2.Operate running systems

1. **Boot, reboot, and shut down a system normally**

To reboot the system, choose one command among these:

# **reboot**

# **systemctl reboot**

# **shutdown -r now**

# **init 6**

# **telinit 6**

To shutdown the system, choose one command among these:

# **halt**

# **systemctl halt**

# **shutdown -h now**

# **init 0**

# **telinit 0**

To switch off the system, choose one command among these:

# **poweroff**

# **systemctl poweroff**

**Advanced Management**

To suspend the system, type:

# **systemctl suspend**

To put the system into hibernation, type:

# **systemctl hibernate**

To put the system into hibernation and suspend it, type:

# **systemctl hybrid-sleep**

1. **Boot systems into different targets manually**

Before **Systemd**, there were the following runlevels:

* 1, single: maintenance level,
* 2: level without network resources (NFS, etc),
* 3: multi-user level without graphical interface,
* 5: multi-user level with graphical interface.

Note: The default run level was set in the **/etc/inittab** file.

To get the current run level with the old way, type:

# **runlevel**

To change the current run level (where **X** is the run level), type:

# **init X**

Note: As seen before, levels **0** and **6** were respectively used for halting a system and for rebooting it.

**Current State**

With **Systemd**, new commands are available:

* **systemctl rescue**: to move to single user mode/maintenance level with mounted local file systems,
* **systemctl emergency**: to move to single user mode/maintenance with only **/root**mounted file system,
* **systemctl isolate multi-user.target**: to move to multi-user level without graphical interface (equivalent to previous run level 3),
* **systemctl isolate graphical.target**: to move to multi-user level with graphical interface (equivalent to previous run level 5),
* **systemctl set-default** **graphical.target**: to set the default run level to multi-user graphical mode,
* **systemctl get-default**: to get the default run level.

# systemctl get-default

graphical.target

OR

# ls -l /etc/systemd/system/default.target

# systemctl set-default multi-user.target

# systemctl set-default multi-user

# systemctl set-default runlevel3.target

In the below example we will temporarily change from the graphical runlevel to multi-usertarget. 

1. **Interrupt the boot process in order to gain access to a system**

Note: This is a **critical** [RHCSA 7 exam objective](https://www.certdepot.net/rhel7-rhcsa-exam-objectives/) (if you can’t take control of a VM through a reboot at the beginning of the exam, you will fail it entirely).

**Presentation**

In **RHEL 7**, the procedure to get access to a system during the boot process and modify the root password has changed because of the adoption of **Systemd**.

There were several procedures floating around to recover the root password. Some were working with physical servers but not with virtual machines, some the other way around.

The following procedure works all the time.

**Procedure**

At the beginning of the boot process, at the **GRUB 2** menu, type the **e** key to edit.

Then, go to the kernel line (the line starting with **linux16**) and add the following statements at the end:

**rd.break enforcing=0**

Caution: The keys to press are those of a **US** keyboard (querty).  
Note: **rd.break** asks for a break at an early stage of the boot process. **enforcing=0** puts the system into **SELinux Permissive** mode. Don’t confuse with **selinux=0** that completely disables **SELinux**.

Press **Ctrl x** to resume the boot process.

Then, mount the **/sysroot** partition as read/write:

switch\_root:/# **mount –o remount,rw /sysroot**

Execute the **chroot** command on the **/sysroot** partition:

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

Change the **root** password:

sh-4.2# **passwd root**

Changing password for user root.

New passwd:

Retype new password:

passwd: all authentication token updated successfully.

sh-4.2# **exit**

exit

switch\_root:/# **exit**

logout

Connect to your server at the console (don’t reboot now!) with the **root** user and the new password:

...

[  OK  ] Started Network Manager Script Dispatcher Service.

[  OK  ] Started Crash recovery kernel arming.

[  OK  ] Reached target Multi-User System.

CentOS Linux 7 (Core)

Kernel 3.10.0-229.14.1.el7.x86\_64 on an x86\_64

vm login: **root**

Password:

Then type:

# **restorecon /etc/shadow**

# **reboot**

If you strictly follow this procedure, you don’t need to force a **SELinux** relabel (# **touch /.autorelabel** or # **fixfiles onboot**) or load the **SELinux** policy (# **/usr/sbin/load\_policy -i**).

You don’t even need to reboot at the end! In this case, type # **setenforce enforcing**

For the **RHCSA** exam, you need to intensely practice this procedure.

Thanks to **salvador**and **hunter86\_bg**for their precious comments.

**Note:** When dealing with boot problems, the following options can be added to the kernel command line, bringing additional information:

**rd.debug rd.udev.debug systems.log\_level=debug**

1. **Identify CPU/memory intensive processes, adjust process priority with renice, and kill processes**

## **System Activities**

To get an instantaneous image of a server activity (use ‘**virt-top**‘ on a **KVM** hypervisor), type:

# top

To get details about processes, type:

# ps -edf

## **Process Priority**

To start a process (here **script.sh**) with a low priority, type:

# nice -n 10 ./script.sh

To change the priority (here **+5**) of an already running process, get its PID (Process ID) through top or ps (here **789**) and type:

# renice +5 789

**Alternatively**:

# renice +5 `pgrep script.sh`

## **Process Deletion**

To kill the process, get its PID through top or ps (here **789**) and type:

# kill -9 789

**Alternatively**:

# pkill script.sh

## System Reporting

To display details about IO activities, type:

# iostat

To show network card activities, type:

# netstat -i

To display socket activities, type:

# netstat -a

To get details about virtual memory activities (memory, swap, run queue, cpu usage, etc) every 5 second, type:

# vmstat 5

To get a full report of a server activity, type:

# sar -A

1. **Locate and interpret system log files and journals**

Most of system log files are located in the **/var/log** directory due to **SYSLOG**default configuration (see **/etc/rsyslog.conf** file).

In addition, all **SELinux** events are written into the **/var/log/audit/audit.log** file.

With **Systemd**, new commands have been created to analyse logs at boot time and later.

## Boot Process

**Systemd** primary task is to manage the boot process and provides informations about it.  
To get the boot process duration, type:

# **systemd-analyze**

Startup finished in 422ms (kernel) + 2.722s (initrd) + 9.674s (userspace) = 12.820s

To get the time spent by each task during the boot process, type:

# **systemd-analyze blame**

7.029s network.service

2.241s plymouth-start.service

1.293s kdump.service

1.156s plymouth-quit-wait.service

1.048s firewalld.service

632ms postfix.service

621ms tuned.service

460ms iprupdate.service

446ms iprinit.service

344ms accounts-daemon.service

...

7ms systemd-update-utmp-runlevel.service

5ms systemd-random-seed.service

5ms sys-kernel-config.mount

Note: You will find additional information on this point in the [Lennart Poettering’s blog](http://0pointer.de/blog/projects/blame-game.html).

## **Journal Analysis**

In addition, **Systemd** handles the system event log, a **syslog** daemon is not mandatory any more.  
The reasons behind **Journald** creation are explained in this [Introduction to Journald](https://docs.google.com/document/pub?id=1IC9yOXj7j6cdLLxWEBAGRL6wl97tFxgjLUEHIX3MSTs&pli=1).

To get the content of the **Systemd** journal, type:

# **journalctl**

To get all the events related to the **crond** process in the journal, type:

# **journalctl /sbin/crond**

Note: You can replace **/sbin/crond** by **`which crond`**.

**Altenatively**, to get all the events related to the **crond** process, you can also type:

# **journalctl -u crond**

To get all the events since the last boot, type:

# **journalctl -b**

To get all the events that appeared today in the journal, type:

# **journalctl --since=today**

To get all the events with a syslog priority of err, type:

# **journalctl -p err**

To get the 10 last events and wait for any new one (like **tail -f /var/log/messages**), type:

# **journalctl -f**

## **Permanent Storage**

By default, **Journald** logs are stored in the **/run/log/journal** directory and disappear after a reboot.

To store **Journald** logs in a more permanent way, type:

# **mkdir /var/log/journal**

# **echo "SystemMaxUse=50M" >> /etc/systemd/journald.conf**

# **systemctl restart systemd-journald**

Note: Setting the **SystemMaxUse** variable is necessary because otherwise **10%** of the filesystem where the **/var/log/journal** directory is stored may be used at maximum by the journal.

1. **Access a virtual machine's console**

## Standard procedure

With **KVM**, to access the virtual machine’s console under **X Window**, type:

# **virt-manager**

If you aren’t under **X Window**, there is another way to access a virtual machine’s console: you can go through a **serial console**.

On the virtual machine, add ‘**console=ttyS0**‘ at the end of the kernel lines in the **/boot/grub2/grub.cfg** file:

# **grubby --update-kernel=ALL --args="console=ttyS0"**

Note: Alternatively, you can edit the **/etc/default/grub** file, add ‘**console=ttyS0**‘ to the **GRUB\_CMDLINE\_LINUX** variable and execute ‘**# grub2-mkconfig -o /boot/grub2/grub.cfg**‘.

Now, reboot the virtual machine:

# **reboot**

With **KVM**, connect to the virtual machine’s console (here **vm.example.com**):

# **virsh console vm.example.com**

Connected to domain vm.example.com

Escape character is ^]

Red Hat Enterprise Linux Server 7.0 (Maipo)

Kernel 3.10.0-121.el7.x86\_64 on an x86\_64

vm login:

## Emergency procedure

Sometimes you have lost all links to your virtual machine (error in the **/etc/fstab** file, ssh configuration, etc) and, as you didn’t set up any virtual console, you are in real trouble. There is still a solution!  
Connect to the physical host and shut down your virtual machine (here called **vm.example.com**):

# **virsh destroy vm.example.com**

Define where the virtual machine image file is located (by default in the **/var/lib/libvirt/images**directory with a name like **vm.example.com.img**):

# **virsh dumpxml | grep "source file="**

<source file='/var/lib/libvirt/images/vm.example.com.img'/>

Map your virtual machine image file into the host environment (**-a** for add and **-v** for verbose):

# **kpartx -av /var/lib/libvirt/images/vm.example.com.img**

add map loop0p1 (253:2): 0 1024000 linear /dev/loop0 2048

add map loop0p2 (253:3): 0 10240000 linear /dev/loop0 1026048

From the previous display, you know that you’ve got two partitions (in fact **/boot** and **/**, distinguishable by their respective size).  
You need to mount the **/boot** partition to be able to change the grub configuration:

# **mount /dev/mapper/loop0p1 /mnt**

Then, edit the **/mnt/grub2/grub.cfg** file and add ‘**console=ttyS0**‘ at the end of every line containing **/vmlinuz** (the **linux** kernel).  
Unmount the partition:

# **umount /mnt**

Unmap the virtual machine image file (**-d** for delete and **-v** for verbose):

# **kpartx -dv /var/lib/libvirt/images/vm.example.com.img**

del devmap : loop0p2

del devmap : loop0p1

loop deleted : /dev/loop0

Restart your virtual machine:

# **virsh start vm.example.com**

Domain vm.example.com started

Connect to your virtual machine console:

# **virsh console vm.example.com**

Connected to domain vm.example.com

Escape character is ^]

CentOS Linux 7 (Core)

Kernel 3.10.0-123.el7.x86\_64 on an x86\_64

vm login:

This procedure works for **RHEL 6**/**CentOS 6** and **RHEL 7**/**CentOS 7**.

1. **Start and stop virtual machines**

When talking about virtual machines, it is mainly question of**KVM** management through the **virsh**and **virt-\*** commands.

## VM Management

To start a virtual machine (here **vm.example.com**), type:

# virsh start vm.example.com

To stop a virtual machine (here **vm.example.com**), type:

# virsh shutdown vm.example.com

To stop immediately a virtual machine (here **vm.example.com**), type:

# virsh destroy vm.example.com

To delete a virtual machine (here **vm.example.com**), type:

# virsh undefine vm.example.com

To reboot a virtual machine (here **vm.example.com**), type:

# virsh reboot vm.example.com

To display configuration information (memory, state, autostart, etc) about a virtual machine (here **vm.example.com**), type:

# virsh dominfo vm.example.com

## VM Reporting

To get the list of all the virtual machines (active and inactive), type:

# virsh list --all

Note: Remove the **–all** option to only get the list of the active virtual machines.

To get a global picture of the VM activities, type:

# virt-top

1. **Start, stop, and check the status of network services**

Even though this topic seems very simple, you need to fully understand all its details.

Also, there is a kind of compatibility between the **RHEL 6** **service** command and the **RHEL 7 systemctl**command: **service** daemon cmd => **systemctl** cmd daemon

## Basic Service Management

To start a network service (here **httpd**), type:

# **systemctl start httpd**

Note: In addition, a service can be restarted with the **restart** option or only reloaded with the **reload** option.

To stop a network service (here **httpd**), type:

# **systemctl stop httpd**

To check if a network service (here **httpd**) is running, type:

# **systemctl is-active httpd**

unknown

Note: If the service is running, you get **active**.

To activate a network service at boot (here **httpd**), type:

# **systemctl enable httpd**

ln -s '/usr/lib/systemd/system/httpd.service' '/etc/systemd/system/multi-user.target.wants/httpd.service'

Note1: Use the **disable** option to inactivate a network service at boot.  
Note2: **Systemd** uses a link mechanism to manage this feature.

To check if a network service (here **httpd**) is enabled at boot, type:

# **systemctl is-enabled httpd**

disabled

Note: If the service is enabled at boot, you get **enabled**.

To check the status of a network service (here **httpd**), type:

# **systemctl status httpd**

httpd.service – **The Apache HTTP Server**

   Loaded: **loaded** (**/usr/lib/ystem/system/httpd.service**; **enabled**)

   Active: **active** (running) since **Fri 2015-06-19 16:47:18 CEST; 6min ago**

 Main PID: **3868** (httpd)

   Status: “**Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec**”

   Cgroup: **/system.slice/httpd.service**

           └─3868 /usr/sbin/httpd -DFOREGROUND

           └─3869 /usr/sbin/httpd -DFOREGROUND

           └─3870 /usr/sbin/httpd -DFOREGROUND

           └─3871 /usr/sbin/httpd -DFOREGROUND

           └─3872 /usr/sbin/httpd -DFOREGROUND

           └─3873 /usr/sbin/httpd -DFOREGROUND

Jun 19 16:47:18 server1.example.com ystem[1]: Starting The Apache HTTP Ser….

Jun 19 16:47:18 server1.example.com ystem[1]: Started The Apache HTTP Server.

Hint: Some lines were ellipsized, use -l to show in full.

Note: There are many information available through this command, learn to use them.

## **Advanced Service Management**

To permanently disable a service (here **httpd**), type:

# **systemctl mask httpd**

ln -s '/dev/null' '/etc/systemd/system/httpd.service'

Note1: Masking a service prevents it from starting even if it is **socket-activated** or **dbus-activated**.  
Note2: Use the **unmask** option to enable the service again.

1. **Securely transfer files between systems**

There are many ways to transfer files from a system to another.  
Here we will consider the **scp** command that relies on **SSH** that is normally installed on most hosts.

## **Transfer of a local file**

First, we create a file called **loc** locally:

# **cd; echo "This is a test." > loc**

To transfer the local file to a remote host (here called **centos**) into the **root**‘s home directory, type:

# **scp loc root@centos:loc**

root@centos's password:

loc                                           100%   16     0.0KB/s   00:00

Note: By default, the file is put into the user’s home directory but it is possible to give a complete path.

To copy all the files from a specified directory, type:

# **scp /etc/ssh/\* root@centos:/tmp**

root@centos's password:

moduli                                        100%  236KB 236.5KB/s   00:00

ssh\_config                                    100% 2123     2.1KB/s   00:00

sshd\_config                                   100% 4442     4.3KB/s   00:00

ssh\_host\_ecdsa\_key                            100%  227     0.2KB/s   00:00

ssh\_host\_ecdsa\_key.pub                        100%  162     0.2KB/s   00:00

ssh\_host\_rsa\_key                              100% 1679     1.6KB/s   00:00

ssh\_host\_rsa\_key.pub                          100%  382     0.4KB/s   00:00

Note: If directories appear in the list created by the **\***, there are not transferred: you get a “**not a regular file**” error (use the **tar** command to transfer directories).

## **Transfer of a remote file**

Conversely, it is possible to transfer a file from a remote host.

On the **centos** host, create the **rem** file in the **/tmp** directory:

# **cd /tmp**

# **echo "This is another test." > rem**

Locally, to transfer the file, type:

# **scp root@centos:/tmp/rem rem**

root@centos's password:

rem                                           100%   22     0.0KB/s   00:00