ~]$

## Lab5: Managing Basic Storage

1. New disks are available on serverb. On the first new disk, create a 2 GB GPT partition named backup. Because it may be difficult to set the exact size, a size between 1.8 GB and 2.2 GB is acceptable. Set the correct file-system type on that partition to host an XFS file system.

The password for the student user account on serverb is student. This user has full root access through **sudo**.

* 1. Use the **ssh** command to log in to serverb as the student user. The systems are configured to use SSH keys for authentication, therefore a password is not required.
  2. **[student@workstation ~]$ ssh student@serverb**
  3. *...output omitted...*

**[student@serverb ~]$**

* 1. Because creating partitions and file systems requires root access, use the **sudo -i** command to switch to the root user. If prompted, use student as the password.
  2. **[student@serverb ~]$ sudo -i**
  3. [sudo] password for student: **student**

**[root@serverb ~]#**

* 1. Use the **lsblk** command to identify the new disks. Those disks should not have any partitions yet.
  2. **[root@serverb ~]# lsblk**
  3. NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
  4. sr0 11:0 1 1024M 0 rom
  5. vda 252:0 0 10G 0 disk
  6. └─vda1 252:1 0 10G 0 part /
  7. **vdb** 252:16 0 5G 0 disk
  8. vdc 252:32 0 5G 0 disk

vdd 252:48 0 5G 0 disk

Notice that the first new disk, vdb, does not have any partitions.

* 1. Confirm that the disk has no label.
  2. **[root@serverb ~]# parted /dev/vdb print**
  3. Error: /dev/vdb: **unrecognised disk label**
  4. Model: Virtio Block Device (virtblk)
  5. Disk /dev/vdb: 5369MB
  6. Sector size (logical/physical): 512B/512B
  7. **Partition Table: unknown**

Disk Flags:

* 1. Use **parted** and the **mklabel** subcommand to define the GPT partitioning scheme.
  2. **[root@serverb ~]# parted /dev/vdb mklabel gpt**

Information: You may need to update /etc/fstab.

* 1. Create the 2 GB partition. Name it backup and set its type to xfs. Start the partition at sector 2048.
  2. **[root@serverb ~]# parted /dev/vdb mkpart backup xfs 2048s 2GB**

Information: You may need to update /etc/fstab.

* 1. Confirm the correct creation of the new partition.
  2. **[root@serverb ~]# parted /dev/vdb print**
  3. Model: Virtio Block Device (virtblk)
  4. Disk /dev/vdb: 5369MB
  5. Sector size (logical/physical): 512B/512B
  6. Partition Table: gpt
  7. Disk Flags:
  8. Number Start End Size File system Name Flags

**1 1049kB 2000MB 1999MB backup**

* 1. Run the **udevadm settle** command. This command waits for the system to detect the new partition and to create the /dev/vdb1 device file. It only returns when it is done.
  2. **[root@serverb ~]# udevadm settle**

**[root@serverb ~]#**

1. Format the 2 GB partition with an XFS file system and persistently mount it at /backup.
   1. Use the **mkfs.xfs** command to format the /dev/vbd1 partition.
   2. **[root@serverb ~]# mkfs.xfs /dev/vdb1**
   3. meta-data=/dev/vdb1 isize=512 agcount=4, agsize=121984 blks
   4. = sectsz=512 attr=2, projid32bit=1
   5. = crc=1 finobt=1, sparse=1, rmapbt=0
   6. = reflink=1
   7. data = bsize=4096 blocks=487936, imaxpct=25
   8. = sunit=0 swidth=0 blks
   9. naming =version 2 bsize=4096 ascii-ci=0, ftype=1
   10. log =internal log bsize=4096 blocks=2560, version=2
   11. = sectsz=512 sunit=0 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

* 1. Create the /backup mount point.
  2. **[root@serverb ~]# mkdir /backup**

**[root@serverb ~]#**

* 1. Before adding the new file system to /etc/fstab, retrieve its UUID.
  2. **[root@serverb ~]# lsblk --fs /dev/vdb1**
  3. NAME FSTYPE LABEL UUID MOUNTPOINT

vdb1 xfs **a3665c6b-4bfb-49b6-a528-74e268b058dd**

The UUID on your system is probably different.

* 1. Edit /etc/fstab and define the new file system.
  2. **[root@serverb ~]# vim /etc/fstab**
  3. *...output omitted...*

**UUID=*a3665c6b-4bfb-49b6-a528-74e268b058dd* /backup xfs defaults 0 0**

* 1. Force systemd to reread the /etc/fstab file.
  2. **[root@serverb ~]# systemctl daemon-reload**

**[root@serverb ~]#**

* 1. Manually mount /backup to verify your work. Confirm that the mount is successful.
  2. **[root@serverb ~]# mount /backup**
  3. **[root@serverb ~]# mount | grep /backup**

**/dev/vdb1 on /backup** type xfs (rw,relatime,seclabel,attr2,inode64,noquota)

1. On the same new disk, create two 512 MB GPT partitions named swap1 and swap2. A size between 460 MB and 564 MB is acceptable. Set the correct file-system type on those partitions to host swap spaces.
   1. Retrieve the end position of the first partition by displaying the current partition table on /dev/vdb. In the next step, you use that value as the start of the swap1 partition.
   2. **[root@serverb ~]# parted /dev/vdb print**
   3. Model: Virtio Block Device (virtblk)
   4. Disk /dev/vdb: 5369MB
   5. Sector size (logical/physical): 512B/512B
   6. Partition Table: gpt
   7. Disk Flags:
   8. Number Start End Size File system Name Flags

1 1049kB **2000MB** 1999MB xfs backup

* 1. Create the first 512 MB partition named swap1. Set its type to linux-swap. Use the end position of the first partition as the starting point. The end position is 2000 MB + 512 MB = 2512 MB
  2. **[root@serverb ~]# parted /dev/vdb mkpart swap1 linux-swap 2000MB 2512M**

Information: You may need to update /etc/fstab.

* 1. Create the second 512 MB partition named swap2. Set its type to linux-swap. Use the end position of the previous partition as the starting point: 2512M. The end position is 2512 MB + 512 MB = 3024 MB
  2. **[root@serverb ~]# parted /dev/vdb mkpart swap2 linux-swap 2512M 3024M**

Information: You may need to update /etc/fstab.

* 1. Display the partition table to verify your work.
  2. **[root@serverb ~]# parted /dev/vdb print**
  3. Model: Virtio Block Device (virtblk)
  4. Disk /dev/vdb: 5369MB
  5. Sector size (logical/physical): 512B/512B
  6. Partition Table: gpt
  7. Disk Flags:
  8. Number Start End Size File system Name Flags
  9. 1 1049kB 2000MB 1999MB xfs backup
  10. **2 2000MB 2512MB 513MB swap1 swap**

**3 2512MB 3024MB 512MB swap2 swap**

* 1. Run the **udevadm settle** command. This command waits for the system to register the new partitions and to create the device files.
  2. **[root@serverb ~]# udevadm settle**

**[root@serverb ~]#**

1. Initialize the two 512 MiB partitions as swap spaces and configure them to activate at boot. Set the swap space on the swap2 partition to be preferred over the other.
   1. Use the **mkswap** command to initialize the swap partitions.
   2. **[root@serverb ~]# mkswap /dev/vdb2**
   3. Setting up swapspace version 1, size = 489 MiB (512749568 bytes)
   4. no label, UUID=**87976166-4697-47b7-86d1-73a02f0fc803**
   5. **[root@serverb ~]# mkswap /dev/vdb3**
   6. Setting up swapspace version 1, size = 488 MiB (511700992 bytes)

no label, UUID=**4d9b847b-98e0-4d4e-9ef7-dfaaf736b942**

Take note of the UUIDs of the two swap spaces. You use that information in the next step. If you cannot see the **mkswap** output anymore, use the **lsblk --fs** command to retrieve the UUIDs.

* 1. Edit /etc/fstab and define the new swap spaces. To set the swap space on the swap2 partition to be preferred over swap1, give it a higher priority with the pri option.
  2. **[root@serverb ~]# vim /etc/fstab**
  3. *...output omitted...*
  4. UUID=a3665c6b-4bfb-49b6-a528-74e268b058dd /backup xfs defaults 0 0
  5. **UUID=*87976166-4697-47b7-86d1-73a02f0fc803* swap swap pri=10 0 0**

**UUID=*4d9b847b-98e0-4d4e-9ef7-dfaaf736b942* swap swap pri=20 0 0**

* 1. Force systemd to reread the /etc/fstab file.
  2. **[root@serverb ~]# systemctl daemon-reload**

**[root@serverb ~]#**

* 1. Use the **swapon -a** command to activate the new swap spaces. Use the **swapon --show** command to confirm the correct activation of the swap spaces.
  2. **[root@serverb ~]# swapon -a**
  3. **[root@serverb ~]# swapon --show**
  4. NAME TYPE SIZE USED PRIO
  5. **/dev/vdb2** partition 489M 0B 10

**/dev/vdb3** partition 488M 0B 20

1. To verify your work, reboot serverb. Confirm that the system automatically mounts the first partition at /backup. Also, confirm that the system activates the two swap spaces.

When done, log off from serverb.

* 1. Reboot serverb.
  2. **[root@serverb ~]# systemctl reboot**
  3. **[root@serverb ~]#**
  4. Connection to serverb closed by remote host.
  5. Connection to serverb closed.

**[student@workstation ~]$**

* 1. Wait a few minutes for serverb to reboot and then log in as the student user.
  2. **[student@workstation ~]$ ssh student@serverb**
  3. *...output omitted...*

**[student@serverb ~]$**

* 1. Verify that the system automatically mounts /dev/vdb1 at /backup.
  2. **[student@serverb ~]$ mount | grep /backup**

**/dev/vdb1 on /backup type xfs** (rw,relatime,seclabel,attr2,inode64,noquota)

* 1. Use the **swapon --show** command to confirm that the system activates both swap spaces.
  2. **[student@serverb ~]$ swapon --show**
  3. NAME TYPE SIZE USED PRIO
  4. **/dev/vdb2** partition 489M 0B 10

**/dev/vdb3** partition 488M 0B 20

* 1. Log off from serverb.
  2. **[student@serverb ~]$ exit**
  3. logout
  4. Connection to serverb closed.

**[student@workstation ~]$**

## Lab7: Managing Logical Volumes

1. Create a 512 MiB partition on /dev/vdb, initialize it as a physical volume, and extend the serverb\_01\_vg volume group with it.
   1. Log in to serverb as the student user.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*

**[student@serverb ~]$**

* 1. Use the **sudo -i** command to switch to the root user. The password for the student user is student.
  2. **[student@serverb ~]$ sudo -i**
  3. [sudo] password for student: **student**

**[root@serverb ~]#**

* 1. Use **parted** to create the 512 MiB partition and set it to type Linux LVM.
  2. **[root@serverb ~]# parted -s /dev/vdb mkpart primary 514MiB 1026MiB**

**[root@serverb ~]# parted -s /dev/vdb set 2 lvm on**

* 1. Use **udevadm settle** for the system to register the new partition.

**[root@servera ~]# udevadm settle**

* 1. Use **pvcreate** to initialize the partition as a PV.
  2. **[root@serverb ~]# pvcreate /dev/vdb2**

Physical volume "/dev/vdb2" successfully created.

* 1. Use **vgextend** to extend the VG named serverb\_01\_vg, using the new /dev/vdb2 PV.
  2. **[root@serverb ~]# vgextend serverb\_01\_vg /dev/vdb2**

Volume group "serverb\_01\_vg" successfully extended

1. Extend the serverb\_01\_lv logical volume to 768 MiB, including the file system.
   1. Use **lvextend** to extend the serverb\_01\_lv LV to 768 MiB.
   2. **[root@serverb ~]# lvextend -L 768M /dev/serverb\_01\_vg/serverb\_01\_lv**
   3. Size of logical volume serverb\_01\_vg/serverb\_01\_lv changed from 256.00 MiB (64 extents) to 768.00 MiB (192 extents).

Logical volume serverb\_01\_vg/serverb\_01\_lv successfully resized.

### Note

Alternatively, you could have used the -L +512M option to resize the LV.

* 1. Use **xfs\_growfs** to extend the XFS file system to the remainder of the free space on the LV.
  2. **[root@serverb ~]# xfs\_growfs /storage/data1**
  3. meta-data=/dev/mapper/serverb\_01\_vg-serverb\_01\_lv isize=512 agcount=4, agsize=16384 blks

*...output omitted...*

### Note

This example shows the **xfs\_growfs** step to extend the file system. An alternative would have been to add the -r option to the **lvextend** command.

1. In the existing volume group, create a new logical volume called serverb\_02\_lv with a size of 128 MiB. Add an XFS file system and mount it persistently on /storage/data2.
   1. Use **lvcreate** to create a 128 MiB LV named serverb\_02\_lv from the serverb\_01\_vg VG.
   2. **[root@serverb ~]# lvcreate -n serverb\_02\_lv -L 128M serverb\_01\_vg**

Logical volume "serverb\_02\_lv" created

* 1. Use **mkfs** to place an xfs file system on the serverb\_02\_lv LV. Use the LV device name.
  2. **[root@serverb ~]# mkfs -t xfs /dev/serverb\_01\_vg/serverb\_02\_lv**
  3. meta-data=/dev/serverb\_01\_vg/serverb\_02\_lv isize=512 agcount=4, agsize=8192 blks

*...output omitted...*

* 1. Use **mkdir** to create a mount point at /storage/data2.

**[root@serverb ~]# mkdir /storage/data2**

* 1. Add the following line to the end of /etc/fstab on serverb:

/dev/serverb\_01\_vg/serverb\_02\_lv /storage/data2 xfs defaults 1 2

* 1. Use **systemctl daemon-reload** to update systemd with the new /etc/fstab configuration.

**[root@servera ~]# systemctl daemon-reload**

* 1. Use **mount** to verify the /etc/fstab entry and mount the new serverb\_02\_lv LV device.

**[root@serverb ~]# mount /storage/data2**

1. When you are done, reboot your serverb machine, then run the command **lab lvm-review grade** from your workstation machine to verify your work.

Wait until serverb is completely up and then proceed to run the evaluation.

**[root@serverb ~]# systemctl reboot**

## Lab8: Implementing Advanced Storage Features

1. From workstation, open an SSH session to serverb as student.
2. **[student@workstation ~]$ ssh student@serverb**
3. *...output omitted...*

**[student@serverb ~]$**

1. Switch to the root user.
2. **[student@serverb ~]$ sudo -i**
3. **[sudo] password for student: student**

**[root@serverb ~]#**

Hide Solution

1. Install the stratisd and stratis-cli packages using yum.
2. **[root@serverb ~]# yum install stratisd stratis-cli**
3. *...output omitted...*
4. **Is this ok [y/N]: y**
5. *...output omitted...*

Complete!

Hide Solution

1. Start and enable the stratisd service using the systemctl command.

**[root@serverb ~]# systemctl enable --now stratisd**

Hide Solution

1. Create the Stratis pool labpool containing the block device /dev/vdb.
   1. Create the Stratis pool labpool using the stratis pool create command.

**[root@serverb ~]# stratis pool create labpool /dev/vdb**

* 1. Verify the availability of labpool using the stratis pool list command.
  2. **[root@serverb ~]# stratis pool list**
  3. Name Total Physical

labpool **5 GiB** / 37.63 MiB / 4.96 GiB

Note the size of the pool in the preceding output.

1. Hide Solution
2. Expand the capacity of labpool using the disk /dev/vdc available in the system.
   1. Add the block device /dev/vdc to labpool using the stratis pool add-data command.

**[root@serverb ~]# stratis pool add-data labpool /dev/vdc**

* 1. Verify the size of labpool using the stratis pool list command.
  2. **[root@serverb ~]# stratis pool list**
  3. Name Total Physical

labpool **10 GiB** / 41.63 MiB / 9.96 GiB

The preceding output shows that the size of labpool has increased after a new disk was added to the pool.

* 1. Use the stratis blockdev list command to list the block devices that are now members of labpool.
  2. **[root@serverb ~]# stratis blockdev list labpool**
  3. Pool Name Device Node Physical Size Tier
  4. labpool /dev/vdb 5 GiB Data

labpool /dev/vdc 5 GiB Data

1. Hide Solution
2. Create a thinly provisioned file system named labfs in the labpool pool. Mount this file system on /labstratisvol so that it persists across reboots. Create a file named labfile1 that contains the text Hello World! on the labfs file system. Don't forget to use the x-systemd.requires=stratisd.service mount option in /etc/fstab.
   1. Create the thinly provisioned file system labfs in labpool using the stratis filesystem create command. It may take up to a minute for the command to complete.

**[root@serverb ~]# stratis filesystem create labpool labfs**

* 1. Verify the availability of labfs using the stratis filesystem list command.
  2. **[root@serverb ~]# stratis filesystem list**
  3. Pool Name Name Used Created Device UUID

labpool **labfs** **546 MiB** Mar 29 2019 07:48 /stratis/labpool/labfs 9825...d6ca

Note the current usage of labfs. This usage of the file system increases on-demand in the following steps.

* 1. Determine the UUID of labfs using the lsblk command.
  2. **[root@serverb ~]# lsblk --output=UUID /stratis/labpool/labfs**
  3. UUID

9825e289-fb08-4852-8290-44d1b8f0d6ca

* 1. Edit /etc/fstab so that the thinly provisioned file system labfs is mounted at boot time. Use the UUID you determined in the preceding step. The following shows the line you should add to /etc/fstab. You can use the vi /etc/fstab command to edit the file.

UUID=***9825...d6ca*** /labstratisvol xfs defaults,x-systemd.requires=stratisd.service 0 0

* 1. Create a directory named /labstratisvol using the mkdir command.

**[root@serverb ~]# mkdir /labstratisvol**

* 1. Mount the thinly provisioned file system labfs using the mount command to confirm that the /etc/fstab file contains the appropriate entries.

**[root@serverb ~]# mount /labstratisvol**

If the preceding command produces any errors, revisit the /etc/fstab file and ensure that it contains the appropriate entries.

* 1. Create a text file named /labstratisvol/labfile1 using the echo command.

**[root@serverb ~]# echo "Hello World!" > /labstratisvol/labfile1**

1. Hide Solution
2. Verify that the thinly provisioned file system labfs dynamically grows as the data on the file system grows by adding a 2 GiB labfile2 to the filesystem.
   1. View the current usage of labfs using the stratis filesystem list command.
   2. **[root@serverb ~]# stratis filesystem list**
   3. Pool Name Name Used Created Device UUID

labpool labfs **546 MiB** Mar 29 2019 07:48 /stratis/labpool/labfs 9825...d6ca

* 1. Create a 2 GiB file in labfs using the dd command. It may take up to a minute for the command to complete.

**[root@serverb ~]# dd if=/dev/urandom of=/labstratisvol/labfile2 bs=1M count=2048**

* 1. Verify that the usage of labfs has increased, using the stratis filesystem list command.
  2. **[root@serverb ~]# stratis filesystem list**
  3. Pool Name Name Used Created Device UUID
  4. labpool labfs **2.53 GiB** Mar 29 2019 07:48 /stratis/labpool/labfs 9825...d6ca

1. Hide Solution
2. Create a snapshot named labfs-snap of the labfs file system. The snapshot allows you to access any file that is deleted from labfs.
   1. Create a snapshot of labfs using the stratis filesystem snapshot command. It may take up to a minute for the command to complete.
   2. **[root@serverb ~]# stratis filesystem snapshot labpool \**

**labfs labfs-snap**

* 1. Verify the availability of the snapshot using the stratis filesystem list command.
  2. **[root@serverb ~]# stratis filesystem list**
  3. *...output omitted...*

labpool **labfs-snap** 2.53 GiB Mar 29 2019 10:28 /stratis/labpool/labfs-snap 291d...8a16

* 1. Remove the /labstratisvol/labfile1 file.
  2. **[root@serverb ~]# rm /labstratisvol/labfile1**

**rm: remove regular file '/labstratisvol/labfile1'? y**

* 1. Create the /labstratisvol-snap directory using the mkdir command.

**[root@serverb ~]# mkdir /labstratisvol-snap**

* 1. Mount the snapshot labfs-snap on /labstratisvol-snap using the mount command.
  2. **[root@serverb ~]# mount /stratis/labpool/labfs-snap \**

**/labstratisvol-snap**

* 1. Confirm that you can still access the file you deleted from labfs using the snapshot labfs-snap.
  2. **[root@serverb ~]# cat /labstratisvol-snap/labfile1**

Hello World!

1. Hide Solution
2. Create the VDO volume labvdo, with the device /dev/vdd. Set its logical size to 50 GB.
   1. Create the volume labvdo using the vdo create command.
   2. **[root@serverb ~]# vdo create --name=labvdo --device=/dev/vdd --vdoLogicalSize=50G**

*...output omitted...*

* 1. Verify the availability of the volume labvdo using the vdo list command.
  2. **[root@serverb ~]# vdo list**

labvdo

1. Hide Solution
2. Mount the volume labvdo on /labvdovol with the XFS file system so that it persists across reboots. Don't forget to use the x-systemd.requires=vdo.service mount option in /etc/fstab.
   1. Format the labvdo volume with the XFS file system using the mkfs command.
   2. **[root@serverb ~]# mkfs.xfs -K /dev/mapper/labvdo**

*...output omitted...*

* 1. Use the udevadm command to register the new device node.

**[root@serverb ~]# udevadm settle**

* 1. Create the /labvdovol directory using the mkdir command.

**[root@serverb ~]# mkdir /labvdovol**

* 1. Determine the UUID of labvdo using the lsblk command.
  2. **[root@serverb ~]# lsblk --output=UUID /dev/mapper/labvdo**
  3. UUID

ef8cce71-228a-478d-883d-5732176b39b1

* 1. Edit /etc/fstab so that labvdo is mounted at boot time. Use the UUID of the volume you determined in the preceding step. The following shows the line you should add to /etc/fstab. You can use the vi /etc/fstab command to edit the file.

UUID=***ef8c...39b1*** /labvdovol xfs defaults,x-systemd.requires=vdo.service 0 0

* 1. Mount the labvdo volume using the mount command to confirm that the /etc/fstab file contains the appropriate entries.

**[root@serverb ~]# mount /labvdovol**

If the preceding command produces any errors, revisit the /etc/fstab file and ensure that it contains the appropriate entries.

1. Hide Solution
2. Create three copies of the file named /root/install.img on the volume labvdo. Compare the statistics of the volume to verify the data deduplication and compression happening on the volume.
   1. View the initial statistics and status of the volume using the vdostats command.
   2. **[root@serverb ~]# vdostats --human-readable**
   3. Device Size Used Available Use% Space saving%

/dev/mapper/labvdo 5.0G **3.0G** 2.0G 60% **99%**

Notice that 3 GB of the volume is already used because when created, the VDO volume reserves 3-4 GB for itself. Also note that the value 99% in the Space saving% field indicates that you have not created any content so far in the volume, contributing to all of the saved volume space.

* 1. Copy /root/install.img to /labvdovol/install.img.1 and verify the statistics of the volume. It may take up to a minute to copy the file.
  2. **[root@serverb ~]# cp /root/install.img /labvdovol/install.img.1**
  3. **[root@serverb ~]# vdostats --human-readable**
  4. Device Size Used Available Use% Space saving%

/dev/mapper/labvdo 5.0G **3.4G** 1.6G 68% **5%**

Notice that the value of the Used field increased from 3.0G to 3.4G because you copied a file in the volume, and that occupies some space. Also, notice that the value of Space saving% field decreased from 99% to 5% because initially there was no content in the volume, contributing to the low volume space utilization and high volume space saving until you created a file in there. The volume space saving is quite low because you created a unique copy of the file in the volume and there is nothing to deduplicate.

* 1. Copy /root/install.img to /labvdovol/install.img.2 and verify the statistics of the volume. It may take up to a minute to copy the file.
  2. **[root@serverb ~]# cp /root/install.img /labvdovol/install.img.2**
  3. **[root@serverb ~]# vdostats --human-readable**
  4. Device Size Used Available Use% Space saving%

/dev/mapper/labvdo 5.0G **3.4G** 1.6G 68% **51%**

Notice that the used volume space did not change. Instead, the percentage of the saved volume space increased, proving that the data deduplication occurred to reduce the space consumption for the redundant copies of the same file. The value of Space saving% in the preceding output may vary on your system.

1. Hide Solution
2. Reboot serverb. Verify that your labvdo volume is mounted on /labvdovol after the system starts back up.
   1. Reboot the serverb machine.

**[root@serverb ~]# systemctl reboot**

### Note

Note: If on a reboot, serverb does not boot to a regular login prompt but instead has "Give root password for maintenance (or press Control-D to continue):" you likely made a mistake in /etc/fstab. After providing the root password of redhat, you will need to remount the root file system as read-write with:

**[root@serverb ~]# mount -o remount,rw /**

Verify that /etc/fstab is configured correctly as specified in the solutions. Pay special attention to the mount options for the lines related to /labstratisvol and /labvdovol.

## Lab9: Accessing Network-Attached Storage

1. Log in to servera and install the required packages.
   1. Log in to servera as the student user.
   2. **[student@workstation ~]$ ssh student@servera**
   3. *...output omitted...*

**[student@servera ~]$**

* 1. Use the **sudo -i** command to switch to the root user. The password for the student user is student.
  2. **[student@servera ~]$ sudo -i**
  3. [sudo] password for student: **student**

**[root@servera ~]#**

* 1. Install the autofs package.
  2. **[root@servera ~]# yum install autofs**
  3. *...output omitted...*
  4. **Is this ok [y/N]: y**

*...output omitted...*

1. Hide Solution
2. Configure an automounter indirect map on servera using shares from serverb. Create an indirect map using files named /etc/auto.master.d/shares.autofs for the master map and /etc/auto.shares for the mapping file. Use the /remote directory as the main mount point on servera. Reboot servera to determine if the autofs service starts automatically.
   1. Test the NFS server before proceeding to configure the automounter.
   2. **[root@servera ~]# mount -t nfs serverb.lab.example.com:/shares /mnt**
   3. **[root@servera ~]# ls -l /mnt**
   4. total 0
   5. drwxrwx---. 2 root managers 25 Apr 4 01:13 management
   6. drwxrwx---. 2 root operators 25 Apr 4 01:13 operation
   7. drwxrwx---. 2 root production 25 Apr 4 01:13 production

**[root@servera ~]# umount /mnt**

* 1. Create a master map file named /etc/auto.master.d/shares.autofs, insert the following content, and save the changes.
  2. **[root@servera ~]# vim /etc/auto.master.d/shares.autofs**

/remote /etc/auto.shares

* 1. Create an indirect map file named /etc/auto.shares, insert the following content, and save the changes.
  2. **[root@servera ~]# vim /etc/auto.shares**

\* -rw,sync,fstype=nfs4 serverb.lab.example.com:/shares/&

* 1. Start and enable the autofs service on servera.
  2. **[root@servera ~]# systemctl enable --now autofs**

Created symlink /etc/systemd/system/multi-user.target.wants/autofs.service → /usr/lib/systemd/system/autofs.service.

* 1. Reboot theservera machine.

**[root@servera ~]# systemctl reboot**

1. Hide Solution
2. Test the autofs configuration with the various users. When done, log off from servera.
   1. After the servera machine has finished booting, log in to servera as the student user.
   2. **[student@workstation ~]$ ssh student@servera**
   3. *...output omitted...*

**[student@servera ~]$**

* 1. Use the **su - manager1** command to switch to the manager1 user and test access.
  2. **[student@servera ~]$ su - manager1**
  3. Password: **redhat**
  4. **[manager1@servera ~]$ ls -l /remote/management/**
  5. total 4
  6. -rw-r--r--. 1 root managers 46 Apr 4 01:13 Welcome.txt
  7. **[manager1@servera ~]$ cat /remote/management/Welcome.txt**
  8. ###Welcome to Management Folder on SERVERB###
  9. **[manager1@servera ~]$ echo TEST1 > /remote/management/Test.txt**
  10. **[manager1@servera ~]$ cat /remote/management/Test.txt**
  11. TEST1
  12. **[manager1@servera ~]$ ls -l /remote/operation/**
  13. ls: cannot open directory '/remote/operation/': Permission denied
  14. **[manager1@servera ~]$ ls -l /remote/production/**
  15. ls: cannot open directory '/remote/production/': Permission denied
  16. **[manager1@servera ~]$ exit**
  17. logout
  18. **[student@servera ~]$**
  19. Switch to the dbuser1 user and test access.
  20. **[student@servera ~]$ su - dbuser1**
  21. Password: **redhat**
  22. **[dbuser1@servera ~]$ ls -l /remote/production/**
  23. total 4
  24. -rw-r--r--. 1 root production 46 Apr 4 01:13 Welcome.txt
  25. **[dbuser1@servera ~]$ cat /remote/production/Welcome.txt**
  26. ###Welcome to Production Folder on SERVERB###
  27. **[dbuser1@servera ~]$ echo TEST2 > /remote/production/Test.txt**
  28. **[dbuser1@servera ~]$ cat /remote/production/Test.txt**
  29. TEST2
  30. **[dbuser1@servera ~]$ ls -l /remote/operation/**
  31. ls: cannot open directory '/remote/operation/': Permission denied
  32. **[dbuser1@servera ~]$ ls -l /remote/management/**
  33. ls: cannot open directory '/remote/management/': Permission denied
  34. **[dbuser1@servera ~]$ exit**
  35. logout
  36. **[student@servera ~]$**
  37. Switch to the contractor1 user and test access.
  38. **[student@servera ~]$ su - contractor1**
  39. Password: **redhat**
  40. **[contractor1@servera ~]$ ls -l /remote/operation/**
  41. total 4
  42. -rw-r--r--. 1 root operators 45 Apr 4 01:13 Welcome.txt
  43. **[contractor1@servera ~]$ cat /remote/operation/Welcome.txt**
  44. ###Welcome to Operation Folder on SERVERB###
  45. **[contractor1@servera ~]$ echo TEST3 > /remote/operation/Test.txt**
  46. **[contractor1@servera ~]$ cat /remote/operation/Test.txt**
  47. TEST3
  48. **[contractor1@servera ~]$ ls -l /remote/management/**
  49. ls: cannot open directory '/remote/management/': Permission denied
  50. **[contractor1@servera ~]$ ls -l /remote/production/**
  51. ls: cannot open directory '/remote/production/': Permission denied
  52. **[contractor1@servera ~]$ exit**
  53. logout
  54. **[student@servera ~]$**
  55. Explore the **mount** options for the NFS automounted share.
  56. **[student@servera ~]$ mount | grep nfs**
  57. rpc\_pipefs on /var/lib/nfs/rpc\_pipefs type rpc\_pipefs (rw,relatime)
  58. **serverb.lab.example.com:/shares/management** on **/remote/management** type **nfs4**
  59. (rw,relatime,**vers=4.2**,rsize=262144,wsize=262144,namlen=255,
  60. sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
  61. local\_lock=none,addr=172.25.250.11)
  62. **serverb.lab.example.com:/shares/operation** on **/remote/operation** type **nfs4**
  63. (rw,relatime,**vers=4.2**,rsize=262144,wsize=262144,namlen=255,
  64. sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,
  65. local\_lock=none,addr=172.25.250.11)
  66. **serverb.lab.example.com:/shares/production** on **/remote/production** type **nfs4**
  67. (rw,relatime,**vers=4.2**,rsize=262144,wsize=262144,namlen=255,
  68. sync,proto=tcp,timeo=600,retrans=2,sec=sys,clientaddr=172.25.250.10,

local\_lock=none,addr=172.25.250.11)

* 1. Log off from servera.
  2. **[student@servera ~]$ exit**
  3. logout

**[student@workstation ~]$**

## Lab10: Controlling the Boot Process

1. On serverb, reset the root password to redhat.

Locate the icon for the serverb console, as appropriate for your classroom environment. Work from that console.

* 1. Send a **Ctrl**+**Alt**+**Del** to your system using the relevant button or menu entry.
  2. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.
  3. Use the cursor keys to highlight the default boot loader entry.
  4. Press **e** to edit the current entry.
  5. Use the cursor keys to navigate to the line that starts with linux.
  6. Press **End** to move the cursor to the end of the line.
  7. Append rd.break to the end of the line.
  8. Press **Ctrl**+**x** to boot using the modified configuration.
  9. At the switch\_root prompt, remount the /sysroot file system read/write, then use **chroot** to go into a **chroot** jail at /sysroot.
  10. **switch\_root:/# mount -o remount,rw /sysroot**

**switch\_root:/# chroot /sysroot**

* 1. Set the root password to redhat.
  2. **sh-4.4# passwd root**
  3. Changing password for user root.
  4. New password: **redhat**
  5. BAD PASSWORD: The password is shorter than 8 characters
  6. Retype new password: **redhat**

passwd: all authentication tokens updated successfully.

* 1. Configure the system to automatically perform a full SELinux relabel after boot.

**sh-4.4# touch /.autorelabel**

* 1. Type **exit** twice to continue booting your system. The system fails to boot because of an issue you resolve in the next step.

Hide Solution

1. The system fails to boot. A start job does not seem to complete. From the console, fix the issue.
   1. Boot the system into emergency mode. To do so, reboot serverb by sending a **Ctrl**+**Alt**+**Del** to your system using the relevant button or menu entry.
   2. When the boot-loader menu appears, press any key to interrupt the countdown, except **Enter**.
   3. Use the cursor keys to highlight the default boot loader entry.
   4. Press **e** to edit the current entry.
   5. Use the cursor keys to navigate to the line that starts with linux.
   6. Press **End** to move the cursor to the end of the line.
   7. Append systemd.unit=emergency.target to the end of the line.
   8. Press **Ctrl**+**x** to boot using the modified configuration.
   9. Log in to emergency mode. The root password is redhat.
   10. Give root password for maintenance
   11. (or press Control-D to continue): **redhat**

**[root@serverb ~]#**

* 1. Remount the / file system read/write.

**[root@serverb ~]# mount -o remount,rw /**

* 1. Use the **mount -a** command to attempt to mount all the other file systems.
  2. **[root@serverb ~]# mount -a**

mount: /olddata: can't find UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc.

* 1. Edit /etc/fstab to remove or comment out the incorrect line.
  2. **[root@serverb ~]# vim /etc/fstab**
  3. *...output omitted...*

**#**UUID=4d5c85a5-8921-4a06-8aff-80567e9689bc /olddata xfs defaults 0 0

* 1. Update systemd for the system to register the new /etc/fstab configuration.
  2. **[root@serverb ~]# systemctl daemon-reload**

**[root@serverb ~]#**

* 1. Verify that your /etc/fstab is now correct by attempting to mount all entries.
  2. **[root@serverb ~]# mount -a**

**[root@serverb ~]#**

* 1. Reboot the system and wait for the boot to complete. Because you created the /.autorelabel file in the first step, after setting the root password, the system runs an SELinux relabel, then reboots again by itself. The system should now boot normally.

**[root@serverb ~]# systemctl reboot**

1. Hide Solution
2. Change the default systemd target on serverb for the system to automatically start a graphical interface when it boots.

No graphical interface is installed yet on serverb. For this exercise, only set the default target and do not install the packages.

* 1. Log in to serverb as the root user. Use redhat as the password.
  2. Use the **systemctl set-default** command to set graphical.target as the default target.

**[root@serverb ~]# systemctl set-default graphical.target**

* 1. Use the **systemctl get-default** command to verify your work.
  2. **[root@serverb ~]# systemctl get-default**

graphical.target

* 1. Log off from serverb.

**[root@serverb ~]# exit**

## Lab11: Managing Network Security

1. From workstation, test access to the default web server at http://serverb.lab.example.com and to the virtual host at http://serverb.lab.example.com:1001.
   1. Test access to the http://serverb.lab.example.com web server. The test currently fails. Ultimately, the web server should return SERVER B.
   2. **[student@workstation ~]$ curl http://serverb.lab.example.com**
   3. curl: (7) Failed to connect to serverb.lab.example.com port **80**: Connection refused
   4. Test access to the http://serverb.lab.example.com:1001 virtual host. The test currently fails. Ultimately, the virtual host should return VHOST 1.
   5. **[student@workstation ~]$ curl http://serverb.lab.example.com:1001**
   6. curl: (7) Failed to connect to serverb.lab.example.com port **1001**: No route to host

Hide Solution

1. Log in to serverb to determine what is preventing access to the web servers.
   1. From workstation, open an SSH session to serverb as student user. The systems are configured to use SSH keys for authentication, so a password is not required.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*
   4. **[student@serverb ~]$**
   5. Determine whether the httpd service is active.
   6. **[student@serverb ~]$ systemctl is-active httpd**
   7. inactive
   8. Enable and start the httpd service. The httpd service fails to start.
   9. **[student@serverb ~]$ sudo systemctl enable --now httpd**
   10. **[sudo] password for student: student**
   11. Created symlink /etc/systemd/system/multi-user.target.wants/httpd.service → /usr/lib/systemd/system/httpd.service.
   12. Job for httpd.service **failed** because the control process exited with error code.
   13. See "**systemctl status httpd.service**" and "journalctl -xe" for details.
   14. Investigate the reasons why the httpd.service service failed to start.
   15. **[student@serverb ~]$ systemctl status httpd.service**
   16. ● httpd.service - The Apache HTTP Server
   17. Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; vendor preset: disabled)
   18. Active: failed (Result: exit-code) since Thu 2019-04-11 19:25:36 CDT; 19s ago
   19. Docs: man:httpd.service(8)
   20. Process: 9615 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
   21. Main PID: 9615 (code=exited, status=1/FAILURE)
   22. Status: "Reading configuration..."
   23. Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Starting The Apache HTTP Server...
   24. Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission denied: AH00072: make\_sock: **could not bind to address [::]:1001**
   25. Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: (13)Permission denied: AH00072: make\_sock: **could not bind to address 0.0.0.0:1001**
   26. Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: **no listening sockets available, shutting down**
   27. Apr 11 19:25:36 serverb.lab.example.com httpd[9615]: AH00015: Unable to open logs
   28. Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Main process exited, code=exited, status=1/FAILURE
   29. Apr 11 19:25:36 serverb.lab.example.com systemd[1]: httpd.service: Failed with result 'exit-code'.
   30. Apr 11 19:25:36 serverb.lab.example.com systemd[1]: Failed to start The Apache HTTP Server.
   31. Use the **sealert** command to check whether SELinux is blocking the httpd service from binding to port 1001/TCP.
   32. **[student@serverb ~]$ sudo sealert -a /var/log/audit/audit.log**
   33. 100% done
   34. found 1 alerts in /var/log/audit/audit.log
   35. --------------------------------------------------------------------------------
   36. **SELinux is preventing /usr/sbin/httpd from name\_bind access on the tcp\_socket port 1001.**
   37. \*\*\*\*\* Plugin bind\_ports (99.5 confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   38. If you want to allow /usr/sbin/httpd to bind to network port 1001
   39. Then you need to modify the port type.
   40. Do
   41. **# semanage port -a -t PORT\_TYPE -p tcp 1001**
   42. **where PORT\_TYPE is one of the following: http\_cache\_port\_t, http\_port\_t, jboss\_management\_port\_t, jboss\_messaging\_port\_t, ntop\_port\_t, puppet\_port\_t.**
   43. \*\*\*\*\* Plugin catchall (1.49 confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
   44. *...output omitted...*

Hide Solution

1. Configure SELinux to allow the httpd service to listen on port 1001/TCP.
   1. Use the **semanage** command to find the correct port type.
   2. **[student@serverb ~]$ sudo semanage port -l | grep 'http'**
   3. http\_cache\_port\_t tcp 8080, 8118, 8123, 10001-10010
   4. http\_cache\_port\_t udp 3130
   5. **http\_port\_t** tcp 80, 81, 443, 488, 8008, 8009, 8443, 9000
   6. pegasus\_http\_port\_t tcp 5988
   7. pegasus\_https\_port\_t tcp 5989
   8. Use the **semanage** command to bind port 1001/TCP to the http\_port\_t type.
   9. **[student@serverb ~]$ sudo semanage port -a -t http\_port\_t -p tcp 1001**
   10. **[student@serverb ~]$**
   11. Confirm that port 1001/TCP is bound to the http\_port\_t port type.
   12. **[student@serverb ~]$ sudo semanage port -l | grep '^http\_port\_t'**
   13. **http\_port\_t** tcp **1001**, 80, 81, 443, 488, 8008, 8009, 8443, 9000
   14. Enable and start the httpd service.
   15. **[student@serverb ~]$ sudo systemctl enable --now httpd**
   16. Verify the running state of the httpd service.
   17. **[student@serverb ~]$ systemctl is-active httpd; systemctl is-enabled httpd**
   18. active
   19. enabled
   20. Exit from serverb.
   21. **[student@serverb ~]$ exit**
   22. logout
   23. Connection to serverb closed.
   24. [student@workstation ~]$

Hide Solution

1. From workstation, test access to the default web server at http://serverb.lab.example.com and to the virtual host at http://serverb.lab.example.com:1001.
   1. Test access to the http://serverb.lab.example.com web server. The web server should return SERVER B.
   2. **[student@workstation ~]$ curl http://serverb.lab.example.com**
   3. SERVER B
   4. Test access to the http://serverb.lab.example.com:1001 virtual host. The test continues to fail.
   5. **[student@workstation ~]$ curl http://serverb.lab.example.com:1001**
   6. curl: (7) Failed to connect to serverb.lab.example.com port 1001: No route to host

Hide Solution

1. Log in to serverb to determine whether the correct ports are assigned to the firewall.
   1. From workstation, log in to serverb as the student user.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*
   4. **[student@serverb ~]$**
   5. Verify that the default firewall zone is set to public.
   6. **[student@serverb ~]$ firewall-cmd --get-default-zone**

public

* 1. If the previous step did not return public as the default zone, correct it with the following command:

**[student@serverb ~]$ sudo firewall-cmd --set-default-zone public**

* 1. Determine the open ports listed in the public network zone.
  2. **[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all**
  3. **[sudo] password for student: student**
  4. public
  5. target: default
  6. icmp-block-inversion: no
  7. interfaces:
  8. sources:
  9. services: cockpit dhcpv6-client http ssh
  10. **ports**:
  11. protocols:
  12. masquerade: no
  13. forward-ports:
  14. source-ports:
  15. icmp-blocks:
  16. rich rules:

Hide Solution

1. Add port 1001/TCP to the permanent configuration for the public network zone. Confirm your configuration.
   1. Add port 1001/TCP to the public network zone.
   2. **[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public \**
   3. **--add-port=1001/tcp**
   4. success
   5. Reload the firewall configuration.
   6. **[student@serverb ~]$ sudo firewall-cmd --reload**
   7. success
   8. Confirm your configuration.
   9. **[student@serverb ~]$ sudo firewall-cmd --permanent --zone=public --list-all**
   10. public
   11. target: default
   12. icmp-block-inversion: no
   13. interfaces:
   14. sources:
   15. services: cockpit dhcpv6-client http ssh
   16. ports: **1001/tcp**
   17. protocols:
   18. masquerade: no
   19. forward-ports:
   20. source-ports:
   21. icmp-blocks:
   22. rich rules:
   23. Exit from serverb.
   24. **[student@serverb ~]$ exit**
   25. logout
   26. Connection to serverb closed.
   27. [student@workstation ~]$

Hide Solution

1. From workstation, confirm that the default web server at serverb.lab.example.com returns SERVER B and the virtual host at serverb.lab.example.com:1001 returns VHOST 1.
   1. Test access to the http://serverb.lab.example.com web server.
   2. **[student@workstation ~]$ curl http://serverb.lab.example.com**
   3. SERVER B
   4. Test access to the http://serverb.lab.example.com:1001 virtual host.
   5. **[student@workstation ~]$ curl http://serverb.lab.example.com:1001**
   6. VHOST 1

Hide Solution

## Lab12: Installing Red Hat Enterprise Linux

1. On serverb, copy /root/anaconda-ks.cfg to /home/student/kickstart.cfg, so the student user can edit it.
   1. Use the **ssh** command to log in to serverb as the student user.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*

[student@serverb ~]$

* 1. Copy /root/anaconda-ks.cfg on serverb to a file called /home/student/kickstart.cfg so that student can edit. Use the **sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg** command to copy the contents of /root/anaconda-ks.cfg to /home/student/kickstart.cfg. If **sudo** prompts for the password of the student user, use student as the password.
  2. **[student@serverb ~]$ sudo cat /root/anaconda-ks.cfg > ~/kickstart.cfg**

**[sudo] password for student: student**

1. Hide Solution
2. Make the following changes to /home/student/kickstart.cfg.
   * Comment out the **reboot** command.
   * Comment out the **repo** command for the BaseOS repository. Modify the **repo** command for the AppStream repository to point to http://classroom.example.com/content/rhel8.2/x86\_64/dvd/AppStream/. The repository name should be set to appstream.
   * Change the **url** command to use http://classroom.example.com/content/rhel8.2/x86\_64/dvd/ as the installation source.
   * Comment out the **network** command.
   * Change the **rootpw** command to use plaintext and set the root password to redhat.
   * Delete the line that uses the **auth** command and add the **authselect select sssd** line to set the sssd service as the identity and authentication source.
   * Simplify the **services** command so that only the kdump and rhsmcertd services are disabled. Leave only the sshd, rngd, and chronyd enabled.
   * Add the **autopart** command. The **part** commands should be commented out.
   * Simplify the %post section so that it only runs a script to append the text Kickstarted on *DATE* to the end of the /etc/issue file. *DATE* is variable information and should be generated by the script using the **date** command with no additional options.
   * Simplify the %package section as follows: include the @core, chrony, dracut-config-generic, dracut-norescue, firewalld, grub2, kernel, rsync, tar, and httpd packages. Ensure that the plymouth package is not installed.
   1. Comment out the reboot directive:

#reboot

* 1. The **repo** command is found twice in kickstart.cfg. Comment out the **repo** command for the BaseOS repository. Modify the **repo** command for the AppStream repository to point to the classroom's AppStream repository:
  2. #repo --name="koji-override-0" --baseurl=http://download-node-02.eng.bos.redhat.com/rhel-8/devel/candidate-trees/RHEL-8/RHEL-8.2.0-updates-20200423.0/compose/BaseOS/x86\_64/os

repo --name="appstream" --baseurl=http://classroom.example.com/content/rhel8.2/x86\_64/dvd/AppStream/

* 1. Change the **url** command to specify the HTTP installation source media used in the classroom:

url --url="http://classroom.example.com/content/rhel8.2/x86\_64/dvd/"

* 1. Comment out the **network** command:

#network --bootproto=dhcp --device=link --activate

* 1. Set the root password to redhat. Change the line that starts with **rootpw** to:

rootpw --plaintext redhat

* 1. Delete the line that uses the **auth** command and add the **authselect select sssd** line to set the sssd service as the identity and authentication source.

authselect select sssd

* 1. Simplify the **services** command to look exactly like the following:

services --disabled="kdump,rhsmcertd" --enabled="sshd,rngd,chronyd"

* 1. Comment out the **part** commands. Add the **autopart** command:
  2. # Disk partitioning information
  3. #part biosboot --fstype="biosboot" --size=1
  4. #part /boot/efi --fstype="efi" --size=100 --fsoptions="..."
  5. #part / --fstype="xfs" --size=10137 --label=root

autopart

* 1. Delete all content between the %post section and its %end. Add the following line: **echo "Kickstarted on $(date)" >> /etc/issue**

The entire %post section should look like this.

%post --erroronfail

echo "Kickstarted on $(date)" >> /etc/issue

%end

* 1. Simplify the package specification to look exactly like the following:
  2. %packages
  3. @core
  4. chrony
  5. dracut-config-generic
  6. dracut-norescue
  7. firewalld
  8. grub2
  9. kernel
  10. rsync
  11. tar
  12. httpd
  13. -plymouth

%end

1. Hide Solution
2. Validate the syntax of kickstart.cfg.
   1. Use the **ksvalidator** command to check the Kickstart file for syntax errors.

**[student@serverb ~]$ ksvalidator kickstart.cfg**

1. Hide Solution
2. Make the /home/student/kickstart.cfg file available at http://serverb.lab.example.com/ks-config/kickstart.cfg
   1. Copy kickstart.cfg to the /var/www/html/ks-config/ directory.

**[student@serverb ~]$ sudo cp ~/kickstart.cfg /var/www/html/ks-config**

1. Hide Solution
2. Return to the workstation system to check your work.
   1. Exit from serverb.
   2. **[student@serverb ~]$ exit**
   3. logout
   4. Connection to serverb closed.

**[student@workstation ~]$**

## Lab13: Running Containers

1. On serverb, install the container tools. Log in to serverb as the student user, and then use the sudo command. The password for the student user is student.
   1. Use the ssh command to log in to serverb as the student user. The systems are configured to use SSH keys for authentication, so a password is not required.
   2. **[student@workstation ~]$ ssh student@serverb**
   3. *...output omitted...*

**[student@serverb ~]$**

* 1. Install the container-tools Yum module using the yum command.
  2. **[student@serverb ~]$ sudo yum module install container-tools**
  3. **[sudo] password for student: student**
  4. *...output omitted...*
  5. **Is this ok [y/N]: y**
  6. *...output omitted...*

Complete!

1. Hide Solution
2. The container image registry at registry.lab.example.com stores the rhel8/mariadb-103 image with several tags. On serverb, as the podsvc user, list those tags and take note of the tag with the *lowest* version number. You will use that image tag to start a container later in this exercise.

The password for the podsvc user is redhat. To query the registry.lab.example.com registry, use the admin account with redhat321 for the password.

* 1. Exit from the student account on serverb.
  2. **[student@serverb ~]$ exit**
  3. logout
  4. Connection to serverb closed.

**[student@workstation ~]$**

* 1. Use the ssh command to log in to serverb as the podsvc user. The systems are configured to use SSH keys for authentication, so a password is not required.
  2. **[student@workstation ~]$ ssh podsvc@serverb**
  3. *...output omitted...*

**[podsvc@serverb ~]$**

* 1. Log in to the container registry using the podman login command.
  2. **[podsvc@serverb ~]$ podman login registry.lab.example.com**
  3. **Username: admin**
  4. **Password: redhat321**

Login Succeeded!

* 1. Use the skopeo inspect command to view information about the registry.lab.example.com/rhel8/mariadb-103 image. The following skopeo inspect command is very long and should be entered as a single line.
  2. **[podsvc@serverb ~]$ skopeo inspect docker://registry.lab.example.com/rhel8/mariadb-103**
  3. {
  4. "Name": "registry.lab.example.com/rhel8/mariadb-103",
  5. "Digest": "sha256:a95b...4816",
  6. "RepoTags": [
  7. "**1-86**",
  8. "1-102",
  9. "latest"
  10. ],

*...output omitted...*

The tag with the lowest number is 1-86.

Hide Solution

1. On serverb, as the podsvc user, create the /home/podsvc/db\_data directory. Prepare the directory so that containers have read/write access. You will use this directory for persistent storage.
   1. Create the /home/podsvc/db\_data directory.
   2. **[podsvc@serverb ~]$ mkdir /home/podsvc/db\_data**

**[podsvc@serverb ~]$**

* 1. Set the access mode of the directory to 777 so that everyone has read/write access.
  2. **[podsvc@serverb ~]$ chmod 777 /home/podsvc/db\_data**

**[podsvc@serverb ~]$**

1. Hide Solution
2. On serverb, as the podsvc user, create a detached MariaDB container named inventorydb. Use the rhel8/mariadb-103 image from the registry.lab.example.com registry, specifying the tag with the lowest version number on that image, which you found in a preceding step. Map port 3306 in the container to port 13306 on the host. Mount the /home/podsvc/db\_data directory on the host as /var/lib/mysql/data in the container. Declare the following variable values:

| **Variable** | **Value** |
| --- | --- |
| MYSQL\_USER | operator1 |
| MYSQL\_PASSWORD | redhat |
| MYSQL\_DATABASE | inventory |
| MYSQL\_ROOT\_PASSWORD | redhat |

1. You can copy and paste these parameters from the /home/podsvc/containers-review/variables file on serverb.
2. To confirm that the MariaDB database is running, use the mysql command. You can find this command in the /home/podsvc/containers-review/testdb.sh script. You can also directly run the script to test the database.
   1. Use the podman run command to create the container. The following podman run command is very long and should be entered as a single line.
   2. **[podsvc@serverb ~]$ podman run -d --name inventorydb -p 13306:3306 -v /home/podsvc/db\_data:/var/lib/mysql/data:Z -e MYSQL\_USER=operator1 -e MYSQL\_PASSWORD=redhat -e MYSQL\_DATABASE=inventory -e MYSQL\_ROOT\_PASSWORD=redhat registry.lab.example.com/rhel8/mariadb-103:1-86**

*...output omitted...*

* 1. Confirm that the database is running.
  2. **[podsvc@serverb ~]$ ~/containers-review/testdb.sh**
  3. Testing the access to the database...

**SUCCESS**

1. Hide Solution
2. On serverb, as the podsvc user, configure systemd so that the inventorydb container starts automatically with the server.
   1. If you used sudo or su to log in as the podsvc user, then exit serverb and use the ssh command to directly log in to serverb as the podsvc user. Remember, systemd requires that the user open a direct session from the console or through SSH.
   2. **[student@workstation ~]$ ssh podsvc@serverb**
   3. *...output omitted...*

**[podsvc@serverb ~]$**

* 1. Create the ~/.config/systemd/user/ directory.
  2. **[podsvc@serverb ~]$ mkdir -p ~/.config/systemd/user/**

**[podsvc@serverb ~]$**

* 1. Use the podman generate systemd command to create the systemd unit file from the running container.
  2. **[podsvc@serverb ~]$ cd ~/.config/systemd/user/**
  3. **[podsvc@serverb user]$ podman generate systemd --name inventorydb --files --new**

/home/podsvc/.config/systemd/user/container-inventorydb.service

* 1. Stop and then delete the inventorydb container.
  2. **[podsvc@serverb user]$ podman stop inventorydb**
  3. 0d28f0e0a4118ff019691e34afe09b4d28ee526079b58d19f03b324bd04fd545
  4. **[podsvc@serverb user]$ podman rm inventorydb**

0d28f0e0a4118ff019691e34afe09b4d28ee526079b58d19f03b324bd04fd545

* 1. Instruct systemd to reload its configuration, and then enable and start the container-inventorydb service.
  2. **[podsvc@serverb user]$ systemctl --user daemon-reload**
  3. **[podsvc@serverb user]$ systemctl --user enable --now container-inventorydb.service**
  4. Created symlink /home/podsvc/.config/systemd/user/multi-user.target.wants/container-inventorydb.service → /home/podsvc/.config/systemd/user/container-inventorydb.service.

Created symlink /home/podsvc/.config/systemd/user/default.target.wants/container-inventorydb.service → /home/podsvc/.config/systemd/user/container-inventorydb.service.

* 1. Confirm that the container is running.
  2. **[podsvc@serverb user]$ ~/containers-review/testdb.sh**
  3. Testing the access to the database...
  4. SUCCESS
  5. **[podsvc@serverb user]$ podman ps**
  6. CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

3ab24e7f000d registry.lab.example.com/rhel8/mariadb-103:1-86 run-mysqld 47 seconds ago Up 46 seconds ago 0.0.0.0:13306->3306/tcp inventorydb

* 1. Run the loginctl enable-linger command for the user services to start automatically when the server starts.
  2. **[podsvc@serverb ~]$ loginctl enable-linger**

**[podsvc@serverb ~]$**

* 1. Exit from serverb.
  2. **[podsvc@serverb ~]$ exit**
  3. logout
  4. Connection to serverb closed.

**[student@workstation ~]$**

## Lab 14: Configuring and Managing Server Security

1. From workstation, open an SSH session to serverb as student.
   1. **[student@workstation ~]$ ssh student@serverb**
   2. *...output omitted...*

Hide Solution

1. Generate SSH keys for the student user on serverb using the ssh-keygen command. Do not protect the private key with a passphrase.
   1. **[student@serverb ~]$ ssh-keygen**
   2. Generating public/private rsa key pair.
   3. **Enter file in which to save the key (/home/student/.ssh/id\_rsa): Enter**
   4. Created directory '/home/student/.ssh'.
   5. **Enter passphrase (empty for no passphrase): Enter**
   6. **Enter same passphrase again: Enter**
   7. Your identification has been saved in /home/student/.ssh/id\_rsa.
   8. Your public key has been saved in /home/student/.ssh/id\_rsa.pub.
   9. The key fingerprint is:
   10. SHA256:1TPZ4TXYwiGWfExUGtRTHgfKQbF9hVuLa+VmH4vgkFY student@serverb.lab.example.com
   11. The key's randomart image is:
   12. +---[RSA 2048]----+
   13. | .+@BO\*\*|
   14. | .=.#+B\*|
   15. | . X.\*o=|
   16. | . E +.+ |
   17. | S o + |
   18. | + . o = |
   19. | . o o + +|
   20. | . . ..|
   21. | |
   22. +----[SHA256]-----+

Hide Solution

1. On servera, configure the student user to accept login authentication using the SSH key pair you created for student on serverb. The student user on serverb should be able to log in to servera using SSH without entering a password. Use student as the password of the student user, when required.
   1. Use the ssh-copy-id command to transfer the public key of the SSH key pair of student on serverb to student on servera. Use student as the password of the student user, if prompted.
   2. **[student@serverb ~]$ ssh-copy-id student@servera**
   3. /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/student/.ssh/id\_rsa.pub"
   4. The authenticity of host 'servera (172.25.250.10)' can't be established.
   5. ECDSA key fingerprint is SHA256:g/fIMtVzDWTbTi1l0OwC30sL6cHmro9Tf563NxmeyyE.
   6. **Are you sure you want to continue connecting (yes/no)? yes**
   7. /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
   8. /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
   9. **student@servera's password: student**
   10. Number of key(s) added: 1
   11. Now try logging into the machine, with: "ssh 'student@servera'"

and check to make sure that only the key(s) you wanted were added.

* 1. Use the ssh command to verify that the student user can log in to servera from serverb without entering a password.
  2. **[student@serverb ~]$ ssh student@servera**
  3. *...output omitted...*

**[student@servera ~]$**

1. Hide Solution
2. On servera, change the default SELinux mode to permissive.
   1. Edit /etc/sysconfig/selinux to set the value of the parameter SELINUX to permissive. You can use the sudo vi /etc/sysconfig/selinux command to edit the configuration file as the superuser. Use the password student, if prompted.
   2. *...output omitted...*
   3. #SELINUX=enforcing
   4. SELINUX=permissive

*...output omitted...*

* 1. Use the sudo systemctl reboot command to reboot the system as the superuser.
  2. **[student@servera ~]$ sudo systemctl reboot**
  3. Connection to servera closed by remote host.
  4. Connection to servera closed.

**[student@serverb ~]$**

1. Hide Solution
2. Configure serverb to automatically mount the home directory of the production5 user when the user logs in, using the network file system /home-directories/production5. This network file system is exported from servera.lab.example.com. Adjust the appropriate SELinux Boolean so that production5 can use the NFS-mounted home directory on serverb after authenticating via SSH key-based authentication. The production5 user's password is redhat.
   1. On serverb, use the sudo -i command to switch to the root user account.
   2. **[student@serverb ~]$ sudo -i**
   3. **[sudo] password for student: student**

**[root@serverb ~]#**

* 1. Install the autofs package.
  2. **[root@serverb ~]# yum install autofs**
  3. *...output omitted...*
  4. **Is this ok [y/N]: y**
  5. *...output omitted...*
  6. Installed:
  7. autofs-1:5.1.4-29.el8.x86\_64

Complete!

* 1. Create the autofs master map file called /etc/auto.master.d/production5.autofs with the following content.

/- /etc/auto.production5

* 1. Retrieve the details of the production5 user to get the home directory path.
  2. **[root@serverb ~]# getent passwd production5**

production5:x:5001:5001::**/localhome/production5**:/bin/bash

* 1. Create the /etc/auto.production5 file with the following content.

/localhome/production5 -rw servera.lab.example.com:/home-directories/production5

* 1. Restart the autofs service.

**[root@serverb ~]# systemctl restart autofs**

1. Hide Solution
2. On servera, verify that the production5 user is not able to log in to serverb using SSH public-key authentication. An SELinux Boolean causes this issue which you will fix in the following steps.
   1. From workstation, open an SSH session to servera as student.
   2. **[student@workstation ~]$ ssh student@servera**
   3. *...output omitted...*

**[student@servera ~]$**

* 1. Switch to the production5 user using the password redhat.
  2. **[student@servera ~]$ su - production5**
  3. Password: **redhat**

**[production5@servera ~]$**

* 1. Use the ssh-keygen command to generate the SSH keys as production5.
  2. **[production5@servera ~]$ ssh-keygen**
  3. Generating public/private rsa key pair.
  4. **Enter file in which to save the key (/home/production5/.ssh/id\_rsa): Enter**
  5. Created directory '/home/production5/.ssh'.
  6. **Enter passphrase (empty for no passphrase): Enter**
  7. **Enter same passphrase again: Enter**
  8. Your identification has been saved in /home/production5/.ssh/id\_rsa.
  9. Your public key has been saved in /home/production5/.ssh/id\_rsa.pub.
  10. The key fingerprint is:
  11. SHA256:zmin1nmCt4H8LA+4FPimtdg81nl7ATbInUFW3HSPxk4 production5@servera.lab.example.com
  12. The key's randomart image is:
  13. +---[RSA 2048]----+
  14. | .oo.o. . |
  15. | .. . .o o |
  16. | . o o E .|
  17. | . o \* + |
  18. | . . .So . |
  19. | . + = . |
  20. | \*.\*+=. . |
  21. | Oo+\*\*\*.o |
  22. | o.=o.=\*\* |

+----[SHA256]-----+

* 1. Use the ssh-copy-id command to transfer the public key of the SSH key pair of production5 on servera to production5 on serverb. Use redhat as the password of the production5 user, if prompted.
  2. **[production5@servera ~]$ ssh-copy-id production5@serverb**
  3. /usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/production5/.ssh/id\_rsa.pub"
  4. The authenticity of host 'serverb (172.25.250.11)' can't be established.
  5. ECDSA key fingerprint is SHA256:ciCkaRWF4g6eR9nSdPxQ7KL8czpViXal6BousK544TY.
  6. **Are you sure you want to continue connecting (yes/no)? yes**
  7. /usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
  8. /usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
  9. **production5@serverb's password: redhat**
  10. Number of key(s) added: 1
  11. Now try logging into the machine, with: "ssh 'production5@serverb'"

and check to make sure that only the key(s) you wanted were added.

* 1. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should fail.
  2. **[production5@servera ~]$ ssh -o pubkeyauthentication=yes \**
  3. **-o passwordauthentication=no production5@serverb**

production5@serverb: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).

1. Hide Solution
2. Set the appropriate SELinux Boolean setting on serverb, so that production5 can log in to serverb using the SSH public key-based authentication and use the home directory.
   1. On serverb as root, set the use\_nfs\_home\_dirs SELinux Boolean to true.

**[root@serverb ~]# setsebool -P use\_nfs\_home\_dirs true**

* 1. Use the SSH public key-based authentication instead of password-based authentication to log in to serverb as production5. This command should succeed.
  2. **[production5@servera ~]$ ssh -o pubkeyauthentication=yes \**
  3. **-o passwordauthentication=no production5@serverb**
  4. *...output omitted...*

**[production5@serverb ~]$**

1. Hide Solution
2. On serverb, adjust the firewall settings so that SSH connections originating from servera are rejected. The servera system uses the IPv4 address 172.25.250.10.
   1. Use the firewall-cmd command to add the IPv4 address of servera to the firewalld zone called block.
   2. **[root@serverb ~]# firewall-cmd --add-source=172.25.250.10/32 \**
   3. **--zone=block --permanent**

success

* 1. Use the firewall-cmd --reload command to reload the changes in the firewall settings.
  2. **[root@serverb ~]# firewall-cmd --reload**

success

1. Hide Solution
2. On serverb, investigate and fix the issue with the Apache HTTPD daemon, which is configured to listen on port 30080/TCP, but which fails to start. Adjust the firewall settings appropriately so that port 30080/TCP is open for incoming connections.
   1. Use the systemctl command to restart the httpd service. This command fails to restart the service.
   2. **[root@serverb ~]# systemctl restart httpd.service**
   3. Job for httpd.service failed because the control process exited with error code.

See "systemctl status httpd.service" and "journalctl -xe" for details.

* 1. Use the systemctl status command to investigate the reason for the failure of the httpd service.
  2. **[root@serverb ~]# systemctl status httpd.service**
  3. ● httpd.service - The Apache HTTP Server
  4. Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
  5. Active: failed (Result: exit-code) since Mon 2019-04-15 06:42:41 EDT; 5min ago
  6. Docs: man:httpd.service(8)
  7. Process: 27313 ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND (code=exited, status=1/FAILURE)
  8. Main PID: 27313 (code=exited, status=1/FAILURE)
  9. Status: "Reading configuration..."
  10. Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Starting The Apache HTTP Server...
  11. **Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission denied: AH00072: make\_sock: could not bind to address [::]:30080**
  12. **Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: (13)Permission denied: AH00072: make\_sock: could not bind to address 0.0.0.0:30080**
  13. Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: no listening sockets available, shutting down
  14. Apr 15 06:42:41 serverb.lab.example.com httpd[27313]: AH00015: Unable to open logs
  15. Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service: Main process exited, code=exited, status=1/FAILURE
  16. Apr 15 06:42:41 serverb.lab.example.com systemd[1]: httpd.service: Failed with result 'exit-code'.

Apr 15 06:42:41 serverb.lab.example.com systemd[1]: Failed to start The Apache HTTP Server.

Notice the permission error in the preceding output, which signifies that the httpd daemon failed to bind to port 30080/TCP. The SELinux policy can be a potential restriction for an application to bind to a port. Press **q** to quit the preceding systemctl command.

* 1. Use the sealert command to determine if an SELinux policy is preventing httpd from binding to port 30080/TCP.
  2. **[root@serverb ~]# sealert -a /var/log/audit/audit.log**
  3. 100% done
  4. found 1 alerts in /var/log/audit/audit.log
  5. --------------------------------------------------------------------------------
  6. SELinux is preventing /usr/sbin/httpd from name\_bind access on the tcp\_socket port 30080.
  7. \*\*\*\*\* Plugin bind\_ports (92.2 confidence) suggests \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*
  8. If you want to allow /usr/sbin/httpd to bind to network port 30080
  9. Then you need to modify the port type.
  10. Do
  11. # semanage port -a -t PORT\_TYPE -p tcp 30080
  12. where PORT\_TYPE is one of the following: http\_cache\_port\_t, http\_port\_t, jboss\_management\_port\_t, jboss\_messaging\_port\_t, ntop\_port\_t, puppet\_port\_t.

*...output omitted...*

The preceding log message reveals that the port 30080/TCP does not have the appropriate SELinux context http\_port\_t, causing SELinux to prevent httpd to bind to this port. The log message also produces the syntax of the semanage port command so that you can easily fix the issue.

* 1. Use the semanage port command to set the appropriate SELinux context on the port 30080/TCP for httpd to bind to it.

**[root@serverb ~]# semanage port -a -t http\_port\_t -p tcp 30080**

* 1. Use the systemctl command to restart httpd. This command should successfully restart the service.

**[root@serverb ~]# systemctl restart httpd**

* 1. Add the port 30080/TCP to the default firewalld zone called public.
  2. **[root@serverb ~]# firewall-cmd --add-port=30080/tcp --permanent**
  3. success
  4. **[root@serverb ~]# firewall-cmd --reload**

success

* 1. Exit the root user's shell.
  2. **[root@serverb ~]# exit**

logout

* 1. Log off from serverb.
  2. **[student@serverb ~]$ exit**
  3. logout

Connection to serverb closed.