

RH254 - Red Hat System Administration 3

Travis Michette

Version 1.0

Table of Contents

Before You Begin	1
1. Controlling Services and Daemons	2
1.1. Controlling Services with systemctl	2
1.2. Controlling the Boot Process	3
1.2.1. systemd Targets	3
1.2.2. Reset Root Password	4
2. Managing IPv6 Networking	15
2.1. Review of IPv4 Networking Configuration	15
2.1.1. Summary of Commands	17
2.2. IPv6 Networking Concepts	17
2.3. IPv6 Networking Configuration	19
3. Configuring Link Aggregation and Bridging	21
3.1. Configuring Network Teaming	21
3.2. Managing Network Teaming	23
3.3. Configuring Software Bridges	24
4. Network Port Security	26
4.1. Managing Firewall	26
4.2. Managing Rich Rules	29
4.3. Masquerading and Port Forwarding	30
4.3.1. Masquerading	31
4.3.2. Port Forwarding	32
4.4. Managing SELinux Port Labeling	32
5. Managing DNS for Servers	35
5.1. DNS Concepts	35
5.2. Configuring a Caching Nameserver	38
5.3. DNS Troubleshooting	40
6. Configuring Email Transmission	43
6.1. Configuring Send-only Email Service	43
7. Providing Remote Block Storage	47
7.1. iSCSI Concepts	47
7.2. Providing iSCSI Targets	49
7.3. Accessing iSCSI Storage	51
8. Providing File-based Storage	54
8.1. Exporting NFS File Systems	54
8.2. Protecting NFS Exports	56
8.3. Providing SMB File Shares	59
8.4. Performing a Multi-user SMB Mount	62
9. Configuring MariaDB Databases	63
9.1. Installing MariaDB	63
9.2. Working with MariaDB Databases	65
9.3. Managing Database Users and Access Rights	66
9.4. Creating and Restoring MariaDB Backups	66
10. Providing Apache HTTPD Web Service	68

10.1. Configuring Apache HTTPD	68
10.1.1. HTTPD Security (SELinux)	69
10.1.2. Write Access to DocumentRoot	70
10.2. Configuring and Troubleshooting Virtual Hosts	70
10.3. Configuring HTTPS	71
10.4. Integrating Dynamic Web Content	73
11. Writing Bash Scripts	75
11.1. Bash Shell Scripting Basics	75
12. Bash Conditionals and Control Structures	77
12.1. Enhancing Bash Shell Scripts with Conditionals and Control Structures	77
13. Configuring the Shell Environment	78
13.1. Changing the Shell Environment	78
Appendix A: Exam Objectives for EX300 RHCE Exam	79
A.1. System configuration and management	79
A.2. Network services	79
A.2.1. HTTP/HTTPS	79
A.2.2. DNS	80
A.2.3. NFS	80
A.2.4. SMB	80
A.2.5. SMTP	80
A.2.6. SSH	80
A.2.7. NTP	80
A.3. Database services	80

Before You Begin

In this course, students will do most hands-on practice exercises and lab work with two computer systems, which will be referred to as desktop and server. These machines have host names `desktopX.example.com` and `serverX.example.com`, where the number `X` in the computers' host names will be a number that will vary from student to student. Both machines have a standard user account, `student`, with the password `student`. The root password on both systems is `redhat`.

The systems used by each student use separate IPv4 subnets. For a specific student, their IPv4 network is `172.25.X.0/24`, where the number `X` matches the number in the host name of their desktop and server systems.

Table 1. Classroom Machines

Machine Name	IP addresses	Role
<code>desktopX.example.com</code>	<code>172.25.X.10</code>	Student "client" computer
<code>serverX.example.com</code>	<code>172.25.X.11</code>	Student "server" computer

Resetting a System

```
[kiosk@foundation0 ~]$ rht-vmctl reset server
```



- Resources for Demos: <http://classroom/materials/>
- Presentation Slides: <http://classroom/content/slides/RH254-RHEL7-en-2-20150427-slides.pdf>

1. Controlling Services and Daemons

1.1. Controlling Services with systemctl

systemd replaces the legacy **init** system process which also replaces less frequently used services like **inetd** and **xinetd**. The **inetd** and **xinetd** services were started on demand.

Benefits of **systemd**

- Parallel loading of daemons (increased boot speed)
- ON-demand starting of daemons w/o separate service
- Automatic service dependency - prevents timeouts
- Tracking related processes with CGroups

systemctl interacts with **systemd** services.

Daemon: Process waiting to run in the background. Daemons listen for connections using a **socket**.

Service: One or more daemons. The act of starting/stopping a service might make a change to the system leaving the daemon process stopped after the change has been made.

Table 2. **systemd** Unit Types

Type	Description
Service Unit	Has a .service extension. Used to start frequently accessed daemons.
Socket Unit	Has a .socket extension representing IPC sockets. Socket units start less frequently used services on demand. Similar to xinetd . Cockpit is a great example of a socket.
Path Unit	Has a .path extension and used to delay service activation until a file system change occurs. This is typically used with printing and the cups daemon.

Listing 1. **systemctl** Man Page

```
# man systemctl
SYSTEMCTL(1)                systemctl                SYSTEMCTL(1)

NAME
  systemctl - Control the systemd system and service manager

SYNOPSIS
  systemctl [OPTIONS...] COMMAND [NAME...]
```

*Example 1. **systemctl** Demos*

Listing 2. Listing Active Services

```
[root@server0 ~]# systemctl
```

Listing 3. Listing Service Status

```
[root@server0 ~]# systemctl status bluetooth
```

Listing 4. Listing Failed Services

```
[root@server0 ~]# systemctl --failed
```



When the **systemctl** command is used with a <name> and without defining the unit type, it defaults to **Service**.



Disabled services can be started manually. However, if the service is masked, it cannot be manually or automatically started.

1.2. Controlling the Boot Process

Boot Process

1. Hardware (BIOS/UEFI)
2. Boot loader (**grub2**)
3. **Kernel** and **initramfs**
4. **systemd**

1.2.1. systemd Targets

systemd target: Set of systemd units that should be started to reach a desired state. Loosely related to run-levels.

Example 2. **systemd** Target ExamplesListing 5. Listing All **systemd** targets

```
[root@server0 ~]# systemctl list-units --type=target --all
```

Listing 6. Listing All **systemd** targets

```
[root@server0 system]# cd /usr/lib/systemd/system
[root@server0 system]# ls *target
```

Listing 7. Getting **systemd** default target

```
[root@server0 system]# systemctl get-default
```

Listing 8. Getting Targets that can be Isolated

```
[root@server0 system]# grep 'AllowIsolate=yes' *target
```



There is a huge difference between **reload** and **restart**. This difference becomes especially important when services like **firewalld** are involved.

1.2.2. Reset Root Password



Prior to RHEL7, changing the root password was easy by just doing a **runlevel 1** as it would bring you to a root prompt without requiring a password. The closest things in RHEL7 are **rescue.target** and **emergency.target** but both of these targets require the root password to login.

Change **grub** Timeout

Before lab, modify the **grub** timeout to be 10 seconds.

Listing 9. Changing **grub** timeout

```
[root@server0 ~]# vim /etc/default/grub
GRUB_TIMEOUT=10

[root@server0 ~]# grub2-mkconfig -o /boot/grub2/grub.cfg
```

1. Interrupt Boot Process
2. Select the Boot Option to edit and hit **e**

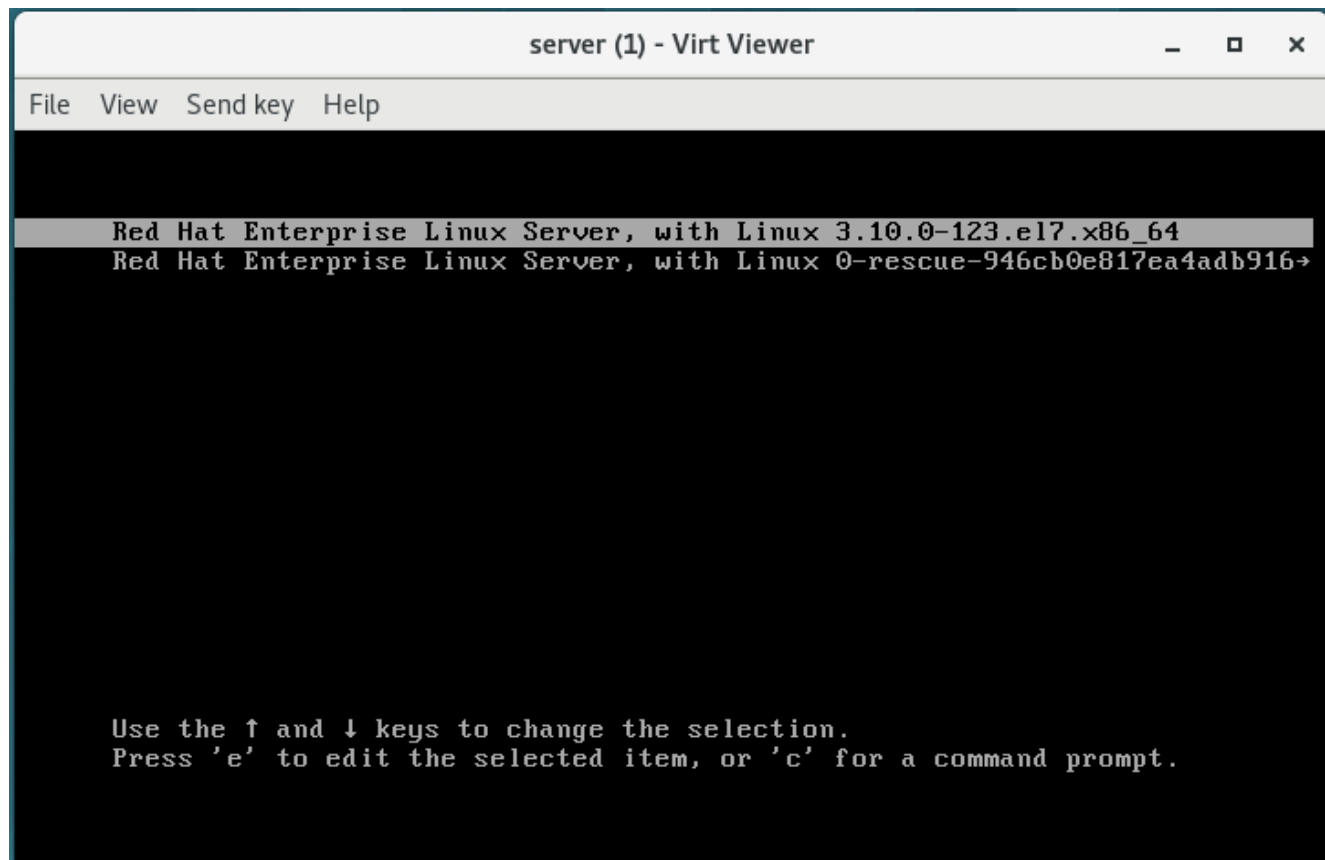


Figure 1. GRUB2 Boot Menu

3. Move cursor to the `linux16` line and append `rd.break` at the end of the line.


```

insmod part_msdos
insmod xfs
set root='hd0,msdos1'
if [ x${feature_platform_search_hint} = xy ]; then
  search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' 9bf6b9f\
7-92ad-441b-848e-0257cbb883d1
else
  search --no-floppy --fs-uuid --set=root 9bf6b9f7-92ad-441b-848e-0257\
cbb883d1
fi
linux16 /boot/vmlinuz-3.10.0-123.el7.x86_64 root=UUID=9bf6b9f7-92ad-44\
1b-848e-0257cbb883d1 ro vconsole.keymap=us console=tty0 crashkernel=auto vcon\
sole.font=latacyrheb-sun16 rd.break
initrd16 /boot/initramfs-3.10.0-123.el7.x86_64.img

Press Ctrl-x to start, Ctrl-c for a command prompt or Escape to
discard edits and return to the menu. Pressing Tab lists
possible completions.

```

Figure 2. linux16 RD.Break Modification

This starts a root shell with */* being mounted as *read-only* on */sysroot*. Next You will need to modify the filesystem to be read/write, then use **chroot** to change */sysroot* to */*. Finally, you will change the password, an set the filesystem to re-label SELinux contexts using



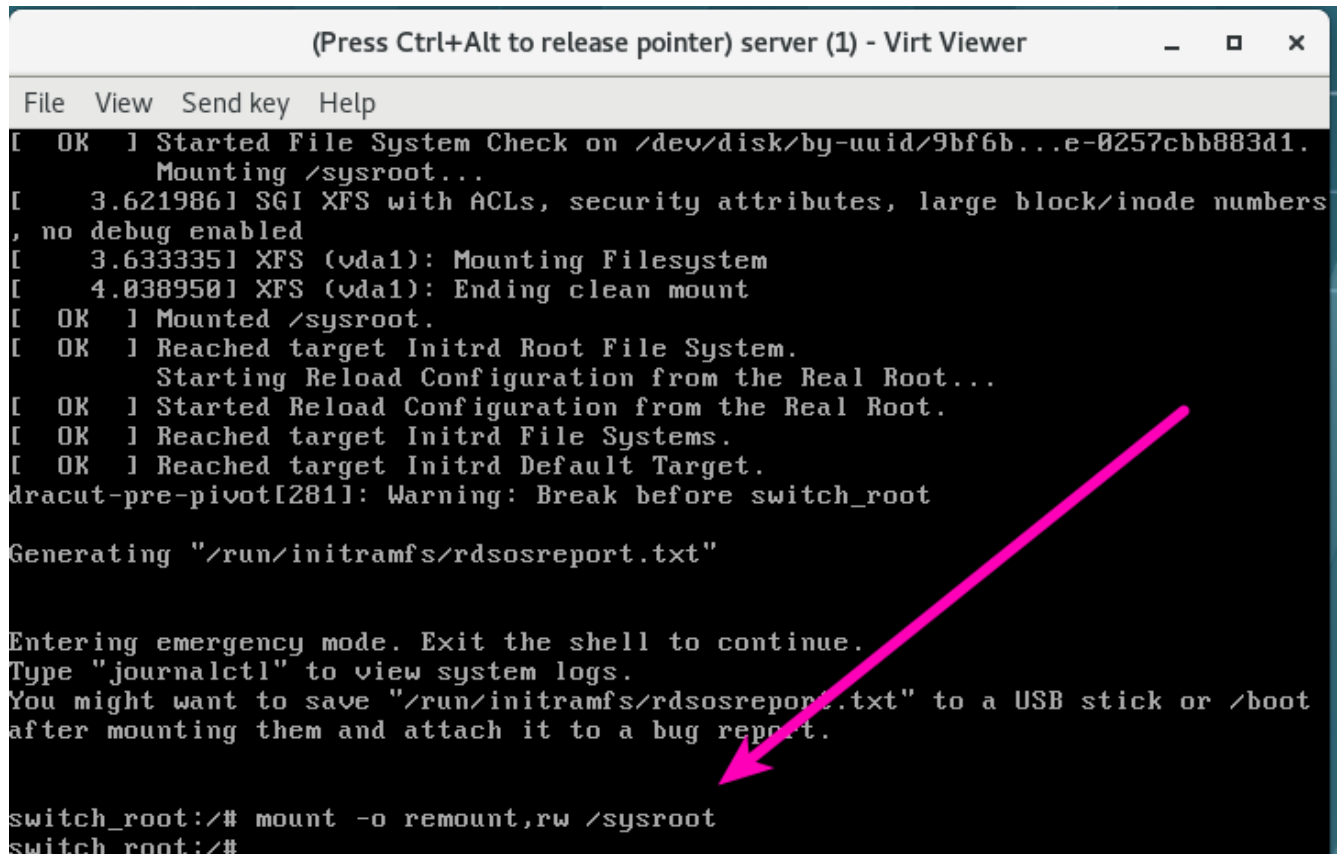
Listing 10. Set Filesystem to Relabel SELinux Contexts

```
sh-4.2# touch /.autorelabel
```

4. Remount Filesystem as Read/Write

Listing 11. Set Filesystem to be read/write

```
switch_root:/# mount -o remount,rw /sysroot
```



```
(Press Ctrl+Alt to release pointer) server (1) - Virt Viewer
File View Send key Help
[ OK ] Started File System Check on /dev/disk/by-uuid/9bf6b...e-0257cbb883d1.
Mounting /sysroot...
[ 3.621986] SGI XFS with ACLs, security attributes, large block/inode numbers
, no debug enabled
[ 3.633335] XFS (vda1): Mounting Filesystem
[ 4.038950] XFS (vda1): Ending clean mount
[ OK ] Mounted /sysroot.
[ OK ] Reached target Initrd Root File System.
Starting Reload Configuration from the Real Root...
[ OK ] Started Reload Configuration from the Real Root.
[ OK ] Reached target Initrd File Systems.
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

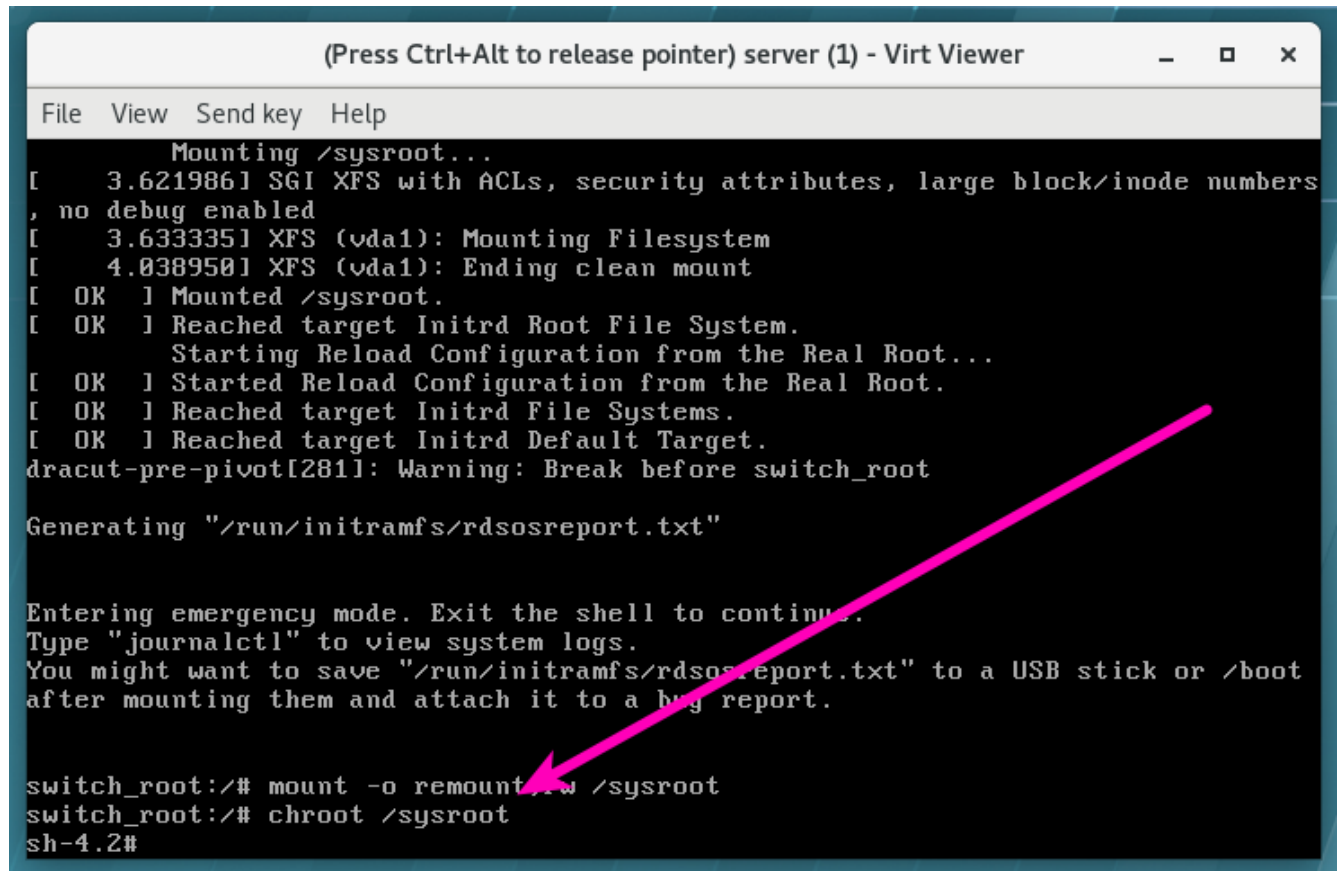
switch_root:/# mount -o remount,rw /sysroot
switch_root:/#
```

Figure 3. Remounting Filesystem

5. Use **chroot** command to change the **/sysroot** to **"/"** root filesystem

Listing 12. Set Filesystem to be read/write

```
switch_root:/# chroot /sysroot
```



```

(Press Ctrl+Alt to release pointer) server (1) - Virt Viewer
File View Send key Help
    Mounting /sysroot...
[   3.621986] SGI XFS with ACLs, security attributes, large block/inode numbers
, no debug enabled
[   3.633335] XFS (vda1): Mounting Filesystem
[   4.038950] XFS (vda1): Ending clean mount
[ OK ] Mounted /sysroot.
[ OK ] Reached target Initrd Root File System.
    Starting Reload Configuration from the Real Root...
[ OK ] Started Reload Configuration from the Real Root.
[ OK ] Reached target Initrd File Systems.
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2#

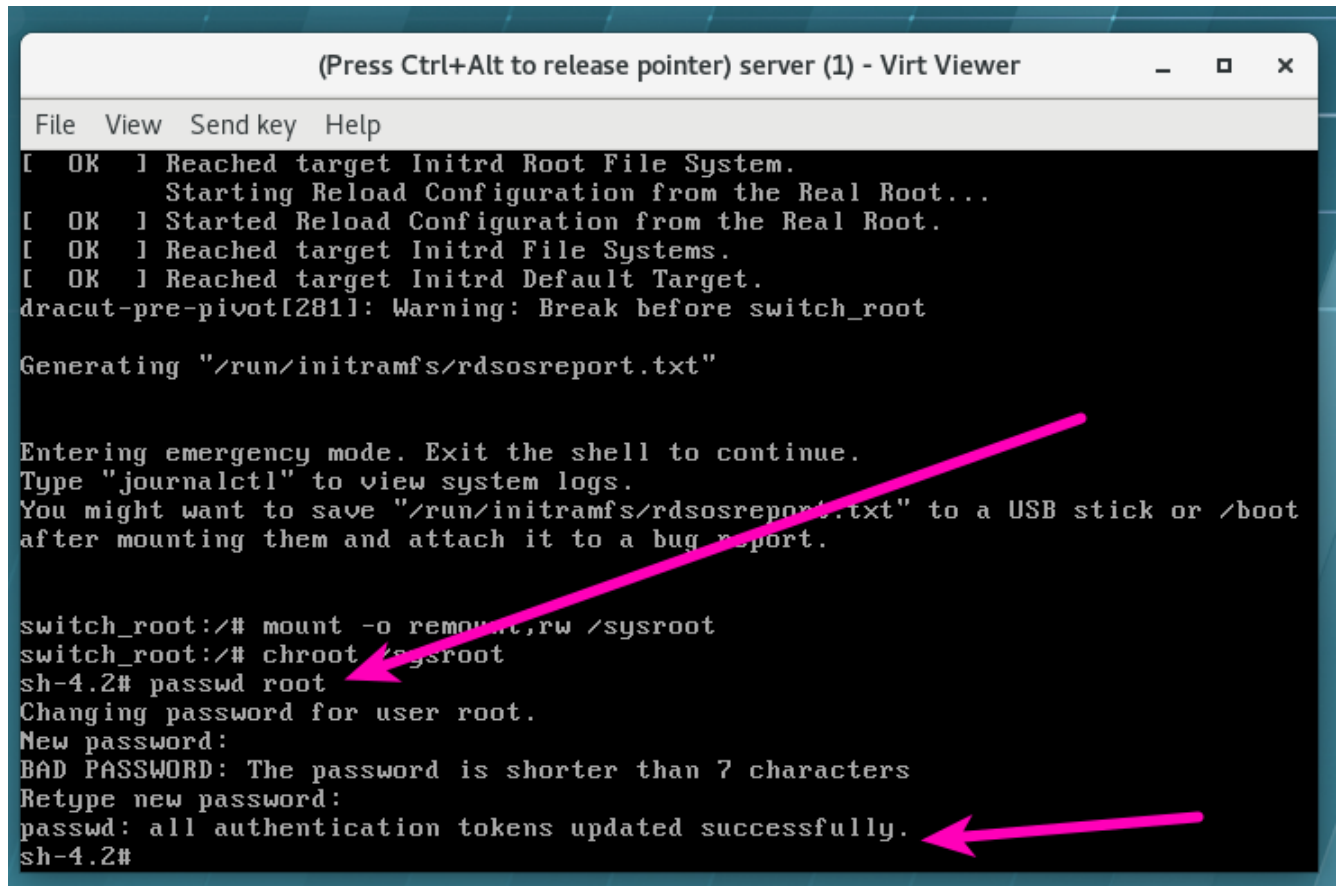
```

Figure 4. chroot of /sysroot

- Use the `passwd` command to reset the root password

Listing 13. Set root password with `passwd`

```
sh-4.2# passwd root
```



```
(Press Ctrl+Alt to release pointer) server (1) - Virt Viewer
File View Send key Help
[ OK ] Reached target Initrd Root File System.
        Starting Reload Configuration from the Real Root...
[ OK ] Started Reload Configuration from the Real Root.
[ OK ] Reached target Initrd File Systems.
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

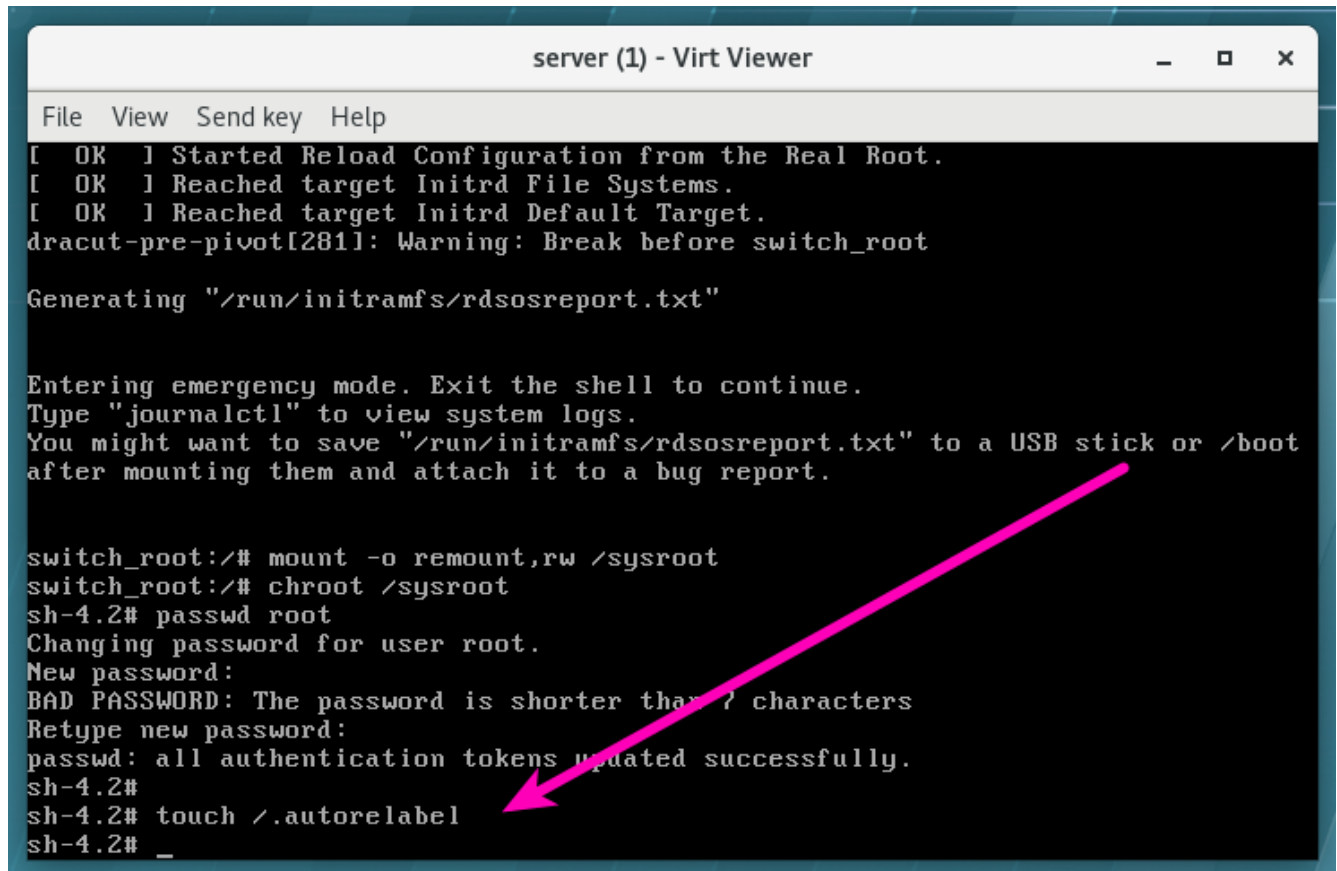
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
```

Figure 5. Setting the Root Password

7. Set the filesystem to relabel SELinux Contexts

Listing 14. Relabel SELinux Contexts

```
sh-4.2# touch /.autorelabel
```



```
server (1) - Virt Viewer
File View Send key Help
[ OK ] Started Reload Configuration from the Real Root.
[ OK ] Reached target Initrd File Systems.
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

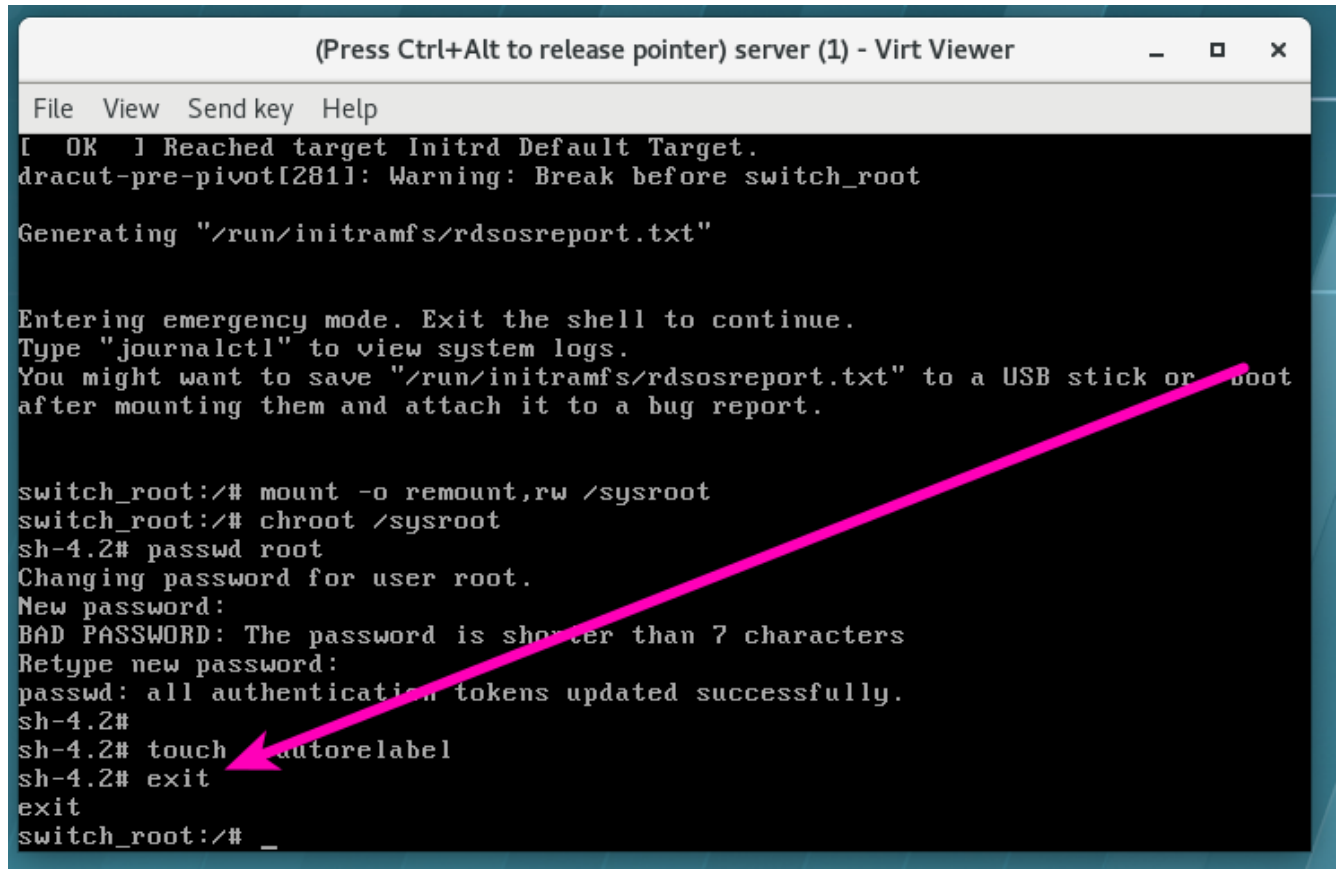
switch_root:~# mount -o remount,rw /sysroot
switch_root:~# chroot /sysroot
sh-4.2# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
sh-4.2# touch /.autorelabel
sh-4.2# _
```

Figure 6. Specify Filesystem Auto-relabel

8. Exit the **chroot** environment

Listing 15. Exit CHROOT

```
sh-4.2# exit
```



```
(Press Ctrl+Alt to release pointer) server (1) - Virt Viewer
File View Send key Help
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or boot
after mounting them and attach it to a bug report.

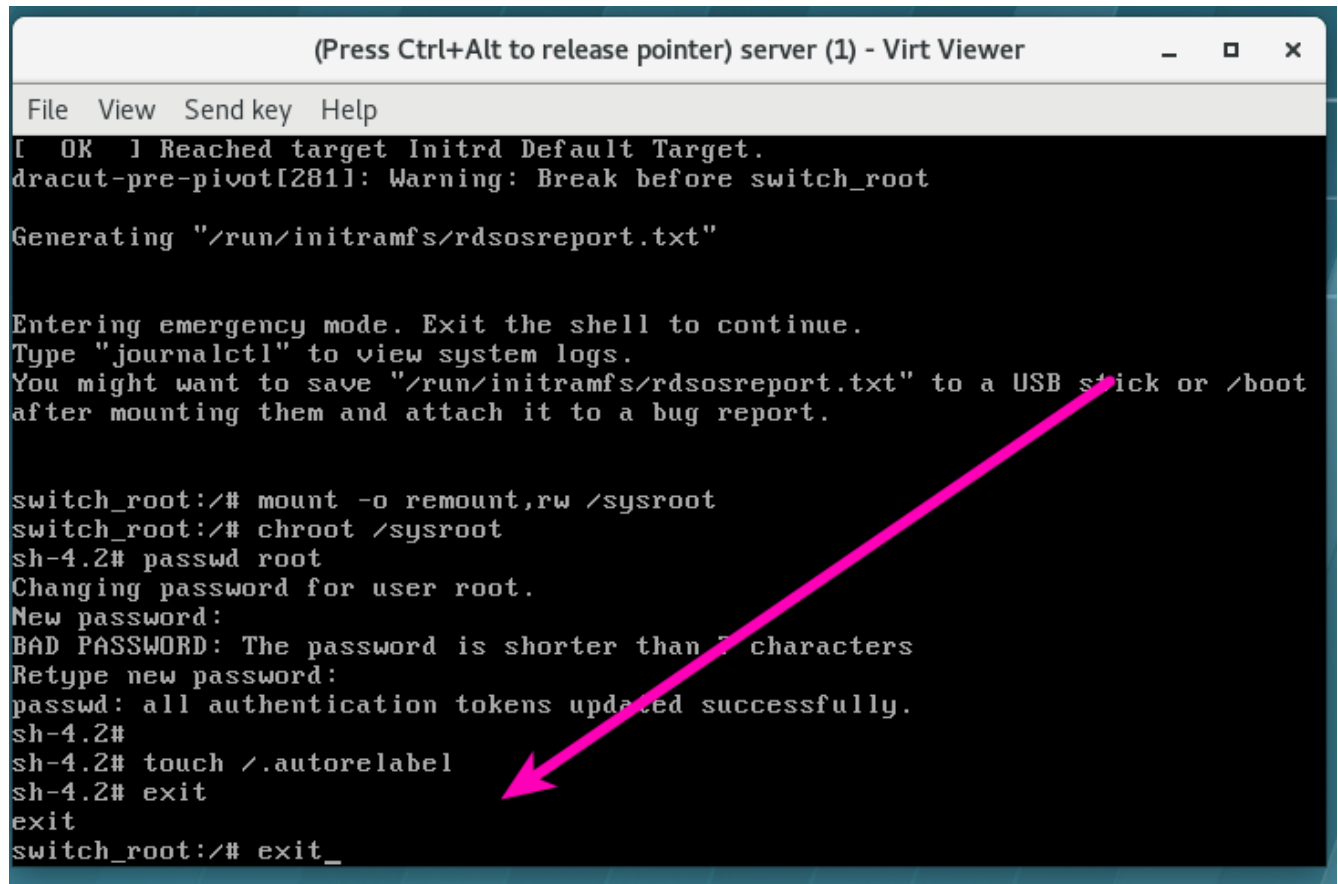
switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
sh-4.2# touch /etc/passwd
sh-4.2# exit
exit
switch_root:/# _
```

Figure 7. Exit CHROOT Environment

9. Exit the `initramfs` debug shell.

Listing 16. Exiting Debug Shell

```
switch_root:/# exit
```



```
(Press Ctrl+Alt to release pointer) server (1) - Virt Viewer
File View Send key Help
[ OK ] Reached target Initrd Default Target.
dracut-pre-pivot[281]: Warning: Break before switch_root

Generating "/run/initramfs/rdsosreport.txt"

Entering emergency mode. Exit the shell to continue.
Type "journalctl" to view system logs.
You might want to save "/run/initramfs/rdsosreport.txt" to a USB stick or /boot
after mounting them and attach it to a bug report.

switch_root:/# mount -o remount,rw /sysroot
switch_root:/# chroot /sysroot
sh-4.2# passwd root
Changing password for user root.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
passwd: all authentication tokens updated successfully.
sh-4.2#
sh-4.2# touch /.autorelabel
sh-4.2# exit
exit
switch_root:/# exit_
```

Figure 8. Exit Debug Shell

```

server (1) - Virt Viewer
File View Send key Help
[ 1051.823079] [drm] RAM header offset: 0x3ffe000
[ 1051.828877] [drm] rom modes offset 0x488 for 128 modes
[ 1051.841384] [TTM] Zone kernel: Available graphics memory: 942648 kiB
[ 1051.849365] [TTM] Initializing pool allocator
[ 1051.855575] [TTM] Initializing DMA pool allocator
[ 1051.864381] [drm] qxl: 16M of VRAM memory size
[ 1051.870784] [drm] qxl: 63M of IO pages memory ready (VRAM domain)
[ 1051.921370] [drm] main mem slot 1 [f4000000,3ffe000)
[ 1051.930533] [drm] fb mappable at 0xf4000000, size 3145728
[ 1051.937127] [drm] fb: depth 24, pitch 4096, width 1024, height 768
[ 1051.946256] fbcon: qxldrmfb (fb0) is primary device
[ 1051.980321] Console: switching to colour frame buffer device 128x48
[ 1051.987803] qxl 0000:00:02.0: fb0: qxldrmfb frame buffer device
[ 1051.987805] qxl 0000:00:02.0: registered panic notifier
[ OK ] Started Activation of DM RAID sets.
[ OK ] Reached target Local File Systems.
[ 1051.995283] [drm] Initialized qxl 0.1.0 20120117 for 0000:00:02.0 on minor 0
Starting Relabel all filesystems, if necessary...
Starting Trigger Flushing of Journal to Persistent Storage...
Starting Create Volatile Files and Directories...
Starting Security Auditing Service...
Starting Tell Plymouth To Write Out Runtime Data...
[ OK ] Reached target Encrypted Volumes.
[ 1052.040919] systemd-journald[430]: Received request to flush runtime journal from PID 1
[ OK ] Started Trigger Flushing of Journal to Persistent Storage.
[ OK ] Started Tell Plymouth To Write Out Runtime Data.
[ 1052.106253] type=1305 audit(1559518801.689:4): audit_pid=480 old=0 auid=4294967295 ses=4294967295 subj=system_u:system_r:auditd_t:s0 res=1
[ OK ] Started Show Plymouth Boot Screen.
[ OK ] Started Monitoring of LVM2 mirrors, snapshots etc. using dmcc.
[ OK ] Started udev Wait for Complete Device Initialization.
Starting Activation of DM RAID sets...
[ OK ] Started Activation of DM RAID sets.
[ OK ] Reached target Local File Systems.
Starting Relabel all filesystems, if necessary...
Starting Trigger Flushing of Journal to Persistent Storage...
Starting Create Volatile Files and Directories...
Starting Security Auditing Service...
Starting Tell Plymouth To Write Out Runtime Data...
[ OK ] Reached target Encrypted Volumes.
[ OK ] Started Trigger Flushing of Journal to Persistent Storage.
[ OK ] Started Tell Plymouth To Write Out Runtime Data.
*** Warning -- SELinux targeted policy relabel is required.
*** Relabeling could take a very long time, depending on file
*** system size and speed of hard drives.
18.7%

```

Figure 9. Reboot Process --Relabel of SELinux Context



The system should reboot and go through a SELinux relabel process to relabel the SELinux contexts on the entire filesystem. If this action does not occur, SELinux could leave the system in an undesirable state.

10. Login to the system from the login screen

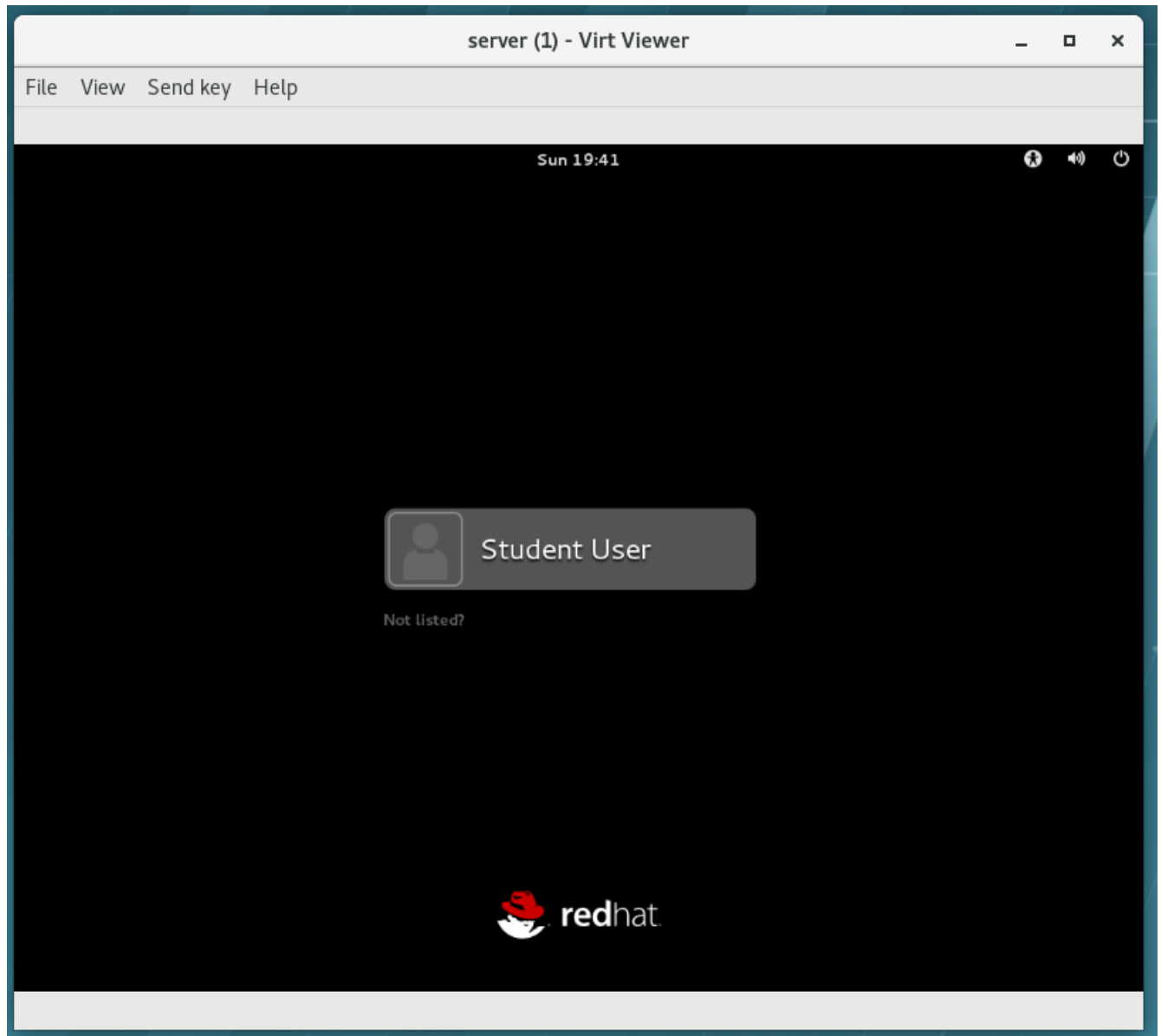


Figure 10. XWindows Login Screen

2. Managing IPv6 Networking

2.1. Review of IPv4 Networking Configuration

NetworkManager

- **Device:** Network Interface
- **Connection:** Collection of settings that are configured for a device
- **Name:** ID or name used by connection to identify it.
- **nmcli:** Utility used to create and edit connection files



Only one connection can be **Active** for any one device at a time. It is possible for multiple connections to exist for use by different devices or to allow configuration to be altered for the same device.

Example 3. nmcli Command Examples

Listing 17. Using nmcli to show device status

```
[root@server0 ~]# nmcli dev status
```

Listing 18. Using nmcli to show connections

```
[root@server0 ~]# nmcli con show
```

Listing 19. Using ip to show Connection Information

```
[root@server0 ~]# ip addr show eth0
```

Listing 20. Using nmcli-examples to show nmcli usage



```
[root@server0 ~]# man nmcli-examples

... output omitted ...

Example 6. Adding a bonding master and two slave connection profiles

$ nmcli con add type bond ifname mybond0 mode active-backup
$ nmcli con add type bond-slave ifname eth1 master mybond0
$ nmcli con add type bond-slave ifname eth2 master mybond0

... output omitted ...

Example 9. Adding an ethernet connection profile with manual IP
configuration

$ nmcli con add con-name my-con-em1 ifname em1 type ethernet ip4 192.168.100.100/24 gw4 192.168.100.1 ip4 1.2.3.4
ip6 abbe::cafe
$ nmcli con mod my-con-em1 ipv4.dns "8.8.8.8 8.8.4.4"
$ nmcli con mod my-con-em1 ipv6.dns "2001:4860:4860::8888 2001:4860:4860::8844"
$ nmcli -p con show my-con-em1
```

Example 4. Demo of nmcli to create a connection

Listing 21. Using nmcli to create a connection

```
[root@server0 ~]# lab ipv6 setup

[root@server0 ~]# ip link

[root@server0 ~]# nmcli conn add con-name static-eno1 type ethernet ifname eno1

[root@server0 ~]# nmcli con show

[root@server0 ~]# nmcli con show static-eno1

[root@server0 ~]# nmcli con mod static-eno1 ipv4.addresses '192.168.0.1/24'

[root@server0 ~]# nmcli con mod static-eno1 ipv4.method manual

[root@server0 ~]# ip a show dev eno1

[root@server0 ~]# nmcli con down static-eno1

[root@server0 ~]# nmcli con up static-eno1

[root@server0 ~]# ip addr show dev eno1

[root@server0 ~]# nmcli con show static-eno1 | grep dns

[root@server0 ~]# nmcli con mod static-eno1 ipv4.dns 192.0.2.1 ①
```

① Note: you can use a +/- to add/delete from the list. Having nothing in front of ipv4.dns will replace all values



Prior to RHEL7, the hostname was stored in `/etc/hostname`. The static hostname is now stored in `/etc/hostname` and should be changed using the `hostnamectl` command. It should also be noted that `/etc/hostname` might not exist or be created until a hostname has been set using `hostnamectl`

Example 5. Demo of `hostnamectl`

Listing 22. Using `hostname-ctl` to modify/show hostnames

```
[root@server0 ~]# hostnamectl status ①
[root@server0 ~]# cat /etc/hostname ②
[root@server0 ~]# hostnamectl set-hostname travis.example.com ③
[root@server0 ~]# hostnamectl status ④
[root@server0 ~]# cat /etc/hostname ⑤
```

- ① Shows hostname status. Note that **static** hostname isn't set
- ② Hostname file doesn't exist, static hostname not set
- ③ Sets static hostname
- ④ Shows hostname status. Note that **static** hostname is now defined
- ⑤ Output static hostname from file

2.1.1. Summary of Commands

Discuss the summary of the commands briefly

2.2. IPv6 Networking Concepts

IPv6 Address: 128-bit number represented by eight colon separated groups of four hex numbers, with each digit being referred to as a four bit nibble. Each 4-digit group is represents 16-bits of the IPv6 address.

Writing Readable Addresses

1. Always suppress leading zeros in a group
2. Use `::` to shorten as much as possible. If two runs of zeros are equal in length, shorten leftmost run.
3. Do not use `::` to shorten one group of zeros. Use `:0:` instead and only use `::` for long runs of zeros
4. Use lowercase hex numbers **a** thru **f**

When including a TCP or UDP network port after an IPv6 address, always enclose the IPv6 address in square brackets.



Listing 23. IPv6 address with TCP/UDP Port Specified

```
[2001:db8:0:10::1]:80
```

IPv6 address is **2001:db8:0:1::1/64**

Allocation from provider is **2001:db8::/48**

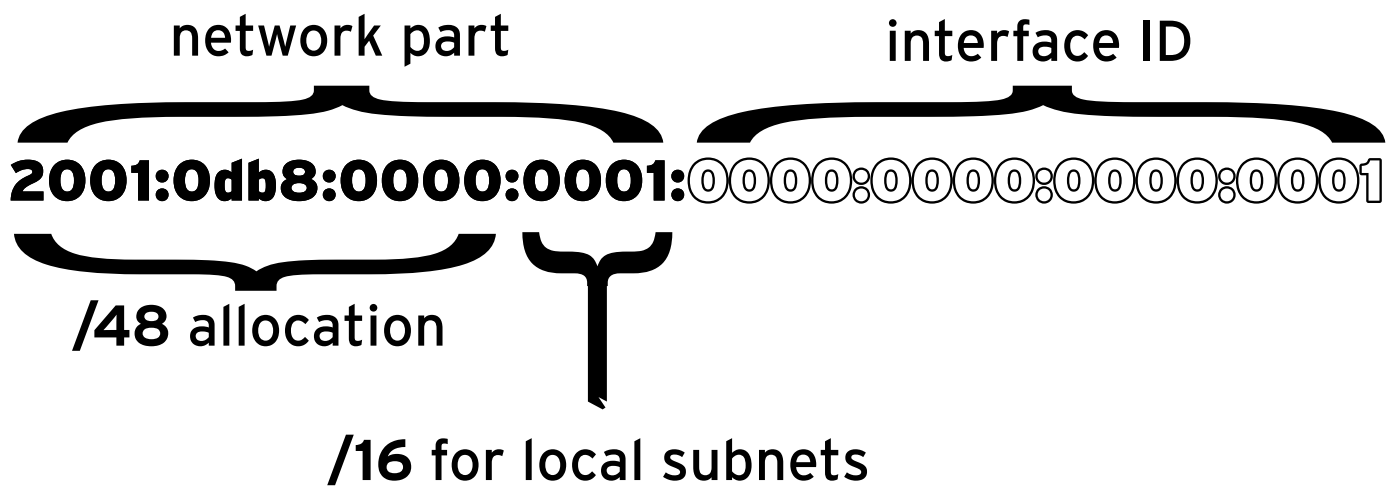


Figure 11. IPv6 Address Parts

IPv6 Subnets

- **Network Prefix:** Identifies subnet, sometimes allocated with */48* allowing */16* for local subnets.
- **Interface ID:** Identifies particular interface on the subnet
- IPv6 has standard subnet mask of */64*.

IPv6 Address Allocation

Link Local Addresses: Unroutable address used only to communicate to hosts on a specific network link. The usual procedure to convert the 48-bit MAC address to a 64-bit interface ID is to invert bit 7 of the MAC address and insert ff:fe between its two middle bytes.

Multicast: Plays larger role for IPv6 as there is no broadcast address.

IPv6 Address Configuration

Static Addressing: Can be set at will. Like with IPv4 having lowest and highest address in the subnet reserved, there are

interface IDs reserved for IPv6 that cannot be used as an address on a host.

DHCPv6 Configuration: As there is no broadcast address, DHCPv6 gets a request from a host's local link address on port 547/UDP (DHCP server link-local must be part of multicast group). DHCPv6 server sends reply to port 546/UDP on client's link-local address.

2.3. IPv6 Networking Configuration

Example 6. Configuring IPv6 Networking

Listing 24. Using nmcli to delete a connection

```
[root@server0 ~]# nmcli con del static-eno1
```

1. Show connections and Interfaces

Listing 25. Using nmcli and ip to show connections

```
[root@server0 ~]# ip link
[root@server0 ~]# nmcli con show
```

2. Define connections

Listing 26. Using nmcli to define a connection

```
[root@server0 ~]# nmcli conn add con-name ExampleIPv6 type ethernet ifname eno1
[root@server0 ~]# nmcli con show
```

3. Configure IPv6 and IPv4 on New Connection

Listing 27. Using nmcli to configure connection

```
[root@server0 ~]# nmcli con show ExampleIPv6 | grep ipv6
nmcli con mod ExampleIPv6 ipv6.addresses 'fd8b:fe2a:ab1e::c0a8:1/64 fd8b:fe2a:ab1e::c0a8:fe' ipv4.addresses '192.168.0.1/24'
[root@server0 ~]# nmcli con mod ExampleIPv6 ipv6.method manual ipv4.method manual
[root@server0 ~]# nmcli con down ExampleIPv6
[root@server0 ~]# nmcli con up ExampleIPv6
```

4. Verify Connectivity Information

Listing 28. Using `nmcli` and `ip` to verify information

```
[root@server0 ~]# ip a show eno1
[root@server0 ~]# nmcli con show ExampleIPv6
[root@server0 ~]# vim /etc/sysconfig/network-scripts/ifcfg-ExampleIPv6
```

5. Using IPv6 Tools for Troubleshooting

Listing 29. Using `ping6`, `ip`, `tracpath6`, and `traceroute -6`

```
[root@server0 ~]# ping6 fddb:fe2a:ab1e::c0a8:1
[root@server0 ~]# ping6 ff02::1%eno1 ①

[root@travis ~]# ssh root@fddb:fe2a:ab1e::c0a8:1
[root@server0 ~]# ip -6 route
[root@server0 ~]# tracpath6 fddb:fe2a:ab1e::c0a8:1
[root@server0 ~]# traceroute -6 fddb:fe2a:ab1e::c0a8:1
```

① Ping gateway over specified interface

Listing 30. Using `ss` and `netstat`

```
[root@server0 ~]# ss -A inet -n
[root@server0 ~]# netstat -46n
```



Point out Table 2.3 and Table 2.4

3. Configuring Link Aggregation and Bridging

3.1. Configuring Network Teaming

Listing 31. Reset the Systems on Practice and Labs

```
[kiosk@foundation0 ~]$ rht-vmctl reset server
```

man nmcli-examples

The **nmcli-examples** man page is a wealth of information providing examples needed to use the **nmcli** command. Often, you can copy, paste, tweak, and execute directly from the man pages.



Example 7. Adding a team master and two slave connection profiles

```
$ nmcli con add type team con-name Team1 ifname Team1 config team1-master-json.conf
$ nmcli con add type team-slave con-name Team1-slave1 ifname em1 master Team1
$ nmcli con add type team-slave con-name Team1-slave2 ifname em2 master Team1
```


Example 7. Demo of Network Teams

```
[root@server0 ~]# lab teambridge setup
[root@server0 ~]# ip link
```

1. Create Team Interface with **nmcli con add type team con-name CNAME ifname INAME [config JSON]**

Listing 32. Team Interface Creation

```
[root@server0 ~]# nmcli con add type team con-name TeamDemo ifname team0 config '{"runner": {"name": "loadbalance"}}
```

2. Configure IP Address for the Team Interface

Listing 33. Team Interface - Set IP Address

```
[root@server0 ~]# nmcli con mod TeamDemo ipv4.addresses '192.168.0.100/24'
[root@server0 ~]# nmcli con mod TeamDemo ipv4.method manual
```

3. Assign Interfaces to the Team Connection

Listing 34. Team Interface - Assigning Network Interfaces/Ports

```
[root@server0 ~]# nmcli con add type team-slave con-name TeamDemo-eno1 ifname eno1 master TeamDemo
[root@server0 ~]# nmcli con add type team-slave con-name TeamDemo-eno2 ifname eno2 master TeamDemo
```

4. Check the state of the Team Interface

Listing 35. Team Interface - Checking Device State

```
[root@server0 ~]# teamdctl team0 state
[root@server0 ~]# ping -I team0 192.168.0.254
```

5. Verify network continues to function after bringing down an interface

Listing 36. Team Interface - Testing Interface

```
[root@server0 ~]# ping -I team0 192.168.0.254 ①
[root@server0 ~]# nmcli dev dis eno1
[root@server0 ~]# teamdctl team0 state

[root@server0 ~]# nmcli dev con eno1
[root@server0 ~]# teamdctl team0 state
```

① Show example with PING in another window

3.2. Managing Network Teaming

- **teamnl**: Team Netlink Interface Tool
- **teamdctl**: Team Daemon Control Tool

Listing 37. Team Interface - Viewing Config Files

```
[root@server0 ~]# cat /etc/sysconfig/network-scripts/ifcfg-TeamDemo
[root@server0 ~]# cat /etc/sysconfig/network-scripts/ifcfg-TeamDemo-eno1
```

Listing 38. Team Interface - teamnl usage

```
[root@server0 ~]# teamnl team0 ports
[root@server0 ~]# teamnl team0 options
```



Changes made using **teamdctl** and to config files directly require the interface to be brought down/up before they go into effect.

Example 8. Demo of teamdctl to modify team connections

Listing 39. Team Interface - teamdctl usage

```
[root@server0 ~]# teamdctl team0 config dump
[root@server0 ~]# teamdctl team0 state
[root@server0 ~]# nmcli dev dis team0
[root@server0 ~]# nmcli con mod TeamDemo team.config '{"runner": {"name": "activebackup"}}' ①
[root@server0 ~]# nmcli con up TeamDemo ②
[root@server0 ~]# teamdctl team0 state
[root@server0 ~]# nmcli con up TeamDemo-eno1 ③
[root@server0 ~]# nmcli con up TeamDemo-eno2 ③
[root@server0 ~]# teamnl team0 getoption activeport
[root@server0 ~]# teamnl team0 ports
```

- ① Changes TeamDemo connection to **activebackup**
- ② Brings up team interface, but the ports are still down
- ③ Brings up team interface ports

Using JSON Files

/tmp/team.conf



```
{
  "device": "team0",
  "mcast_rejoin": {
    "count": 1
  },
  "notify_peers": {
    "count": 1
  },
  "ports": {
    "eno1": {
      "prio": -10,
      "sticky": true,
      "link_watch": {
        "name": "ethtool"
      }
    },
    "eno2": {
      "prio": 100,
      "link_watch": {
        "name": "ethtool"
      }
    }
  },
  "runner": {
    "name": "activebackup"
  }
}
```

3.3. Configuring Software Bridges

Example 9. Demo of Network Bridges

- **brctl**: Ethernet Bridge Administration command

```
[root@server0 ~]# lab teambridge setup
```

1. Create a Connection

Listing 40. Creating BridgeDemo

```
[root@server0 ~]# nmcli con add type bridge con-name BridgeDemo ifname br0
```

2. Add Interfaces

Listing 41. Creating BridgeDemo - Adding Interfaces

```
[root@server0 ~]# nmcli con add type bridge-slave con-name BridgeDemo-eno1 ifname eno1 master BridgeDemo  
[root@server0 ~]# nmcli con add type bridge-slave con-name BridgeDemo-eno2 ifname eno2 master BridgeDemo
```

3. Configure BridgeDemo

Listing 42. Configuring BridgeDemo - Adding Network Settings

```
[root@server0 ~]# nmcli con mod BridgeDemo ipv4.addresses 192.168.0.100/24 ipv4.method manual  
[root@server0 ~]# nmcli con down BridgeDemo  
[root@server0 ~]# nmcli con up BridgeDemo
```

4. Verify BridgeDemo Settings

Listing 43. BridgeDemo - Verification

```
[root@server0 ~]# cat /etc/sysconfig/network-scripts/ifcfg-BridgeDemo  
[root@server0 ~]# cat /etc/sysconfig/network-scripts/ifcfg-BridgeDemo-eno1  
[root@server0 ~]# cat /etc/sysconfig/network-scripts/ifcfg-BridgeDemo-eno2  
[root@server0 ~]# brctl show
```

4. Network Port Security

4.1. Managing Firewall



Preventing multiple firewall services from running

```
[root@server0 ~]# for SERVICE in iptables ip6tables ebtables; do  
> systemctl mask ${SERVICE}.service  
> done
```



Ways to Manage **firewalld**

- Using command line tool **firewall-cmd**
- Using graphical tool **firewall-config**
- Using configuration files in **/etc/firewalld**

Example 10. FirewallD Demos

Listing 44. Checking Service Status

```
[root@server0 ~]# systemctl status firewalld.service
```

Listing 45. Listing Firewall Rules

```
[root@server0 ~]# firewall-cmd --list-all
```

Listing 46. Listing Firewall Active Zones

```
[root@server0 ~]# firewall-cmd --get-active-zones
```

Listing 47. Investigating Zones and Config Files

```
[root@server0 ~]# cd /etc/firewalld/
[root@server0 firewalld]# tree
[root@server0 firewalld]# grep DefaultZone firewalld.conf
[root@server0 ~]# firewall-cmd --set-default-zone=dmz
[root@server0 firewalld]# grep DefaultZone firewalld.conf

[root@server0 ~]# firewall-cmd --permanent --zone=work --add-source=172.25.X.0/24
[root@server0 ~]# firewall-cmd --zone=work --list-all
[root@server0 firewalld]# tree
[root@server0 firewalld]# cat zones/work.xml
```



Unless overridden by **NetworkManager** the default zone for any network interface is the **public** zone.



It is a good idea to test firewall rules by applying only to the run-time firewall (omitting the **--permanent**) option. Furthermore, the **--timeout=<TIMEINSECONDS>** can be used to prevent accidental lockout by having the rule disappear after **<TIMEINSECONDS>**.

Permanent vs. Runtime Firewall Rules

It is important to note the difference between **reload** and **restart** for the **firewalld** service. It is common practice to use two separate **firewalld** commands to make the firewall rules apply to the running firewall as well as permanent configuration.



```
firewall-cmd --add-port=443/tcp
firewall-cmd --permanent --add-port=443/tcp
```

Firewalld Services

In addition to adding firewall rules by ports, it is also possible to add firewall rules using pre-defined services. Several applications and services have defined firewall rules in a `<ServiceName>.xml` file. This allows the service to be added to the firewall. Adding a firewall rule with a service involves adding one or more firewall rules with the `firewalld` command.

Listing 48. Adding firewall rules with a Service Definition

```
firewall-cmd --add-service=https
```

The `firewall-cmd` can be used to list services and expand rich rules with the `firewall-cmd --info -service=<Service_Name>` syntax.

Listing 49. https Service Definition

```
[root@foundation0 ~]# firewall-cmd --info-service=https
```

Listing 50. freeipa-ldap Service Definition

```
[root@foundation0 ~]# firewall-cmd --info-service=freeipa-ldap
```



RHEL 7.0 didn't have the full `firewall-cmd` functionality that is available in later RHEL7 versions.

Viewing and Creating firewalld Services

Firewalld service files are located in two places:

- `/usr/lib/firewalld/services`: Contains system and service defined services. Generally provided with default installation of packages.
- `/etc/firewalld/services`: Contains custom `firewalld` service definitions.

Listing 51. Viewing Kerberos Service

```
[root@server0 /]# cat /usr/lib/firewalld/services/kerberos.xml
```

Listing 52. Viewing freeipa-ldap Service

```
[root@foundation0 ~]# cat /usr/lib/firewalld/services/freeipa-ldap.xml
```



Example 11. Creating Custom Firewall Service

1. Copy or create a file into `/etc/firewalld/services`

Listing 53. Copy of SSH Service File to Use as Template

```
[root@server0 ~]# cp /usr/lib/firewalld/services/ssh.xml /etc/firewalld/services/travis.xml
```

2. Modify Custom Service File

Listing 54. `cat /etc/firewalld/services/travis.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<service>
  <short>Travis</short>
  <description>The Travis service uses SSH on a special port.</description>
  <port protocol="tcp" port="2222"/>
</service>
```

3. Attempt to use Custom Service File

```
[root@server0 ~]# firewall-cmd --add-service=travis
[root@server0 ~]# firewall-cmd --reload
[root@server0 ~]# firewall-cmd --add-service=travis --permanent
[root@server0 ~]# firewall-cmd --reload
```



More **firewalld** information around adding a service can be found: <https://firewalld.org/documentation/howto/add-a-service.html>

4.2. Managing Rich Rules

Rich Rules: **firewalld** rich rules allow expressive language to build custom firewall rules not covered by basic **firewalld** syntax.



- **Direct Rules:** Direct rules allow administrators to insert hard-coded rules into zones managed by **firewalld**. This is a means of exposing the **netfilter** features provided by the Linux kernel. Documentation for direct rules is available in **firewall-cmd** and **firewalld.direct** man pages.

Direct Rules

Listing 55. Rich Rule Help

```
[root@server0 ~]# man firewalld.richlanguage
```

Rule Ordering

Basic ordering of firewall rules inside a zone is the same for all zones.

1. Port forwarding and masquerading rules for that zone
2. Logging rules set for the zone
3. Deny rules set for the zone
4. Allow rules set for the zone

The first matched rule wins. IF there is no match, the default is to typically deny.

Example 12. Rich Rule Example

1. Install and Start HTTP Services

```
[root@server0 ~]# yum install httpd
[root@server0 ~]# systemctl start httpd.service
```

2. Create Rich Rule for Firewall and Load it

```
[root@server0 ~]# firewall-cmd --permanent --add-rich-rule='rule family=ipv4 source address=172.25.0.10/32 service name="http" log
level=notice prefix="ACCESS TO HTTP" limit value="5/h" accept' ①
[root@server0 ~]# firewall-cmd --reload
```

① Rate limits to 5 messages per hour. The **limit** doesn't limit the number of connections, rather it limits what gets logged.

3. Watch Tail and try to connect

```
[root@server0 ~]# echo "This is a test" > /var/www/html/index.html
[root@server0 ~]# tail -f /var/log/messages

[root@desktop0 ~]# curl http://server0.example.com; curl http://server0.example.com ; curl http://server0.example.com; curl
http://server0.example.com ; curl http://server0.example.com; curl http://server0.example.com ; curl http://server0.example.com;
curl http://server0.example.com ; curl http://server0.example.com; curl http://server0.example.com
```

4. Look at Config File

```
[root@server0 firewallld]# cd /etc/firewalld
[root@server0 firewallld]# vim zones/public.xml
```

4.3. Masquerading and Port Forwarding

firewalld supports two types of Network Address Translation (NAT). The two supported types are **Masuerading** and **Port**

Forwarding. Both of the NAT forms modify certain packet aspects before sending it on.



Masquerading in RHEL7

<https://access.redhat.com/solutions/1518903>

1. Put each interface in a separate zone

```
# nmcli connection modify eth0 connection.zone work
# nmcli connection modify eth1 connection.zone public
# nmcli connection reload
```

2. Enable masquerading on the outbound (WAN) interface

```
# firewall-cmd --add-masquerade --zone=public --permanent
# firewall-cmd --reload
```

Example 13. Demo of Masquerading and Port Forwarding

1. Check current firewall

```
[root@server0 ~]# firewall-cmd --list-all
public (default, active)
  interfaces: eth0
  sources:
  services: dhcpv6-client ssh
  ports:
  masquerade: no
  forward-ports:
  icmp-blocks:
  rich rules:
```

2. Add Rich Rule to forward from Desktop on port 8080 to SSH on port 22.

```
[root@server0 ~]# sudo firewall-cmd --permanent --add-rich-rule 'rule family=ipv4 source address=172.25.0.10/32 forward-port port=8080 protocol=tcp to-port=22'

[root@server0 ~]# firewall-cmd --reload

[root@server0 ~]# firewall-cmd --list-all
```

4.3.1. Masquerading

- **Masquerading:** Forward packets not addressed to itself to the intended recipient while changing the source address of the packets that go through the public IP address. The firewall performs address translation and modifies the destination address to address of original host. Masquerading is a form of NAT (Network Address Translation).

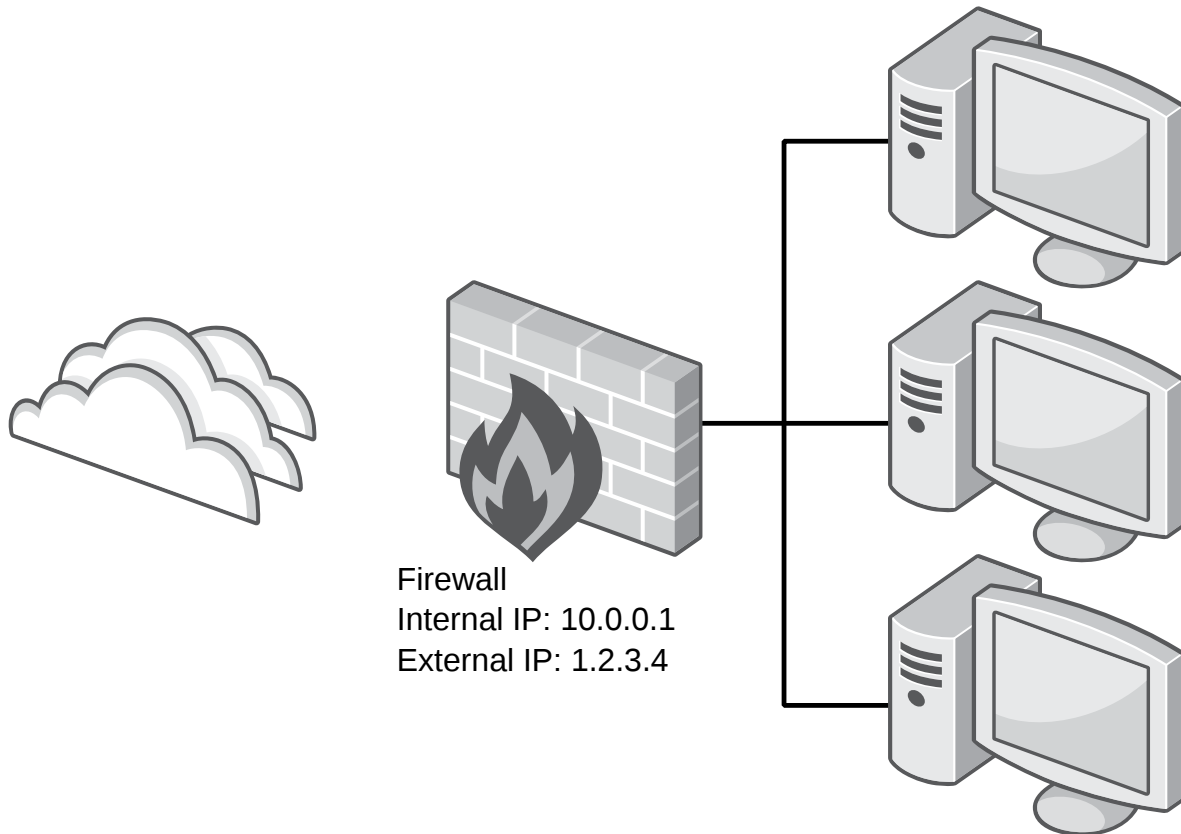


Figure 12. IP NAT



Masquerading can only be used with IPv4, not with IPv6.

4.3.2. Port Forwarding

- **Port Forwarding:** With port forwarding, traffic to a single port is forwarded either to a different port on the same machine, or to a port on a different machine.



Port forwarding is often used with masquerading as back-and-forth communications must be masqueraded through the firewall.

4.4. Managing SELinux Port Labeling

In addition to file and process labeling, **SELinux** controls network traffic by labeling network ports.

semanage port Syntax

The **semanage port** command can be used with the **-a**, **-d**, or **-m** options to either add, delete, or modify the SELinux port label.

Listing 56. **semanage port** Basic Syntax

```
semanage port -[a,d,m] -t port_label -p tcp|udp PORTNUMBER
```

Some helpful resources

```
[root@server0 ~]# man semanage-port
```



SELinux has a GUI tool that can be installed with **policycoreutils-gui** package.

Example 14. SELinux Port Labeling

Listing 57. Port 22 for SSH

```
[root@server0 ~]# semanage port -l | grep 22$
[root@server0 ~]# semanage port -l | grep ssh
ssh_port_t                tcp          22
```

Listing 58. Adding a Port Label

```
[root@server0 ~]# semanage port -a -t ssh_port_t -p tcp 2222
[root@server0 ~]# semanage port -l | grep ssh
```

Listing 59. Deleting a Port Label

```
[root@server0 ~]# semanage port -d -t ssh_port_t -p tcp 2222
[root@server0 ~]# semanage port -l | grep ssh
```



To get all **man** pages and help files, it is necessary to install the **selinux-policy-devel** package and perform a **mandb** command. This will update the **man** database and allow for the additional SELinux man pages to be searched by using:

```
[root@server0 ~]# man -k _selinux | grep ssh
```



SELinux has a graphical utility **system-config-selinux** provided by the **policycoreutils-gui** package.



In addition to network port labeling, SELinux provides booleans that can further allow/restrict access to systems. This is beyond the scope of this course and module, but will be covered more in depth in the RH415 Security course.

Listing 60. SSH SELinux Boolean Values

```
[root@server0 ~]# getsebool -a | grep ssh
```

5. Managing DNS for Servers

5.1. DNS Concepts

- **Domain Name System (DNS):** Hierarchical naming system that is a directory of hosts and resources. Each level of DNS is marked by the "." as top level. Typically, top level domains are **com**, **net**, and **org**.
- **Domain:** Collection of resources ending in a common name representing an entire DNS sub-tree.
- **Sub Domain:** A subtree of another domain. It is essentially a domain within a domain.
- **Zone:** Portion of domain for which a particular nameserver is directly responsible otherwise referred to as authoritative. This can be an entire domain or a portion of the domain with some or all subdomains.



The **host** and **dig** commands allow manual lookup of DNS names in addition to the standard **nslookup** command.

DNS Servers listed in **/etc/resolv.conf**

DNS Lookups and Answers

- **Local authoritative data:** Server providing response is authoritative for data being provided.
- **Local cached non-authoritative data:** Server is not DNS authority but recently obtained a record from a previous DNS lookup/query.
- **Remote non-authoritative data via recursion:** DNS isn't authoritative and it doesn't possess the record in cache. DNS server then goes to a root nameserver to retrieve the information, and attempts to find an authoritative server.

DNS Resource Record Types

- **A record:** Maps host name to IPv4 address
- **AAAA record:** Maps host name to IPv6 address (sometimes called "quad-A" record)
- **CNAME (canonical name) record:** Aliases one hostname to another name which corresponds to **A** or **AAAA** records.
- **MX (mail exchange) records:** Maps domain name to *mail exchange* which accepts email for that name
- **NS (name server) Record:** Maps domain name to DNS name server that is authoritative for its DNS zone.
- **PTR Record:** Maps IPv4 or IPv6 to a host name. Used for *reverse DNS*
- **SOA (start of authority) record:** Provides information on how a DNS zone works
- **SRV (service) record:** Used to locate hosts supporting a particular service for a domain
- **TXT (text) record:** Use to map a name to arbitrary human readable text.

Typical Host Contains

- One or more **A/AAAA** records
- A **PTR** record for each IP address for reverse lookup
- One or more **CNAM** records for alternate names

Typical DNS Zone

- Single **SOA** record
- **NS** record for each authoritative name server
- One or more **MX** records
- One or more **TXT** records for SPF or Site verification
- One or more **SRV** records to locate services in the domain

Example 15. Examples of DNS Record types and Lookups

Listing 61. A record check

```
[root@server0 ~]# host -v -t A example.com
```

Listing 62. Record check revealing CNAME

```
[root@server0 ~]# host -v -t A ipa-ca.server0.example.com
```

Listing 63. MX record check

```
[root@server0 ~]# host -v -t MX example.com
```

Listing 64. NS record check

```
[root@server0 ~]# host -v -t NS example.com
```

Listing 65. PTR record check

```
[root@server0 ~]# host -v -t PTR 172.25.0.10
```

Listing 66. SOA record check

```
[root@server0 ~]# host -v -t SOA example.com
```

Listing 67. SRV record check

```
[root@server0 ~]# host -v -t SRV _ldap._tcp.server0.example.com
```

Listing 68. TXT record check

```
[root@server0 ~]# host -v -t TXT example.com
```

Listing 69. dig command examples

```
[root@server0 ~]# dig example.com ANY
[root@server0 ~]# dig server0.example.com
[root@server0 ~]# dig -t mx example.com ①
[root@server0 ~]# dig mx example.com +short ②
```

- ① Should be noted, the "-t MX" specifies MX record. The result is the same without the "-t".
- ② You can use "+short" to get only the record searched for returned.

5.2. Configuring a Caching Nameserver

Commands/Packages Discussed

- **unbound**: Unbound DNS validating resolver
- **unbound-checkconf**: Check unbound configuration file for errors
- **unbound-control**: Unbound remote server control utility
- **Caching Nameserver**: Stores DNS query results in a local cache and records get removed when TTL expires. It is common to have DNS caching locally to reduce DNS traffic across the WAN.
- **DNSSEC Validation**: Provides a level of security on DNS servers to assist in avoiding tampering and spoofing. When used, it enables a caching nameserver to verify authenticity and integrity of records by providing validation on records prior to be placed in the cache.

Example 16. Demo of Installing Caching Nameserver with Unbound

1. Install **unbound**Listing 70. Installing **unbound**

```
[root@server0 ~]# yum install unbound
```

2. Starting **unbound.service**Listing 71. Starting **unbound**

```
[root@server0 ~]# systemctl enable unbound.service
[root@server0 ~]# systemctl start unbound.service
```

3. Configure Firewall

Listing 72. Configure Firewall for DNS

```
[root@server0 ~]# firewall-cmd --permanent --add-service=dns
[root@server0 ~]# firewall-cmd --reload
```

4. Configure **unbound** by modifying **/etc/unbound/unbound.conf**Listing 73. Configuring **unbound**

```
[root@server0 ~]# cat /etc/unbound/unbound.conf

interface: 0.0.0.0 ①
access-control: 172.25.0.0/24 allow ②

forward-zone: ③
  name: "."
  forward-addr: 172.25.254.254

domain-insecure: example.com ④
```

① Search for **interface**

② Search for **access-control**

③ Search for **forward-zone**

④ Search for **domain-insecure**

5. Restart the **Unbound** serviceListing 74. Restart **unbound** to load configuration

```
[root@server0 ~]# systemctl restart unbound.service
```

Example 17. Demo: **unbound** in action

Listing 75. Using **unbound-control**

```
[root@server0 ~]# unbound-control dump_cache ①
```

① Currently shows no entries

Listing 76. Using **dig** to force cache entries in **unbound**

```
[root@server0 ~]# dig @server0.example.com desktop0.example.com ①
[root@server0 ~]# dig @server0.example.com desktop1.example.com
[root@server0 ~]# unbound-control dump_cache ②
```

① Forces lookup on cache host and retrieves entries not in cache

② Shows cached entries

Listing 77. Using **unbound-control** to flush records

```
[root@server0 ~]# unbound-control dump_cache
[root@server0 ~]# unbound-control flush desktop1.example.com
[root@server0 ~]# unbound-control dump_cache
[root@server0 ~]# dig @server0.example.com desktop1.example.com
[root@server0 ~]# unbound-control dump_cache
[root@server0 ~]# unbound-control flush_zone example.com
[root@server0 ~]# unbound-control dump_cache
```

5.3. DNS Troubleshooting

Commands Used

- **dig**: DNS lookup utility
- **getent**: Get entries from NSS libraries
- **gethostip**: Convert IP address into various formats

Name Resolution

Name resolution order is specified in **/etc/nsswitch.conf** file. Typically, it is set to use **files** before **dns**. The **files** tells the system to use the **/etc/hosts** file prior to attempting to use DNS resolution.



By default **DNS** queries with **dig** use the UDP protocol. Specifying **+tcp** with the **dig** command will force the DNS lookup to be performed over the TCP protocol.

The **syslinux** package provides the **getent** and **gethostip** commands.



It is possible that `/etc/nsswitch.conf` can override name resolution before DNS is used.

```
[root@server0 ~]# grep hosts /etc/nsswitch.conf
```

Listing 78. Using **dig** to troubleshoot DNS

```
[root@server0 ~]# dig -t A example.com
```

Table 3. DNS Return Codes

Code	Meaning
SERVFAIL	The nameserver encountered a problem while processing the query.
NXDOMAIN	The queried name does not exist in the zone.
REFUSED	The nameserver refused the client's DNS request due to policy restrictions.

DNS Return Codes

DNS return codes can provide some indication of what might be wrong when attempting DNS lookups.

- **SERVFAIL**: Indicates failure of DNS server to communicate with getting an authoritative answer for the name being queried. Using the **+trace** option with **dig** can provide additional helpful information
- **NXDOMAIN**: No records were found with the name queried. If it is returned from a non-authoritative server, it is possible that the cache is bad for the name.
- **REFUSED**: Indicates policy restriction preventing the DNS server from responding to the request. This can also be caused by sending query to wrong DNS server or a DNS server misconfiguration.

Common DNS Issues

- **Outdated Cached Data: NOERROR** indicates that there were no errors, but doesn't mean there aren't DNS issues. It is possible to receive incorrect results from cached data.
- **Responses to nonexistent records**: If record removed from a zone and response is received, this could be coming from cached data or from a wildcard in authoritative DNS server's zone record.
- **Non-FQDN error**: Hostnames not expressed with FQDN will be expanded to FQDN by appending the name to zone. To indicate name is FQDN in a zone file, there **must** be a "." at the end of the name.
- **Looping CNAME Records**: **CNAME** records pointing to **CNAME** records should be avoided.
- **Missing PTR Records**: Absence of PTR records prevents reverse lookups and can result in DNS issues. Some services, including **SSH** perform reverse lookups

- **Round-robin DNS:** Often used for load-balancing to distribute loads across hosts. When using round-robin DNS, it is important to understand the ordering of the queries and records

By default, SSSHd uses reverse DNS lookup. This can cause slowness with SSH connections when lookups cannot be performed.



```
[root@server0 ~]# grep -i dns /etc/ssh/sshd_config  
#UseDNS yes
```

The **UseDNS** line in **/etc/ssh/sshd_config** can be uncommented and changed to **no** to prevent the SSSHd daemon from using reverse lookups.

6. Configuring Email Transmission

Commands Introduced

- **postconf**: Postfix configuration utility

6.1. Configuring Send-only Email Service

`/usr/sbin/sendmail` is the application acting as a **Sendmail** server. In RHEL7, **Postfix** provides the **Sendmail** application.

Null Client: Machine running a local mailserver that forwards all e-mail to an outbound mail relay server for eventual deliver. A NULL client doesn't accept local delivery for any messages.



The main configuration file of the **Postfix** mail server is `/etc/postfix/main.cf`

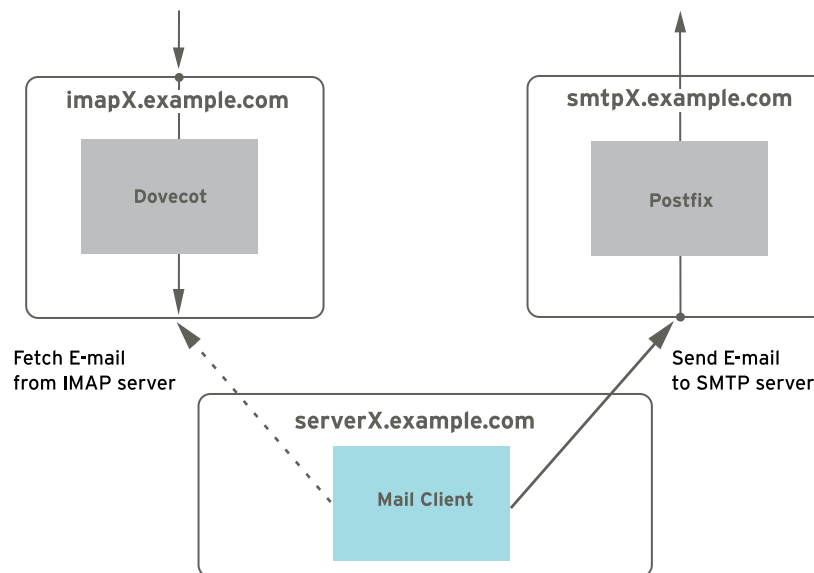


Figure 13. E-mail Client Communication

Table 4. Postfix Important Configuration Settings

Setting	Purpose
<code>inet_interfaces</code>	<p>Controls which network interfaces Postfix listens on for incoming and outgoing messages. One or more host names and IP addresses, separated by white space, can be listed.</p> <p>Default: <code>inet_interfaces = localhost</code></p>

Setting	Purpose
myorigin	<p>Rewrite locally posted email to appear to come from this domain. This helps ensure responses return to the correct domain the mail server is responsible for.</p> <p>Default: myorigin = \$myhostname</p>
relayhost	<p>Forward all messages to the mail server specified that are supposed to be sent to foreign mail addresses. Square brackets around the host name suppress the MX record lookup.</p> <p>Default: relayhost =</p>
mydestination	<p>Configure which domains the mail server is an end point for. Email addressed to these domains are delivered into local mailboxes.</p> <p>Default: mydestination = \$myhostname, localhost.\$mydomain, localhost</p>
local_transport	<p>Determine how email addressed to \$mydestination should be delivered. By default, set to local:\$myhostname, which uses the local mail delivery agent to deliver incoming mail to the local message store in <code>/var/spool/mail</code>.</p> <p>Default: local_transport = local:\$myhostname</p>
mynetworks	<p>Allow relay through this mail server from a comma-separated list of IP addresses and networks in CIDR notation to anywhere, without further authentication.</p> <p>If the mynetworks setting is not explicitly set in <code>/etc/postfix/main.cf</code>, it will be filled automatically using the setting for <code>mynetworks_style</code>.</p> <p>It is recommended that a mynetworks setting gets added manually, or <code>mynetworks_style</code> is set to <code>host</code>.</p> <p>Default: mynetworks = 127.0.0.0/8 [::1]/128</p>



The **postfix** service requires a **reload** or **restart** after the changes have been made to `/etc/postfix/main.cf`.

*Example 18. Demo: Getting Postconf values**Listing 79. Listing all postconf values*

```
[root@server0 ~]# postconf
```

Listing 80. Query specific postconf value

```
[root@server0 ~]# postconf inet_interfaces myhostname
```

```
[root@server0 ~]# postconf inet_interfaces myorigin ①
```

- ① The output of this shows **\$myhostname** to be the value for the **myorigin** parameter. This references the variable **myhostname**. This allows for the value to be updated once instead of multiple locations.

Listing 81. Edit postconf value

```
[root@server0 ~]# postconf -e 'myorigin = example.com'
```

```
[root@server0 ~]# postconf myorigin
```

*SETUP Of Environment for DEMO**Listing 82. Setup Server for Mail Demo*

```
[root@server0 ~]$ lab smtp-nullclient setup
```

Listing 83. Setup Desktop for Mail Demo

```
[root@desktop0 ~]$ lab smtp-nullclient setup
```



Example 19. DEMO - Configuring Postfix NULL Client

```
[root@server0 ~]# postconf -e "relayhost=[smtp0.example.com]"
```

```
[root@server0 ~]# postconf -e "inet_interfaces=loopback-only"
```

```
[root@server0 ~]# postconf -e "mynetworks=127.0.0.0/8 [::1]/128"
```

```
[root@server0 ~]# postconf -e "myorigin=desktop0.example.com"
```

```
[root@server0 ~]# postconf -e "mydestination="
```

```
[root@server0 ~]# postconf -e "local_transport=error: local delivery disabled"
```

```
[root@server0 ~]# systemctl restart postfix
```

```
[root@server0 ~]# mail -s "server0 null client test message for demo." student@desktop0.example.com
This is a null client demo test
.
EOT
```

7. Providing Remote Block Storage

7.1. iSCSI Concepts

- **iSCSI**: Internet Small Computer System Interface (iSCSI) is a TCP/IP-based protocol for emulating a SCSI high-performance local storage bus over IP networks



SCSI should also be implemented on cabling that is independent of standard LAN traffic.

- **Backstore**: Logical volume serving as underlying storage to iSCSI servers emulating a SCSI disk. (note can be file backed and other options as well). Used to create the LUNs for the target device

Table 5. iSCSI Components

Term	Description
ACL	An Access Control List (entry), an access restriction using the node IQN (commonly the iSCSI Initiator Name) to validate access permissions for an initiator.
discovery	Querying a target server to list configured targets. Target use requires an additional access steps (see login).
initiator	An iSCSI client, typically available as software but also implemented as iSCSI HBAs. Initiators must be given unique names (see IQN).
IQN	<p>An iSCSI Qualified Name, a worldwide unique name used to identify both initiators and targets, in the mandated naming format:</p> <p>iqn.YYYY-MM.com.reversed.domain[:optional_string]</p> <ul style="list-style-type: none"> • iqn — Signifying that this name will use a domain as its identifier. • YYYY-MM — The first month in which the domain name was owned. • com.reversed.domain — The reversed domain name of the organization creating this iSCSI name. • optional_string — An optional, colon-prefixed string assigned by the domain owner as desired while remaining worldwide unique. It may include colons to separate organization boundaries.

Term	Description
login	Authenticating to a target or LUN to begin client block device use.
LUN	A Logical Unit Number, numbered block devices attached to and available through a target. One or more LUNs may be attached to a single target, although typically a target provides only one LUN.
node	Any iSCSI initiator or iSCSI target, identified by its IQN.
portal	An IP address and port on a target or initiator used to establish connections. Some iSCSI implementations use portal and node interchangeably.
target	An iSCSI storage resource, configured for connection from an iSCSI server. Targets must be given unique names (see IQN). A target provides one or more numbered block devices called logical units (see LUN). An iSCSI server can provide many targets concurrently.
TPG	Target Portal Group, the set of interface IP addresses and TCP ports to which a specific iSCSI target will listen. Target configuration (e.g., ACLs) can be added to the TPG to coordinate settings for multiple LUNs.

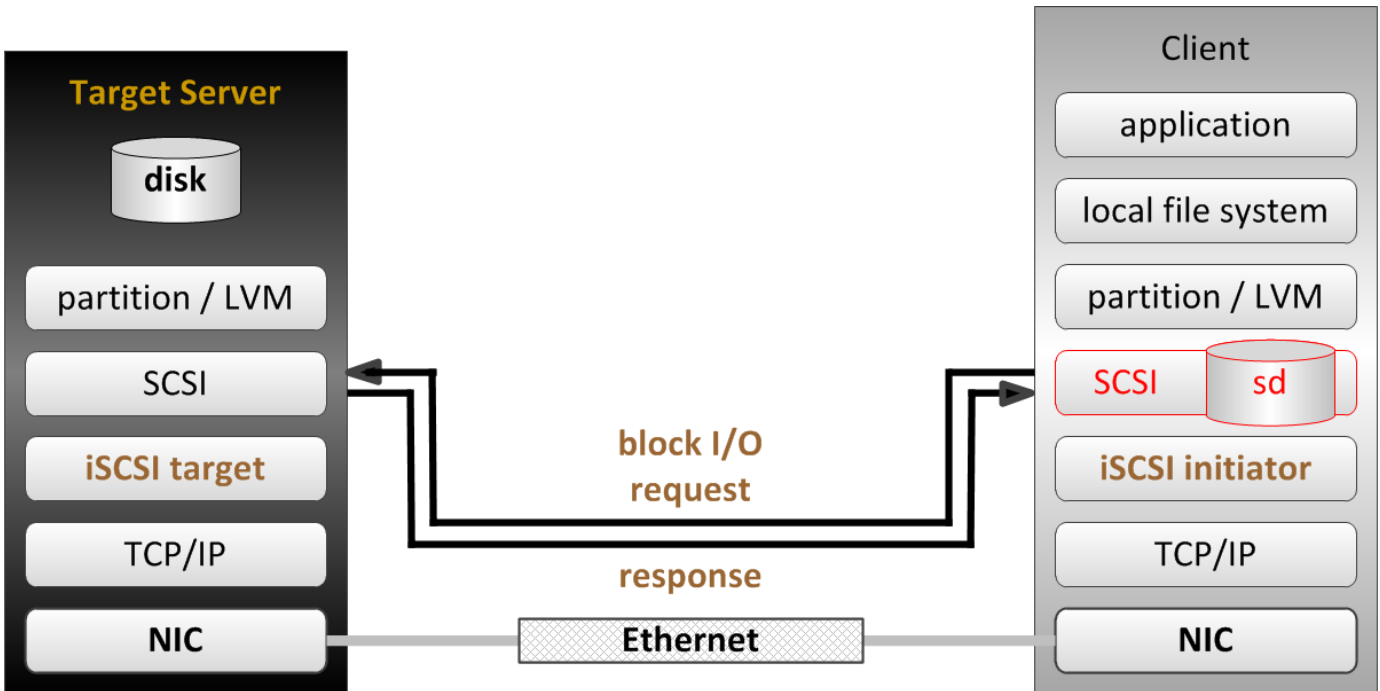


Figure 14. iSCSI Block I/O Network Stack



iSCSI uses ACLs to perform LUN masking to manage access to targets and LUNs from initiators.

7.2. Providing iSCSI Targets

Commands Used

- **targetcli**: Administration Shell (CLI) for iSCSI Storage Targets

Basic Steps for Configuring iSCSI Targets

1. Install **targetcli**
2. Setup **firewalld** rules
3. Configure **targetcli** service
4. Create/Configure Backstore
5. Create/Configure backstore
 - Might be LVMS, etc
 - Might be files
 - Create/Configure backstore in **targetcli** accordingly
6. Create/Configure iSCSI target name **IQN/TPG**
7. Create/Configure **ACL** for client
8. Create/Configure **LUN**

9. Verify target settings

10. Exit and Save

*Example 20. Demo of Providing iSCSI Target**Listing 84. Install targetcli*

```
[root@server0 ~]# yum -y install targetcli
```

Listing 85. Enable iSCSI firewall ports

```
[root@server0 ~]# firewall-cmd --permanent --add-port=3260/tcp
[root@server0 ~]# firewall-cmd --reload
```

Listing 86. Enable target Service

```
[root@server0 ~]# systemctl enable target; systemctl start target
```

Listing 87. Prepare Disk/LVM/File for Backing Store

```
[root@server0 ~]# mkdir /iSCSI_Backing
```

Listing 88. Run targetcli to Configure iSCSI Target

```
[root@server0 ~]# targetcli ①

/> cd /backstores/fileio ②
/backstores/fileio> create Demo_iSCSI /iSCSI_Backing/DemoBackstore.img 2G
Created fileio Demo_iSCSI with size 2147483648

/backstores/fileio> cd /iscsi ③
/iscsi> create iqn.2014-06.com.example:server0
Created target iqn.2014-06.com.example:server0.
Created TPG 1.

/iscsi> cd iqn.2014-06.com.example:server0/tpg1/acls ④
/iscsi/iqn.20...er0/tpg1/acls> create iqn.2014-06.com.example:desktop0
Created Node ACL for iqn.2014-06.com.example:desktop0

/iscsi/iqn.20...er0/tpg1/acls> cd ../luns ⑤
/iscsi/iqn.20...er0/tpg1/luns> create /backstores/fileio/Demo_iSCSI

/iscsi/iqn.20.../tpg1/portals> create 172.25.0.11 ⑥
Using default IP port 3260
Created network portal 172.25.0.11:3260.

cd ⑦
exit ⑧
```

① Launch **targetcli**

② Create **backstore**

- ③ Create iSCSI Target IQN
- ④ Create TPG ACL for iSCSI Initiator
- ⑤ Create/Associate LUN to the iSCSI target backstore
- ⑥ Create the Portal
- ⑦ Verify iSCSI Target settings
- ⑧ Exit and save data

```

- / ..... [ ... ]
o- backstores ..... [ ... ]
| o- block ..... [Storage Objects: 0]
| | o- fileio ..... [Storage Objects: 1]
| | | o- Demo_iSCSI ..... [/iSCSI_Backup/DemoBackstore.img (2.0GiB) write-back activated]
| | o- pscsi ..... [Storage Objects: 0]
| | o- ramdisk ..... [Storage Objects: 0]
o- iscsi ..... [Targets: 1]
| o- iqn.2014-06.com.example:server0 ..... [TPGs: 1]
| | o- tpg1 ..... [no-gen-acls, no-auth]
| | | o- acls ..... [ACLs: 1]
| | | | o- iqn.2014-06.com.example:desktop0 ..... [Mapped LUNs: 1]
| | | | | o- mapped_lun0 ..... [lun0 fileio/Demo_iSCSI (rw)]
| | | o- luns ..... [LUNs: 1]
| | | | o- lun0 ..... [fileio/Demo_iSCSI (/iSCSI_Backup/DemoBackstore.img)]
| | | o- portals ..... [Portals: 1]
| | | | o- 172.25.0.11:3260 ..... [OK]
o- loopback ..... [Targets: 0]
    
```

Figure 15. iSCSI Block I/O Network Stack



targetcli can be run in command-line mode. This is often useful for scripted tasks and other items. If using command-line mode, you **MUST** use **targetcli saveconfig** to save the configuration.

7.3. Accessing iSCSI Storage

Commands Used

- **iscsiadm**: open-iscsi administration utility

Config Files

- **/etc/iscsi/iscsid.conf**: Sets **iSCSI InitiatorName** used for ACLs
- **/etc/iscsi/initiatorname.iscsi**: Contains default settings for node records including;; timeouts, retry settings, and authentication settings.

Client portion of iSCSI Configuration



iSCSI initiators can be either software defined (in the case of the labs) or can be implemented with hardware initiators in terms of iSCSI HBAs

Basic Steps for iSCSI Clients (Initiators)

1. Ensure Initiator Utilities are installed
2. Ensure that the Initiator name is specified properly
3. Ensure the `iscsi` service is started and enabled
4. Use `iscsi` command to get target
 - a. Perform Target Discovery
 - b. Perform Target Login
5. Verify Disks presented after login
6. Use disks

Example 21. Demo Accessing iSCSI Storage

Listing 89. Install iscsi-initiator-utils

```
[root@desktop0 ~]# yum install -y iscsi-initiator-utils
```

Listing 90. Modify Initiator Name for ACL

```
[root@desktop0 ~]# echo "InitiatorName=iqn.2014-06.com.example:desktop0" > /etc/iscsi/initiatorname.iscsi
```

Listing 91. Enable and Start iSCSI Client Service

```
[root@desktop0 ~]# systemctl enable iscsi; systemctl start iscsi
```

Listing 92. Discover iSCSI Targets

```
[root@desktop0 ~]# iscsiadm -m discovery -t st -p 172.25.0.11
```

Listing 93. Login to iSCSI Targets

```
[root@desktop0 ~]# iscsiadm -m node -T iqn.2014-06.com.example:server0 -p 172.25.0.11 -l
```

Listing 94. Verify Disks Presented

```
[root@desktop0 ~]# lsblk
NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
sda   8:0    0  2G  0 disk ①
```

① sda is the iSCSI disk presented from the iSCSI server

Listing 95. Use disk by creating partition, filesystem and mounting it.

```
[root@desktop0 ~]# fdisk /dev/sda ①
[root@desktop0 ~]# mkfs.xfs /dev/sda1 ②
[root@desktop0 ~]# mkdir /iSCSI_Disk
[root@desktop0 ~]# mount -t xfs /dev/sda1 /iSCSI_Disk ③
[root@desktop0 ~]# touch /iSCSI_Disk/DummyFile
[root@desktop0 ~]# ls /iSCSI_Disk ④
```

- ① Create 2GB partition
- ② Create filesystem
- ③ Create directory and mount new disk
- ④ Create a test file and list the contents of the directory

Listing 96. Displaying Session Informant

```
[root@desktop0 ~]# iscsiadm -m session -P 3
```



Getting Detailed Information from iSCSI

- **iscsiadm -m discovery [-P 0|1]** - Shows information about discovered targets.
- **iscsiadm -m node [-P 0|1]** - Shows information about known targets.
- **iscsiadm -m session [-P 0|1|2|3]** - Shows information about active sessions.

8. Providing File-based Storage

8.1. Exporting NFS File Systems

Commands Used

- **nfs-server**: NFS Server Daemon
- **exportfs**: maintain table of exported NFS file systems



Exporting the same directory with NFS and Samba is not supported on Red Hat Enterprise Linux 7, because NFS and Samba use different file locking mechanisms, which can cause file corruption.

Example 22. DEMO NFS Share

Listing 97. Start and Enable `nfs-server`

```
[root@server0 ~]# systemctl enable nfs-server ; systemctl start nfs-server
```

Listing 98. Configure Firewall for NFS

```
[root@server0 ~]# firewall-cmd --permanent --add-service=nfs
[root@server0 ~]# firewall-cmd --reload
```

Listing 99. Create/Configure Directory to be Exported and Export it

```
[root@server0 ~]# mkdir /ExportDemo
[root@server0 ~]# echo "/ExportDemo desktop0.example.com(rw)" > /etc/exports
[root@server0 ~]# exportfs -ra
```

Listing 100. Mount NFS Share on Client

```
[root@desktop0 ~]# mkdir /NFS_Share_Demo
[root@desktop0 ~]# mount server0:/ExportDemo /NFS_Share_Demo
```

Listing 101. Test NFS Share on Client

```
[root@desktop0 ~]# touch /NFS_Share_Demo/Demo_Test.txt
touch: cannot touch '/NFS_Share_Demo/Demo_Test.txt': Permission denied ❶
[root@desktop0 ~]# umount /NFS_Share_Demo
```

- ❶ Share isn't configured to allow `NO_ROOT_SQUASH`

Listing 102. Adjust NFS Share and Retest

```
[root@server0 ~]# cat /etc/exports
/ExportDemo desktop0.example.com(rw,no_root_squash) ❶

[root@server0 ~]# exportfs -ra ❷

[root@desktop0 ~]# mount server0:/ExportDemo /NFS_Share_Demo ❸
[root@desktop0 ~]# touch /NFS_Share_Demo/Demo_Test.txt
```

- ❶ Allow Root Squash in `/etc/exports`
- ❷ Reload new config and re-export
- ❸ Mount and test files

`/etc/exports.d/` can be used to create multiple export files to configure exports. Files here could be `<name>.exports`.

Listing 103. Using the `exports.d` directory



```
[root@server0 exports.d]# echo "/ExportDemo2 desktop0.example.com(rw,no_root_squash)" > /etc/exports.d/Demo2.exports
[root@server0 exports.d]# mkdir /ExportDemo2
[root@server0 exports.d]# exportfs -ra
[root@desktop0 ~]# mkdir /NFS_Share_Demo2
[root@desktop0 ~]# mount server0:/ExportDemo2 /NFS_Share_Demo2
[root@desktop0 ~]# touch /NFS_Share_Demo2/secondDemo

[root@server0 ~]# tree /ExportDemo*
/ExportDemo
├── Demo_Test.txt
/ExportDemo2
├── secondDemo
```

8.2. Protecting NFS Exports

Important Files

- `/etc/krb5.keytab`

Table 6. Security Methods

Method	Description
<code>none</code>	Anonymous access to the files, writes to the server will be allocated UID and GID of <code>nfsnobody</code> . This requires the SELinux Boolean <code>nfsd_anon_write</code> to be active.
<code>sys</code>	File access based on standard Linux file permissions for UID and GID values. If not specified, this is the default. The NFS server trusts any UID sent by the client.
<code>krb5</code>	Clients must prove identity using Kerberos and then standard Linux file permissions apply. UID/GID is determined based upon the Kerberos principal from the accessing user.
<code>krb5i</code>	Adds a cryptographically strong guarantee that the data in each request has not been tampered with. UID/GID is determined based upon the Kerberos principal from the accessing user.

Method	Description
krb5p	Adds encryption to all requests between the client and the server, preventing data exposure on the network. This will have a performance impact, but provides the most security. UID/GID is determined based upon the Kerberos principal from the accessing user.



If using security options that require a Kerberos server, **nfs-secure-server** must be running in addition to **nfs-server** on the server, and the client system will need **nfs-secure** service to be running.

*Example 23. Secure NFS Demo**Listing 104. Preparing Systems*

```
[root@desktop0 ~]# lab nfskrb5 setup
[root@server0 ~]# lab nfskrb5 setup
```

Listing 105. Install Keytab

```
[root@server0 ~]# wget -O /etc/krb5.keytab http://classroom.example.com/pub/keytabs/server0.keytab
```

Listing 106. Start and Enable nfs-secure-server Service

```
[root@server0 ~]# systemctl enable nfs-secure-server ; systemctl start nfs-secure-server
```

Listing 107. Configure Firewall

```
[root@server0 ~]# firewall-cmd --permanent --add-service=nfs
[root@server0 ~]# firewall-cmd --reload
```

Listing 108. Prepare Export Directory

```
[root@server0 ~]# mkdir /Demo_securedexport
[root@server0 ~]# echo "/Demo_securedexport *.example.com(sec=krb5p,rw)" >/etc/exports
[root@server0 ~]# exportfs -ra
```

Listing 109. Configure Desktop with Keytab

```
[root@desktop0 ~]# wget -O /etc/krb5.keytab http://classroom.example.com/pub/keytabs/desktop0.keytab
```

Listing 110. Install NFS-Utills and Configure the NFS Secure Service

```
[root@desktop0 ~]# yum -y install nfs-utils
[root@desktop0 ~]# systemctl enable nfs-secure ; systemctl start nfs-secure
```

Listing 111. Create Directory and Mount Exported Filesystem

```
[root@desktop0 ~]# mkdir /SecureDemo
[root@desktop0 ~]# mount -o sec=krb5p server0:/Demo_securedexport /SecureDemo
```

Listing 112. Test the NFS Secure Filesystem

```
[root@desktop0 ~]# df -h
```

SELinux and labeled NFS

NFS mounts have SELinux context of **nfs_t**. It is possible to enforce mounting of NFS exports to enforce SELinux contexts. In order to successfully utilize SELinux contexts with NFS, the NFS server must be switched to version 4.2.



The **/etc/sysconfig/nfs** must be modified so that the **RPCNFSDARGS** contains the text from below:

```
RPCNFSDARGS="-V 4.2"
```



In a default installation of RHEL 7, the **nfs_export_all_ro** and **nfs_export_all_rw** SELinux Booleans are both enabled.

8.3. Providing SMB File Shares

SMB: Server Message Block is the standards file-sharing protocol for windows. (REF: **cifs**)

Commands and Packages

- **cif-utils:** Package to provide SMB file-sharing capabilities
- **smbpasswd:** Create Windows file-share storage username/password combination for the Samba security database
- **testparm:** Test parameters for the **smb.conf** file

The basic steps that must be performed in order to configure Samba to provide an SMB file share as a workgroup member are:

1. Install the samba package.
2. Prepare the permissions on the directory to be shared.
3. Configure **/etc/samba/smb.conf**.
4. Set up appropriate Linux users with NTLMv2 passwords.
5. Start Samba and open the local firewall.
6. Verify that the share can be mounted from a client.



The directive **read only = no** is the same as **writable = yes**,

SELinux and Samba

The `samba_enable_home_dirs` SELinux Boolean allows local Linux home directories to be shared by Samba to other systems. This needs to be enabled for `[homes]` to work (`setsebool -P samba_enable_home_dirs=on`).



The `use_samba_home_dirs` Boolean, on the other hand, allows remote SMB file shares to be mounted and used as local Linux home directories. It is easy to confuse the two options.

Listing 113. SELinux samba Boolean Values

```
[root@server0 /]# getsebool -a | grep -i samba
```

*Example 24. Samba Demo**Listing 114. Install samba package*

```
[root@server0 ~]# yum install samba
```

Listing 115. Create samba Shared Directory

```
[root@server0 ~]# mkdir /DemoSamba
```

Listing 116. Place Samba SELinux Labels on shared directory and files

```
[root@server0 ~]# semanage fcontext -a -t samba_share_t '/DemoSamba(/.*)?'
[root@server0 ~]# restorecon -vvFR /DemoSamba
```

Listing 117. Setup Filesystem Ownership and Permissions

```
[root@server0 /]# chmod -R 775 DemoSamba/
[root@server0 /]# chgrp users DemoSamba/
```

Listing 118. Install Samba Client

```
[root@server0 ~]# yum -y install samba-client
```

Listing 119. Creating Users

```
[root@server0 ~]# useradd -s /sbin/nologin -G users travis
[root@server0 ~]# smbpasswd -a travis
```

Listing 120. Edit `/etc/samba/smb.conf` File

```
[root@server0 ~]# vim /etc/samba/smb.conf
security = user ①
passdb backend = tdbsam ②
workgroup = DEMOWORKGROUP ③
hosts allow = 172.25. .example.com ④

[demoshare]
path = /DemoSamba
writable = yes
valid users = travis
write list = travis
```

Listing 121. Enable Firewall for Samba

```
[root@server0 ~]# firewall-cmd --permanent --add-service=samba
[root@server0 ~]# firewall-cmd --reload
```

Listing 122. Test `smb.conf` Configurations

```
[root@server0 ~]# testparm
```

Listing 123. Enable Samba Services

```
[root@server0 ~]# systemctl enable smb nmb ; systemctl start smb nmb
```

Listing 124. Test Samba on Client

```
[root@desktop0 ~]# yum -y install cifs-utils
[root@desktop0 ~]# mkdir /SambaTest

[root@desktop0 ~]# mount -o username=travis //server0/DemoShare /SambaTest
Password for travis@//server0/DemoShare: *****

[root@desktop0 ~]# echo "This is a test of Travis Demo Share" >> /SambaTest/Testfile.txt
```



Samba checks periodically to determine if `/etc/samba/smb.conf` has been changed. If the configuration file has changed, Samba automatically reloads it. This will not affect any connections already established to the Samba service, until the connection is closed or Samba is completely restarted.

The command `systemctl reload smb nmb` can be used to reload the configuration file immediately, or `systemctl restart smb nmb` to restart Samba entirely.

Getting Additional Help

Listing 125. Getting man pages



```
[root@desktop0 ~]# yum install selinux-policy-devel.noarch

[root@desktop0 ~]# man samba_selinux
```


8.4. Performing a Multi-user SMB Mount

- **cifscreds**: Manages NTLM credentials in kernel keyring

For multi-user shares the following takes place:

1. Root mounts the share using multiuser option on mount command and a valid low-permission SMB user (service account)
2. SMB users stash UN/PW combination in session keyring with **cifscreds** command



RHEL7 allows use of **sec=ntlmssp**. Keep in mind the **mount.cifs** man page does not list this as an option

Table 7. **cifscreds** Options

Option	Description
add	Adds SMB credentials to the session keyring of a user. This option is followed by the host name of the SMB file server.
update	Updates existing credentials in the session keyring of the user. This option is followed by the host name of the SMB file server.
clear	Removes a particular entry from the session keyring of the user. This option is followed by the host name of the Samba server.
clearall	Clears all existing credentials from the session keyring of the user.



By default, **cifscreds** assumes that the username to use with the SMB credentials matches the current Linux username. A different username can be used for SMB credentials with the **-u username** option after the **add**, **update**, or **clear** action.

9. Configuring MariaDB Databases

9.1. Installing MariaDB

MariaDB Server

- **mariadb**
 - **mariadb-server**: The MariaDB server and related files (mandatory package)
 - **mariadb-bench**: MariaDB benchmark scripts and data (optional package).
 - **mariadb-test**: The test suite distributed with MariaDB (optional package).

MariaDB Client

- **mariadb**: A community-developed branch of MySQL (mandatory package).
- **MySQL-python**: A MariaDB interface for Python (default package).
- **mysql-connector-odbc**: ODBC driver for MariaDB (default package).
- **libdbi-dbd-mysql**: MariaDB plug-in for libdbi (optional package).
- **mysql-connector-java**: Native Java driver for MariaDB (optional package).
- **perl-DBD-MySQL**: sA MariaDB interface for Perl (optional package)

MariaDB Config File

- **/etc/my.cnf**: Default configurations for MariaDB



Instead of adding new configurations to the **/etc/my.cnf** file, a newly created file named ***.cnf** can be added to the **/etc/my.cnf.d/** directory holding the configuration of MariaDB.

*Example 25. DEMO: Installation of MAriaDB**Listing 126. Install MariaDB Client and Server Packages*

```
[root@server0 ~]# yum groupinstall mariadb mariadb-client -y
```

Listing 127. Configure Firewall for MariaDB

```
[root@server0 ~]# firewall-cmd --add-service=mysql --permanent
[root@server0 ~]# firewall-cmd --reload
```

Listing 128. Start and Enable mariadb Service

```
[root@server0 ~]# systemctl enable mariadb ; systemctl start mariadb
```

Listing 129. Check to Ensure DB is Running

```
[root@server0 ~]# systemctl status mariadb
[root@server0 ~]# mysql
```

Secure Installation

- Sets password for **root** accounts
- Removes **root** accounts accessible from outside of **localhost**
- Removes **anonymous-user** accounts
- Removes **test** database

```
[root@server0 ~]# mysql_secure_installation
```

Secure installation can be run after the regular installation to secure and fix the MySQL installation.

*Listing 130. Start and Login to a Password Protected MariaDB Server*

```
[root@server0 ~]# mysql -p
```



It should be noted that MariaDB service listens on all interfaces by default, no users have remote access permission, also by default.



There can be only one **bind-address** entry in **/etc/my.cnf**. On a system with multiple addresses, selecting a single address is possible, or all addresses, but nothing in between.

9.2. Working with MariaDB Databases

Commands Used

- **mysql**: Provided by the **mariadb-client** group.



<http://classroom/materials/mariadb/inventory.dump>

```
MariaDB [mysql]> create database inventory; MariaDB [(none)]> \q [root@server0 /]# mysql -u root inventory < /root/inventory.dump -p
```

Example 26. Demo Using a Database

Listing 131. Connecting as Root on Localhost

```
[root@server0 /]# mysql -u root -h localhost -p
```

Listing 132. Showing Databases

```
MariaDB [(none)]> show databases;
```

Listing 133. Creating a Database

```
MariaDB [(none)]> create database demo;
MariaDB [(none)]> use demo;
MariaDB [demo]> show tables;
```

Information Commands

- **SHOW**: Provides listing
- **DESCRIBE**: Provides attribute and detailed listing

Listing 134. Viewing Database Information

```
MariaDB [mysql]> SHOW TABLES;
MariaDB [mysql]> describe time_zone;
```

SQL Language CRUD Commands

- **Create**: Uses **insert** directive
- **Read**: Uses **select** directive
- **Update**: Uses **update** directive
- **Delete**: Uses **delete** directive

9.3. Managing Database Users and Access Rights

MariaDB handles authentication and authorization through the **user** table in the **mysql** database.

Example 27. DEMO - Managing Users

Listing 135. Creating a New User

```
[root@server0 ~]# mysql -u root -h localhost -p
MariaDB [(none)]> create user travis@localhost identified by 'redhat';
MariaDB [(none)]> use mysql;
MariaDB [mysql]> SELECT host,user,password FROM user WHERE user = 'travis';
```

Listing 136. Granting Access

```
MariaDB [(none)]> use inventory;
MariaDB [inventory]> GRANT SELECT, UPDATE, DELETE, INSERT on inventory.category to travis@localhost;
```

Listing 137. Revoking Access

```
MariaDB [inventory]> REVOKE DELETE on inventory.category from travis@localhost;
```

Listing 138. Displaying Access Permissions

```
MariaDB [inventory]> SHOW GRANTS FOR travis@localhost;
```



1. After granting/revoking privileges, all privileges must be reloaded from the privilege table.

```
MariaDB [inventory]> flush privileges;
```

2. If an account is *DROP*ed while connected, it won't be deleted until the connection is closed.

Troubleshooting

- Check **skip-networking** in **my.cnf** if external connections not available
- Check **bind-address** in **my.cnf** if external connections not available
- Use can connect, but only sees **information_schema** and **test** means no privileges granted to user.

9.4. Creating and Restoring MariaDB Backups

Commands

- **mysql**: Client for accessing MariaDB

- **mysqladmin**: Client for administering a MySQL server

Two Backup Methods for MariaDB

- **Logical**: Export of information and records in plain text files
- **Physical (raw)**: Physical backups consisting of files and directories storing content

Listing 139. Logical Backup of a Database

```
[root@server0 ~]# mysqldump -u root -p inventory > /tmp/DemoBackup_Inventory.dump
[root@server0 ~]# cat /tmp/DemoBackup_Inventory.dump
```



Backing up All Databases

To perform a logical dump of all databases use the **--all-databases** instead of the name of the database.

```
[root@server0 ~]# mysqldump -u root -p --all-databases > /tmp/Maria_Complete_Dump.dump
```

Listing 140. Logical Restore of a Database

```
[root@server0 ~]# mysql -u root inventory < /root/inventory.dump -p
```



Discuss physical restore from Student Workbook

10. Providing Apache HTTPD Web Service

10.1. Configuring Apache HTTPD

- **httpd**: Package providing Apache webserver
- **httpd-manual**: Package providing the Apache manual

Apache Services

- **http**: Provides **http** access on port 80 by default
- **https**: Provides **https** secure SSL/TLS access on port 443 by default.



Accessing the Apache Manual

Need to install the **http-manual** package and access it from the Apache webserver using <http://localhost/manual>.

The **httpd** manual provides a comprehensive guide on using the Apache webserver with many examples that can be used to copy/paste into configuration files

Apache Configuration Files and Directories

- **/etc/httpd/conf/httpd.conf**: Default **httpd** configuration file.
- **/var/www/html**: Default document hosting location



See student guide for description of **httpd.conf** file.

*Example 28. DEMO Installing and Using Apache**Listing 141. Installing Apache*

```
[root@server0 /]# yum -y install httpd httpd-manual
```

Listing 142. Configuring Firewall

```
[root@server0 /]# firewall-cmd --permanent --add-service=http --add-service=https
[root@server0 /]# firewall-cmd --reload
```

Listing 143. Configuring Apache Service

```
[root@server0 /]# systemctl enable httpd.service ; systemctl start httpd.service
```

- Check Apache Manual

Figure 16. Apache Manual

- Test Serving Websites

Listing 144. Create Index.html

```
[root@server0 /]# echo "Welcome to my website demo" > /var/www/html/index.html
```

Figure 17. Apache Manual

10.1.1. HTTPD Security (SELinux)

The **httpd** service has SELinux configuration rules.

HTTPD SELinux Labels

- Port Labels
- Filesystem Labels

Apache binds to specific ports and can be managed with:

- **semanage port**

Apache needs to have a filesystem context as well:

- **httpd_sys_content_t**

SELinux Considerations

httpd allows the service to be configured to run on a variety of different ports. Additionally, **httpd** allows the DocumentRoot to be placed in a variety of locations.

*Changing Default Service Ports (Listens on Port 81)*

- `semanage port -a -t http_port_t -p tcp 81`

Changing Content Root (Content in /MyWebsites)

- `semanage fcontext -a -t httpd_sys_content_t '/MyWebsites(/.*)?'`

10.1.2. Write Access to DocumentRoot

ACLs can be used to allow read/write access to document root locations to allow more than just the root user to update content.

Listing 145. Existing Default Document Root

```
[root@server0 ~]# setfacl -R -m g:webmasters:rwX /var/www/html
[root@server0 ~]# setfacl -R -m d:g:webmasters:rwX /var/www/html
```

Listing 146. New Document Root

```
[root@server0 ~]# mkdir -p -m 2775 /MyWebsites
[root@server0 ~]# chgrp webmasters /MyWebsites
```



- **httpd-manual**: Installed with **httpd-manual**
- **httpd_selinux**: Installed with **selinux-policy-devel**
- **man semanage-port**

10.2. Configuring and Troubleshooting Virtual Hosts

- **Virtual Host**: Allows single HTTPD server to serve contents for multiple domains. This can serve up different sites based on IP address, hostname, or a combination of the two.



While it is possible to place virtual hosts in the main Apache config file: `/etc/httpd/conf/httpd.conf`, the recommended practice is to use separate `*.conf` files to define additional sites.

Listing 147. Sample Virtual Host

```
[root@server0 ~]# vim /etc/httpd/conf.d/DemoSite.conf
<Directory /srv/site1/www>
  Require all granted
  AllowOverride None
</Directory>

<VirtualHost 192.168.0.1:80>
  DocumentRoot /MyWebsite
  ServerName demosite.example.com
  ServerAdmin webmaster@site1.example.com
  ErrorLog "logs/site1_error_log"
  CustomLog "logs/site1_access_log" combined
</VirtualHost>
```



httpd can have the configuration tested to look for errors

Listing 148. Checking Apache Configuration

```
[root@server0 ~]# apachectl configtest
```

10.3. Configuring HTTPS

Packages and Applications

- **mod_ssl**: **SSL/TLS** module for Apache/httpd server
- **genkey**: Provided by the **crypto-utils** package. Used to generate a key and create certificate signing request. Easier to use than the **OpenSSL** method of generating certificates and keys.
- **TLS**: Transport Layer Security. Method of encrypting communications and is the replacement for **SSL (Secure Socket Layers)**. Based around the concept of certificates.

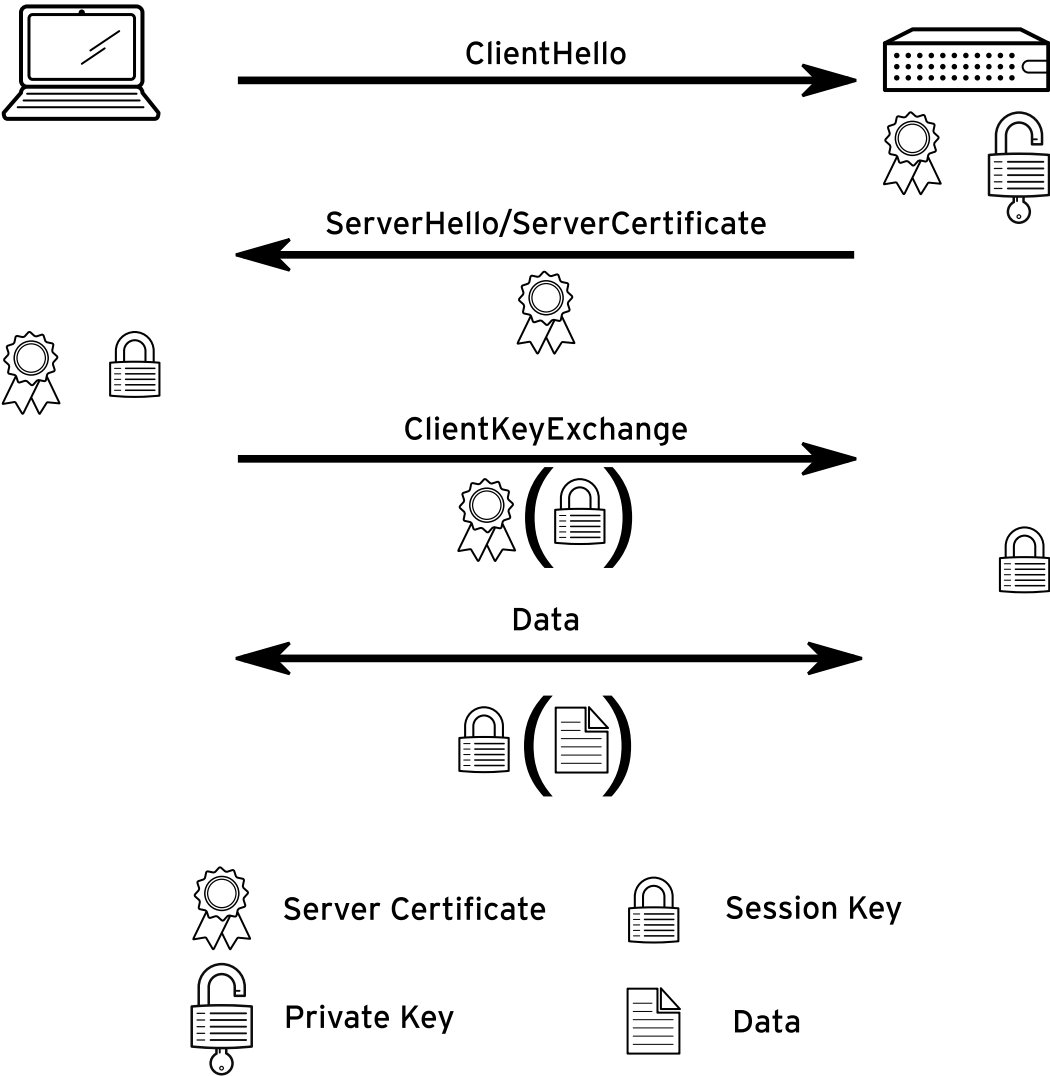


Figure 18. TLS Handshake

Discussion from student handbook on TLS and configuration.

Show Demo in Apache Manual

SSL/TLS Strong Encryption x +

server0/manual/ssl/ssl_howto.html

Modules | Directives | FAQ | Glossary | Sitemap

Apache HTTP Server Version 2.4

Apache > HTTP Server > Documentation > Version 2.4 > SSL/TLS

SSL/TLS Strong Encryption: How-To

Available Languages: en | fr

This document is intended to get you started, and get a few things working. You are strongly encouraged to read the rest of the SSL documentation, and arrive at a deeper understanding of the material, before progressing to the advanced techniques.

- Basic Configuration Example
- Cipher Suites and Enforcing Strong Security
- Client Authentication and Access Control
- Logging
- Comments

Basic Configuration Example

Your SSL configuration will need to contain, at minimum, the following directives.

```
LoadModule ssl_module modules/mod_ssl.so

Listen 443
<VirtualHost *:443>
    ServerName www.example.com
    SSLEngine on
    SSLCertificateFile /path/to/www.example.com.cert
    SSLCertificateKeyFile /path/to/www.example.com.key
</VirtualHost>
```

Cipher Suites and Enforcing Strong Security

- How can I create an SSL server which accepts strong encryption only?
- How can I create an SSL server which accepts all types of ciphers in general, but requires

Figure 19. HTTPS Manual Page

10.4. Integrating Dynamic Web Content

Dynamic Content

- **CGI**: Common Gateway Interface. Oldest form of generating dynamic web content.
- **PHP**: Can use `php` and `mod_php` to serve PHP content internally by adding the `mod_php` module to `httpd` as a PHP interpreter.
- **Python**: Serves web content using *Web Server Gateway Interface (WSGI)*. Provides a `WSGIScriptAlias` directive for all requests.



Discussion pieces from Student guide

Show content from Apache manual

Man Pages

- `httpd`
- `httpd_php_selinux`

11. Writing Bash Scripts

11.1. Bash Shell Scripting Basics

BASH

- **bash**: Bourne-Again SHell, an **sh**-compatible command language interpreter used as the Linux shell. There is also **ksh** and **csh** that are alternative shells to bash.

First lines of bash shell script is the command interpreter.

Listing 149. Command Interpreter

```
#!/bin/bash
...
```

Example 29. DEMO of Packages

Listing 150. Kernel Packages Script

```
[root@server0 ~]# vim package_listing.sh

#!/bin/bash
#
# This script provides information regarding when kernel-related packages
# are installed on a system by querying information from the RPM database.
#
# Variables
PACKAGETYPE=kernel
PACKAGES=$(rpm -qa | grep $PACKAGETYPE)
# Loop through packages
for PACKAGE in $PACKAGES; do
    # Determine package install date and time
    INSTALLEPOCH=$(rpm -q --qf "%{INSTALLTIME}\n" $PACKAGE)

    # RPM reports time in epoch, so need to convert
    # it to date and time format with date command
    INSTALLDATETIME=$(date -d @$INSTALLEPOCH)

    # Print message
    echo "$PACKAGE was installed on $INSTALLDATETIME"
done
```

Listing 151. Make Script Executable

```
[root@server0 ~]# chmod +x package_listing.sh
```

Listing 152. Running the Script

```
[root@server0 ~]# ./package_listing.sh
```



Discussion from Student Guide

12. Bash Conditionals and Control Structures

12.1. Enhancing Bash Shell Scripts with Conditionals and Control Structures

BASH Conditionals

- **if/then**: Logical tested on conditions. Also part of **if/then/else** statements.
- **case**: Used for testing conditions where multiple **else ifs** would be used. This conditional makes **if/then/elif/then/else** much easier to read.



Use student guide to go through discussions.

13. Configuring the Shell Environment

13.1. Changing the Shell Environment

Bash Environment

- **Environment Variables:** Special variables that go with the shell
- **/etc/profile:** Location for shell scripts and start-up scripts. These are executed first
- **~/.bash_profile:** User's home directory BASH profile startup scripts executed after **/etc/profile**
- **~/.bashrc:** Bash Resource file. Can contain aliases and are the RCs (run commands) for each time a shell is created
- **alias:** Short term for a command



Go through instructions in book

Appendix A: Exam Objectives for EX300 RHCE Exam

The following exam objectives have been taken from the **only** official source for the RHCE EX300 exam. The current objectives are available at the following location: <https://www.redhat.com/en/services/training/ex300-red-hat-certified-engineer-rhce-exam-red-hat-enterprise-linux-7>

The official course for the EX300 is the RH254 which is located at the following location:

<https://www.redhat.com/en/services/training/rh254-red-hat-system-administration-iii>

A.1. System configuration and management

- Use network teaming or bonding to configure aggregated network links between two Red Hat Enterprise Linux systems
- Configure IPv6 addresses and perform basic IPv6 troubleshooting
- Route IP traffic and create static routes
- Use firewalld and associated mechanisms such as rich rules, zones and custom rules, to implement packet filtering and configure network address translation (NAT)
- Configure a system to authenticate using Kerberos
- Configure a system as either an iSCSI target or initiator that persistently mounts an iSCSI target
- Produce and deliver reports on system utilization (processor, memory, disk, and network)
- Use shell scripting to automate system maintenance tasks

A.2. Network services

Network services are an important subset of the exam objectives. RHCE candidates should be capable of meeting the following objectives for each of the network services listed below:

- Install the packages needed to provide the service
- Configure SELinux to support the service
- Use SELinux port labeling to allow services to use non-standard ports
- Configure the service to start when the system is booted
- Configure the service for basic operation
- Configure host-based and user-based security for the service

A.2.1. HTTP/HTTPS

- Configure a virtual host
- Configure access restrictions on directories
- Deploy a basic CGI application
- Configure group-managed content

- Configure TLS security

A.2.2. DNS

- Configure a caching-only name server
- Troubleshoot DNS client issues

A.2.3. NFS

- Provide network shares to specific clients
- Provide network shares suitable for group collaboration
- Use Kerberos to control access to NFS network shares

A.2.4. SMB

- Provide network shares to specific clients
- Provide network shares suitable for group collaboration

A.2.5. SMTP

- Configure a system to forward all email to a central mail server

A.2.6. SSH

- Configure key-based authentication
- Configure additional options described in documentation

A.2.7. NTP

- Synchronize time using other NTP peers

A.3. Database services

- Install and configure MariaDB
- Backup and restore a database
- Create a simple database schema
- Perform simple SQL queries against a database