Network:

FTP = 21, ssh = 22 SMTP = 25 DNS = 53 POP3 = 110 NTP = 123	Class-A = 10.0.0.0 – 10.255.255.255 Class-B = 172.16.0.0 – 172.31.255.255 Class-C = 192.168.0.0 – 192.168.255.255 Multicast = 224.0.0.0 – 239.255.255.255	Vi: Search and replace globally: **:%s/string1/string2/g** Just on the current line: **:s/string1/string2/g**

Misc info and Linux commands:

/etc/skel/ # contains files that are copied to the user's homedir when a new user is created first
pwconv # sets the /etc/shadow entry if a corresponding entry is not in shadow file
/etc/login.defs # has the user's settings when a use is created, also has global setting umask=007 (recap?)
/var/spool/main/<user name>/ # has the location of mails for each user
Command precedence **:** alias >> function >> built-ins >> external command
enable || **compgen -b** # shows shell built-in commands

setting date and time
hwclock [**--show**] # to see RTC time.
hwclock –set –date "<time in format dd mm YYYY HH:MM>" # to set the RTC clock time
hwclock --systohc --set # to set the system clock to match RTC clock time
timedatectl status # shows current time set to the system
timedatectl set-timezone America/New-York # sets the timezone
date -s "Mon June 27 09:00:00" # to set the system date
tzselect # to set system time zone, its reflected at **/etc/localtime** -> link to **/America/new-York** (e.g.)

find command
find -size 5M # finds file with exactly 5M size (+5M find files with more than 5M size and -5M **..** you know **..**)
find / -type -f -perm -u=s -ls # list all files with setuid bit set
find / -type -f -perm -g=s -ls # list all files with setgid bit set

parted -l /dev/nvme0n1 # shows partitions created
blockdev --getbsz /dev/nvme0n1 # shows block size (the bare disk shows 4096 but the nvme0n1p1 shows 512)
mkswap /dev/sda2 >> **swapon /dev/sda2** >> add this to **/etc/fstab** ## adds swaps
mount -f -o remount <fs_mount_path> # mounts the file system

The column VIRT reflects amount of virtual memory used by the process, RSS reflects real memory used

ps -o pid,pri,ni,cmd # shows nice value and priority of a process
nice -n 19 bash | **renice 10 8389** # will change the priority of the program or pid
SIGHUP (same as **kill -1 <pid>**) # restarts the process with same pid, generally used to reread the config file
CRT + Z (SIGSTOP) e.g. **sleep 1000** >> **Crtl + Z** (suspend the process) >> **bg** (resume the process in background, **fg** brings it in foreground)

/etc/syslog.conf:
<facility>.<priority> <destination_file_name_to_log>
user.info /var/log/messages

to test
logger -p <facility>.<priority> "log message" # to test

Typical troubleshooting command:
lshw -short | **uptime**| **sar 2 5** | **more /proc/meminfo** | **iotop** | **xfs_repair /dev/sda1** | OOM killer?

cat /proc/sys/net/ipv4/ip_forward # to check ip forwarding is enabled for layer 3 , should be 1

Misc commands:
chronyc sources
chronyc tracking

nfs
showmount -e system5 # At NFS server, to see mounts exported to client system5
server side: **nfsstat** client side: **nfsiostat 1**

Logrotate crap:

/etc/logrotate.conf # main config file, just place a file under /etc/logrorate.d

```
/var/log/apache2/* {
    weekly            # rotates the log on weekly basis
    rotate 3          # only 3 rotated logs should be kept
    size 10M          # sets min for rotation to take place is 10M
    compress
    delaycompress     # compress/ decompress tells all rotated logs except of recent one should be compressed
}
```

logrotate -d /etc/logrotate.d/apache2.conf # to test it

At/cron/systemd timer crap:

```
at|cron|anachron  # st starts with systemctl status atd.service; cron starts with crond.service
e.g.
# at> updated > $HOME/file
# at><EOT>
atq  # displays job using at

/etc/crontab # system wide crontab, crontab -e # to edit the commands, crontab -l to display
/etc/cron.hourly |cron.daily |cron.weekly |cron.monthly #holds commands that is used to run scripts in

systemctl list-unit-files *.timer  # list current services using system.timer, e.g.

# Example-1: sysstat using system timer, sysstat.service and sysstat-collect.timer
[Unit]
Description=Resets System Activity Logs

[Service]
Type=oneshot
RemainAfterExit=yes
User=root
ExecStart=/usr/lib64/sa/sa1 --boot

[Install]
WantedBy=multi-user.target
Also=sysstat-collect.timer
Also=sysstat-summary.timer
```

```
[Unit]
Description=Run system activity accounting tool every 10 minutes

[Timer]
OnCalendar=*:00/10

[Install]
WantedBy=sysstat.service
```

```
# Example-2: running free using systemd timer, myMonitor.service

[Unit]
Description=Logs system statistics to the systemd journal
Wants=myMonitor.timer

[Service]
Type=oneshot
ExecStart=/usr/bin/free

[Install]
WantedBy=multi-user.target
```

```
# myMonitor.timer

[Unit]
Description=Logs some system statistics to the
systemd journal
Requires=myMonitor.service

[Timer]
Unit=myMonitor.service
OnCalendar=*-*-* *:*:00  # triggers every min

[Install]
WantedBy=timers.target
```

Hardware mgmt. + LVM:

```
lspci -n # can show vid:pid | lspci -tv # shows pci tree and devices
udevadm info -q property -n /dev/sdb

lshw -short [ --class storage | --class volume ] #shows HW path., device, class, description; Only storage | volume
lsusb -t # ..
lsscsi -d | lsscci --hosts # disks ;
lpadmin | lpremove # to add, remove printers

dmesg -T -f kern -l notice | grep sdb
abrt | abrt-auto-reporting-enabled | abrt-cli list | abrt-cli list since 024567287 # to display fault commands
```

yum/sysctl/nmcli crap:

```
subscription-manager register

yum repo list # lists enabled repo in my system
yum repo list –all # lists enabled and disabled repo in my system
yum repo list –all | grep supplementary # ..
yum config-manager ––enable <one from the above> # ..
yum config-manager –add-repo=http://mirror .../../..   # this will add a .repo file in /etc/yum.repos.d/
yum module list | dnf module list
dnf module enable seala:2-10. | <module>:<stream> | dnf module info <module>:<stream>

yum list available # list of available packages in all enabled repo
yum list installed
yum list kernel # installed and available pkg with specific pkg name “kernel” (e.g. yum list kernel)
yum info <pkg>
yum search <search string> # searches pkg name and description of pkg on enabled repos
yum enablerepo=<repo-name> install <pkg>
yum update <pkg>
yum install <pkg> ––downloadonly –downloaddir=<dirname> ## does not install, only downloads
yum install yum-plugin downloadonly
yum deplist bash # what other pkg needed by bash
yum list available kernel* # list all available kernel pkgs
yum check-update # check enabled repo for available pkg update
dnf history | dnf history undo last  # you know it

dnf check-update  # to see which packages are ready for update
dnf remove <pkg>  #..
dnf grouplist
dnf groupinstall “System Tools”
dnf history info 3
dnf history rollback 8  # rolls back everything after transaction no 8
dnf history undo 10 –y  # undo a single transaction

# let’s say you have your own slurm repo
dnf install –y createrepo >> mkdir –p /var/tmp/myrepo; cd /var/tmp/myrepo; >> mkdir slurm; cd slurm ; now copy the
rpms you built here >> createrepo . >> cd /etc/yum.repos.d/ >> vi slurm.repo and adds following contents
[slurm.repo]
name=My local slurm repo
baseurl=file:///var/tmp/myrepo/slurm
enabled=1
gpgcheck=0
gpgkey=file:///etc/..

# synching a repo (from satellite?), example shows only synching baseos rom of rh8

subscription-manager repos ––disable=”*”
subscription-manager repos ––enable=”rhel-8-for-x86_64-baseos-rpms”
cd /var/tmp/repos
reposync ––newest-only ––download-metadata ––destdir /var/tmp/repos
# download-metadata download fully functional repomod.xml, so you do not have to build it.
```

Systemd interractions:

```
systemctl list-unit-files *.target  # same as ls –l /usr/lib/system/system/*.target

systemctl get-default # to check default target (runlevel in sysV style)
systemctl set-default multi-user.target
# default.target is a link to multiuser.target
graphical.target = runlevel 5; multiuser.target = 3; rescue.target = 1; emergency.target = S; reboot.target = 6;
poweroff = 0


systemctl list-dependencies <unit-name>  #list units that a unit depends on
systemctl list-dependencies –reverse <unit-name> # opposite of above
systemctl –type=target all  # list all available targets
systemctl list-dependencies –all multi-user.target | rescue.target

systemctl isolate multiuser.target # to go to multiuser
systemctl list-units –type target
runlevel          O/P: 5 3 means, earlier it was 5 and now 3

systemctl list-sockets # lists sockets and what activates
systemctl daemon-reload # you know it
systemctl reboot | poweroff | emergency # use this always

journalctl # to view syslog messages
journalctl –u network.service # to see network service messages
journalct –k # show only kernel messages (dmesg equivalent)
```

Systemd service file(/etc/system/system):

Service definition file(/etc/system/system):
ExecStart = ..
ExecPrestart = ..
Type = timer|socket|slice (it's the O/P of systemctl -t help)
..

 # Type=simple

```
# no Restart, file name: simple-test.service
[Unit]
Description="Simple service"

[Service]
Type=simple
ExecStart=/var/tmp/toexit_1simple_date.sh

[Install]
WantedBy=multi-user.target
```

 # observation: will execute printing current date and if exited successfully, system will deactivate the service afterwards.

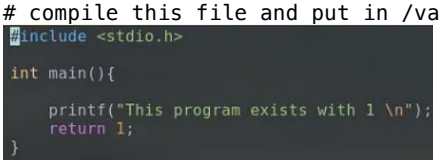
Note:
 # Restart=always
 # RestartSec=10

means no matter what the exit code is, it will restart after 10 sec. You can chain the service also. e.g. file name: dep-simple.service

```
[Unit]
Description="Dependent Simple service"
After=simple-test.service
Requires= simple-test.service

[Service]
ExecStart=/bin/bash -c "echo dependent service"
```

```
# with Restart=on-failure
# compile this file and put in /var/tmp/toexit_1
```




```
[Unit]
Description="Simple service that will have exit code of 1"

[Service]
Type=simple
ExecStart=/var/tmp/toexit_1
Restart=on-failure
RestartSec=10

[Install]
WantedBy=multi-user.target
```

systemcctl daeon-reload; systemctl start simple_exit1.service; journalctl -xeu simple_exit1.service

 # observation: the service will fail with exit code 1 >> system will restart after 10 sec and this cycle repeats

If my C code calls "abort();" instead of "return 1;" then **Restart=on-abort** will re-execute the program after it does a core dump.

 # Type=oneshot, Type=forking

 # observation: dependent service will not start until the non dependent oneshot service completes.

```
# execute something at startup and then shutdown
[Unit]
Description=Oneshot service test with ExecStop and RemainAfterExit

[Service]
Type=oneshot
RemainAfterExit=yes
ExecStart=/bin/bash -c "echo Oneshot service - start && sleep 60 && echo Oneshot service - end"
ExecStop=/bin/bash -c "echo Oneshot service - stop"

[Install]
WantedBy=multi-user.target
```

 # observation: when system starts up, it executes Execstart and while going down it executes ExecStop. RemainAfterExit=yes means it is still Loaded and Active after it exists; no means Loaded and inactive (dead)

Type=forking # system will call fork() on specified executable and then wait for the child, recommended to combine with PIDFile so system can later reliably identify the process

Example **/usr/lib/systemd/system/chronyd.service**

```
[Unit]
Description=NTP client/server
Documentation=man:chronyd(8) man:chrony.conf(5)
After=ntpdate.service sntp.service ntpd.service
Conflicts=ntpd.service systemd-timesyncd.service
ConditionCapability=CAP_SYS_TIME

[Service]
Type=forking
PIDFile=/run/chrony/chronyd.pid
EnvironmentFile=-/etc/sysconfig/chronyd
ExecStart=/usr/sbin/chronyd $OPTIONS
ExecStartPost=/usr/libexec/chrony-helper update-daemon
ExecStopPost=/usr/libexec/chrony-helper remove-daemon-state
PrivateTmp=yes
ProtectHome=yes
ProtectSystem=full

[Install]
WantedBy=multi-user.target
```

ip/ arp/and default route:

```
ip addr show | ip addr show dev eth0
ip [-s] link | ip -s link show dev eth0

ip route | ip maddr | ip madd show dev eth0




ip neigh | ip neigh show dev eth0



# send ARP request to 192.168.1.1 via eth0
arping -I eth0 192.168.1.1

# check dup MAC addr at 192.168.1.1 on eth0
arping -D -I eth0 192.168.1.1
```

```
ip addr [add|del] 192.168.1.2/24 dev eth0
ip link set eth0 mtu 9000 | ip link set eth0 up

ip route [add|del] default via 192.168.1.1 dev eth0
ip route add 192.168.1.0/24 via 192.168.1.1
ip route add 192.168.1.0/24 dev eth0
ip route get 192.168.1.5
ip maddr add 33:33:00:00:00:01 dev eth0

ip neigh add 192.168.1.1 lladdr 1.2.3.4.5.6 dev eth0
ip neigh del 192.168.1.1 dev eth0

ss -a # show all sockets
ss -e # show socket details
ss -o # show timer info
ss -p # show process using the socket
ss -neopa | same as netstat -neopa
```

Dhcp /resolve.conf /traceroute /nmcli crap:

DHCP config (/etc/sysconfig/network-scripts/ifcfg-eth0):

```
BOOTPROTO='dhcp'
NAME='eth0'
NETMASK=''
NETWORK=''

dhclient eth0 -v # To acquire a dhcp address:
```

```
# cat /etc/resolv.conf
; generated by /usr/sbin/dhclient-script
search example.com
nameserver 10.0.0.1
```

Troublshooting:

```
traceroute -I 8.8.8.8 | traceroute -n 8.8.8.8
nc -l 1.2.3.4 9000 | nc 1.2.3.4 9000
```

bond0: | **slaves:**

```
DEVICE=bond0                          |MASTER=bond0
NAME=bond0                            |SLAVE=yes
TYPE=Bond
BONDING_OPTS="mode=1 miimon=100"
```

```
man nmcli-example      # examples
/etc/NetworkManager/*  # NetworkManager config file

nmcli con show | con show <name> | dev status
nmcli con add <con-name> | mod <con-name> | del <con-name>
nmcli con up|down <name>

nmcli con add con-name eth0 type ethernet ifname eth0
ipv4.address 5.5.5.5/24 ipv4.dateway 5.5.5.254

# to add / delete IP (it writes the ifcfg-eth0 file)
nmcli con modify eth0 +ipv4.addresses 10.0.0.9/24
nmcli con up enp0s25

nmcli con modify eth0 -ipv4.addresses 10.0.0.9/24
nmcli con up enp0s25
```

```
nmcli con show
NAME         UUID                                  TYPE      DEVICE
myairport-5G 50454af6-3f91-4d17-8186-726e811f261e  wifi      wlp3s0
enp0s25      c0fd303b-45be-42c6-9abd-72d92336ee34  ethernet  --
wl3p0s3      a668adb3-5378-42d2-a935-94058bb20e6f  wifi      --
```

```
# configures enp0s25 as dhcp
nmcli connection modify enp0s25 ipv4.method auto
nmcli connection up enp0s25
```

```
dnf install  nmstate -y

enp0s25.yml
---
interfaces:
- name: enp0s25
  type: ethernet
  state: up
  ipv4:
    enabled: true
    address:
    - ip: 5.5.5.5
      prefix-length: 24
    dhcp: false

nmstatectl apply enp0s25.yml # assigns ip

# adding bond0
nmcli connection add type bond con-name bond0
ifname bond0 bond.options "mode=active-
backup,miimon=1000"

nmcli dev status

nmcli connection add type ethernet slave-type bond
con-name bond0-port1 ifname enp0s25 master bond0

nmcli connection modify bond0 ipv4.addresses
'192.0.2.1/24' ipv4.gateway '192.0.2.254'

nmcli connection up bond0

nmcli conn delete bond-port1
```

Boot process + kdump + systemd:

BIOS >> GRUB >> Kernel >> Systemd

Legacy/UEFI: Legacy BIOS is 16-bit, can address only 1MB of memory and could not boot from drive > 2.1TB. UEFI is OS-neutral software interface between firmware and OS to overcome the BIOS issue, operates in 32|64 bit mode.

Kernel panics = kernel oops. When a kernel panic occurs, **kdump utility saves crash dump to memory.** Kdump requires system memory to be reserved via boot time (grub menu, **crashkernel=auto** directive). Kexec utility boots the kernel from another kernel but does not use BIOS. You can analyze the crash using crash utility.

kexec-tools rpm and kdump.service enables kdump. **kexec** is a system call that enables you to load and boot into another kernel from the currently running kernel. **kexec** performs the function of the boot loader from within the kernel. The primary difference between a standard system boot and a **kexec** boot is that the hardware initialization normally performed by the BIOS or firmware (depending on architecture) is not performed during a **kexec** boot. This has the effect of reducing the time required for a reboot.

kexec -l /boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64 --initrd=/boot/initramfs-4.18.0-477.10.1.el8_8.x86_64.img --reuse-cmdline; systemctl kexec # points to another kermnel

The kexec command loads a new kernel and jumps directly to it, bypassing firmware and grub. It is most often used as the first step in generating a crash dump, but it can also be used to perform an administrative reboot. The time saved by skipping firmware is substantial on a server with large memory, many CPUS, and many devices. This is particularly useful during kernel development when you frequently rebuild and reboot the kernel.

/etc/kdump.conf contains **/var/crash** where the kernel state is dumped. To test: **echo c > /proc/sysrq-trigger** will dump kernel memory to /var/crash.

To start crash:
subscription-manager repos --enable rhel-8-for-x86_64-baseos-debug-rpms
dnf install -y crash kernel-debuginfo
crash /usr/lib/debug/lib/modules/4.18.0-513.9.1.el8_9.x86_64/vmlinux vmcore
crash> log | less || bt || ps || vm || vm <pid> || files || sys || sys -i (shows vendor parts as key:value with DMI_... prefix)

```
crash> bt
PID: 1914     TASK: ffff88a25b168000  CPU: 1    COMMAND: "bash"
 #0 [ffffa7da417e7cd8] machine_kexec at ffffffffa9e6d8c3
 #1 [ffffa7da417e7d30] __crash_kexec at ffffffffa9fb757a
 #2 [ffffa7da417e7df0] panic at ffffffffa9ef813f
 #3 [ffffa7da417e7e70] sysrq_handle_crash at ffffffffaa402741
 #4 [ffffa7da417e7e78] __handle_sysrq.cold.13 at ffffffffaa403064
 #5 [ffffa7da417e7ea8] write_sysrq_trigger at ffffffffaa402f0b
 #6 [ffffa7da417e7eb8] proc_reg_write at ffffffffaa1f03b9
 #7 [ffffa7da417e7ed0] vfs_write at ffffffffaa168e75
 #8 [ffffa7da417e7f00] ksys_write at ffffffffaa1690ff
 #9 [ffffa7da417e7f38] do_syscall_64 at ffffffffa9e0539b
#10 [ffffa7da417e7f50] entry_SYSCALL_64_after_hwframe at ffffffffaaa000a9
    RIP: 00007f9582d55de8  RSP: 00007ffef7f15c18  RFLAGS: 00000246
    RAX: ffffffffffffffda  RBX: 0000000000000002  RCX: 00007f9582d55de8
    RDX: 0000000000000002  RSI: 0000563082fba000  RDI: 0000000000000001
    RBP: 0000563082fba000   R8: 000000000000000a   R9: 00007f9582db5fc0
    R10: 000000000000000a  R11: 0000000000000246  R12: 00007f9582ff66e0
    R13: 0000000000000002  R14: 00007f9582ff1860  R15: 0000000000000002
    ORIG RAX: 0000000000000001  CS: 0033  SS: 002b
```

crash> dis -rl ffffffff8111ede6 | tail

mod -t # shows tainted modules

example: I had cpu exception found via crash >> **log | less**

grep mcelog sos_commands/systemd/systemctl_list-units

Boot troubleshooting:

#1: Boot to alternative kernel
Root cause: Updated new kernel with yum, experiencing issues ..
Todo:
Boot to previous kernel from the grub splash screen
grubby --default-kernel
/boot/vmlinuz-4.18.0-513.9.1.el8_9.x86_64

grubby --info=ALL | grep title
title="Red Hat Enterprise Linux (4.18.0-513.9.1.el8_9.x86_64) 8.9 (Ootpa)" <<< 0
title="Red Hat Enterprise Linux (4.18.0-477.10.1.el8_8.x86_64) 8.8 (Ootpa)". <<< 1
title="Red Hat Enterprise Linux (0-rescue-48a3c080fb5b40b3b3100af7e604437c) 8.8 (Ootpa)". <<< 2

grub2-set-default 1 OR
grubby --set-default-index=1 OR
grubby --set-default /boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64

```
grubby --default-kernel
/boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64

Reboot

Check grubenv file:
grub2-editenv /boot/grub2/grubenv list  # for Legacy

saved_entry=1
kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet
systemd.unified_cgroup_hierarchy=1
boot_success=0

grub2-editenv /boot/efi/EFI/redhat/grubenv list   # for EFI

saved_entry=1
kernelopts=root=/dev/mapper/rhel-root ro crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet
systemd.unified_cgroup_hierarchy=1
boot_success=0



There are 2 ways to change the default kernel. Each kernel info exists in /boot/loader/entries.

ls -l /boot/loader/entries/
total 12
-rw-r--r--. 1 root root 408 Nov 30 08:38 48a3c080fb5b40b3b3100af7e604437c-0-rescue.conf
-rw-r--r--. 1 root root 371 Nov 30 08:38 48a3c080fb5b40b3b3100af7e604437c-4.18.0-477.10.1.el8_8.x86_64.conf
-rw-r--r--  1 root root 366 Dec 19 18:05 48a3c080fb5b40b3b3100af7e604437c-4.18.0-513.9.1.el8_9.x86_64.conf


You can do
grub2-set-default 48a3c080fb5b40b3b3100af7e604437c-4.18.0-477.10.1.el8_8.x86_64.conf

If the /proc/cmdline is not taking effect after doing grub2-mkconfig -o /boot/grub/grub2.cfg then look for the file
/boot/loader/entries/51…3d-4.18……el8-8.x86_64.conf file, it should have this line ..

options $kernelopts $tuned_params


To confirm default kernel is reflected:

grub2-editenv /boot/grub2/grubenv list (for Legacy) | grub2-editenv /boot/efi/EFI/redhat/grubenv list (for UEFI)

confirm these ..
GRUB_DEFAULT=saved in cat /etc/sysconfig/grub

grubby --default-kernel
/boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64

Boot troubleshooting, cont..

grubby --info /boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64

index=1
kernel="/boot/vmlinuz-4.18.0-477.10.1.el8_8.x86_64"
args="ro crashkernel=auto rd.lvm.lv=rhel/root rhgb quiet systemd.unified_cgroup_hierarchy=1 $tuned_params"
root="/dev/mapper/rhel-root"
initrd="/boot/initramfs-4.18.0-477.10.1.el8_8.x86_64.img $tuned_initrd"
title="Red Hat Enterprise Linux (4.18.0-477.10.1.el8_8.x86_64) 8.8 (Ootpa)"
id="48a3c080fb5b40b3b3100af7e604437c-4.18.0-477.10.1.el8_8.x86_64"



#2: Huge page misconfigured in grub command line, total value of huge pages has surpassed amount of RAM installed.

At boot time ‘e’ to edit and then just do crtl+x >> edit GRUB_CMDLINE_LINUX in /etc/default/grub >> grub2-mkconfig
-o /boot/grub2/grubenv


#3: Huge page misconfigured in configuration file
At boot time ‘e’ to edit >> add systemd.mask=systemd-sysctl.service rd.systemd.unit=emergency.target >> crtl+x

Remove hugepage entries from following files:
/etc/sysctl.conf
/etc/sysctl.d/*.conf
/usr/lib/sysctl.d/*.conf
```

```
/lib/sysctl.d/*.conf
/usr/local/lib/sysctl.d/*.conf

Reboot >> 'e' to edit >> delete "systemd.mask=systemd-sysctl.service rd.systemd.unit=emergency.target" >> crtl+x

#4: after update kernel panic at boot, unable to mount root fs, unknown block(0,0)
Boot from older kernel >> yum remove kernel-core-<newversion>-<release>.<arch> >> yum install kernel-core-
<newversion>-<release>.<arch>
Ensure initrd is created in /boot/ and initrd statement is present in /boot/grub/grub.conf


#5: System is dropping to maintenance mode during boot, /etc/fstab is corrupted entry

Give the root password for maintenance
(Or press Control-D to continue): <<< give root pwd >>

mount -o remount, rw /

lvm lvs
lvm vgchange -ay
mount -a -v

vi /etc/fstab and correct the entry
systemctl get-default


#6: repair file system in rescue environment
Boot from rescue menu item of the boot menu or from .iso file in ilo
lvm vgchange -ay
xfs_repair /dev/sdb1 (or /dev/vg01/vol1)
exit >> reboot

#7: system shuts down before completing boot. It had separate /var file, was full
Boot in single user mode and clean up the space

#8: how to boot in emergency mode (Works!)

At boot time 'e' to edit >> add systemd.unit=emergency.target >> crtl+x >> give root pwd
You can do df -h
mount -o remount -o rw /


OR

grubby -args="system.unit=emergency.target" --update-kernel=/boot/ vmlinuz-4.18.0-513.9.1.el8_9.x86_64
systemctl reboot
Maintenance-prompt> << give pwd
fsck /boot
mount -o remount -o rw /
mount /boot

Then after maintenance is done ..
grubby -remove-args="="system.unit=emergency.target" --update-kernel=/boot/ vmlinuz-4.18.0-513.9.1.el8_9.x86_64
reboot

#9: how to boot in emergency mode (works!!)
At boot menu press 'e' >> add rd.break >> crtl + x >> it will ask for root pwd >> proceed as usual
Or
At boot prompt add init=/bin/bash >> crtl + x >> it will ask for root pwd >> proceed as usual
chroot /sysroot
mount -o remount -o rw /
<change the root passwod>
sync
reboot
```

Cgroups

```
# To control user not to use more than 10% of CPU
/etc/systemd/system/user-1000.slide.d/CPUQuota.conf
[Slice]
CPUQuota=10%

# Change it (not persistent upon reboot)
systemctl set-property user-1000.slice CPUQuota=50%

# For trading
/etc/systemd/system.control/system.slice.d/
```

```
# Create persistent cgroup and start all process under it
/etc/systemd/system/hog_pen.slice
[Slice]
CPUQuota=50%   # cumulative sum of all process max 50%
MemoryMax=100M # cumulative sum of all process max 100M

systemctl daemon-reload >>
systemd-run -u hog1 --slice=hog_pen.slice ~/hog1
systemd-run -u hog2 --slice=hog_pen.slice ~/hog2
```

50-AllowedCPUs.conf **[Unit]** **Description=Testing cgroup V2** **DefaultDependencies=no** **Before=slices.target** **[Slice]** **AllowedCPUs=0-2** **/etc/systemd/system.control/user.slice.d/** 50-AllowedCPUs.conf **/etc/systemd/system.control/user-.slice.d/** 50-AllowedCPUs.conf	**docker run -d --name web --cgroup-parent=hog_pen.slice nginx** **docker run -d --name web --cgroup-parent=hog_pen.slice redis** # Pids are in /sys/fs/cgroup/system.slice/docker.service/cgroup.procs **system-cgtop >> system-cgls** # to monitor

Kerberos single-sign-on setup steps:

A **realm** is a logical network, similar to a domain, that defines a group of systems under the same **master KDC.** Realms can relate to one another. Some realms are hierarchical, where one realm is a superset of the other realm. Otherwise, the realms are nonhierarchical (or "direct") and the mapping between the two realms must be defined. Kerberos **cross-realm authentication** enables authentication across realms. Individually, the terms **'domain'** and **'realm'** mean nearly the same thing, but for different systems. Realms and realm names come from the Kerberos authentication protocol, where they serve practically the same purpose as domains and domain names. Kerberos realm name *is* always case-sensitive and by convention always uppercase. Each Active Directory domain **acts as** a Kerberos realm, and has exactly one realm name.

Keytab is a file containing **pairs** of **Kerberos principals** and **encrypted keys** that are derived from the **Kerberos password.** It is a **representation of the service and its long-term key.** It is **used de-crypt the Kerberos service ticket** of an **inbound AD user** to the service OR **authenticate the service itself to another service** on the network. a service cannot manually type in it's password to authenticate itself, so the long-term key is helpfully encoded into the file. This is why the keytab file itself is sensitive and needs to be protected.

kadmin.local >> addprinc --randkey HTTP://server.example.com >> ktadd -k /etc/krb5.keytab HTTP://server.example.com
kadmin.local -q "addprinc --randkey host/aap01.example.com"

```
# tpx230: KDC server
# aap01: Kerberos enabled application service server
# msirtx4050: end user server who wants a service from aap01
# make sure that /etc/hosts of each host can resolve and time is synched.
# kinit <username>@hostname.com | klist -a | klist -v | kdestroy  # Kerberos commands
# Do these..

Tpx230:
dnf install krb5-server krb5-libs krb5-workstation   <<< rpms needed

vi /var/kerberos/krb5kdc/kdc.conf

[kdcdefaults]
    kdc_ports = 88
    kdc_tcp_ports = 88
    spake_preauth_kdc_challenge = edwards25519

[realms]
EXAMPLE.COM = {
     #master_key_type = aes256-cts
     acl_file = /var/kerberos/krb5kdc/kadm5.acl
     dict_file = /usr/share/dict/words
     admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
     supported_enctypes = aes256-cts:normal aes128-cts:normal arcfour-hmac:normal camellia256-cts:normal
camellia128-cts:normal
}

# to reflect the realm name and domain-to-realm mappings
vi /etc/krb5.conf

# To opt out of the system crypto-policies configuration of krb5, remove the
# symlink at /etc/krb5.conf.d/crypto-policies which will not be recreated.
includedir /etc/krb5.conf.d/

[logging]
    default = FILE:/var/log/krb5libs.log
    kdc = FILE:/var/log/krb5kdc.log
    admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    dns_lookup_realm = false
    ticket_lifetime = 24h
```

```
    renew_lifetime = 7d
    forwardable = true
    rdns = false
    pkinit_anchors = FILE:/etc/pki/tls/certs/ca-bundle.crt
    spake_preauth_groups = edwards25519
    default_realm = EXAMPLE.COM
    default_ccache_name = KEYRING:persistent:%{uid}

[realms]
 EXAMPLE.COM = {
     kdc = tpx230.example.com
     admin_server = tpx230.example.com
 }

[domain_realm]
 .example.com = EXAMPLE.COM
 example.com = EXAMPLE.COM


# create database
kdb5_util create -s  # supply the database random password

# configure Kerberos ACL
vi /var/kerberos/krb5kdc/kadm5.acl
*/admin@EXAMPLE.COM    *

# Create the first principal using kadmin.local.  <<< adding principal
kadmin.local
..
kadmin.local:  addprinc dbaplus/admin
..
kadmin.local:  exit
Kerberos single-sign-on setup steps (Cont.)

# start Kerberos
systemctl start krb5kdc.service
systemctl start kadmin.service

# verify
netstat -plntu

kerberos  tcp/88 & udp/88    (Kerberos network authentication protocol server - KDC)
kadmin    tcp/749            (Kerberos Administration Protocol)
kpasswd   tcp/464 & udp/464  (Kerberos password server)


# verify that KDC is working, run kinit to get a ticket from kdc
kinit dbaplus/admin

klist

# if you have firewalld running then add these..
firewall-cmd --add-service=kerberos --permanent
firewall-cmd --add-service=kadmin --permanent
firewall-cmd --add-service=kpasswd --permanent
systemctl restart firewalld.service

aap01:
# install rpms
yum install krb5-workstation krb5-libs

# copy /etc/krb5.conf
scp tpx230:/etc/krb5.conf /etc/krb5.conf

tpx230:
# add the host principal for each such server in KDC
kadmin.local -q "addprinc --randkey host/aap01.example.com"

aap01:
# run kinit ro obtain initial ticket from KDC server
kinit dbaplus/admin

# Run kadmin >> ktadd on server aap01 to download principle "host/aap01.example.com" created before at KDC svr
kadmin -q "ktadd host/aap01.example.com"

# run kadmin command ktadd
kadmin
```

```
kadmin: ktadd -k /etc/krb5.keytab host/aap01.example.com.   <<< keytab

OR

kutil
kutil: add_entry -password -p user01/example.com -k 1 -e des3-cbc-shha1-kd




# Edit file /etc/ssh/sshd_config and add following lines,
 KerberosAuthentication yes
 KerberosTicketCleanup yes
 KerberosOrLocalPasswd yes

* KerberosAuthentication  Specifies whether the password provided by the user for PasswordAuthentication will be
validated through the Kerberos KDC.  To use this option, the server needs a Kerberos keytab which allows the
verification of the KDC's identity.

* KerberosTicketCleanup   Specifies whether to automatically destroy the user's ticket cache file on logout.

* KerberosOrLocalPasswd   If password authentication through Kerberos fails then the password will be validated via
       any additional local mechanism such as  /etc/passwd.



# Restart sshd
systemctl restart sshd

# create user user01
useradd user01

passwd --status user01  << it should show locked

tpx230:
# Create principals user01 & user02 on KDC
kadmin.local -q "addprinc user01"



msirtx4050:
# install Kerberos client
yum install -y krb5-workstation krb5-devel krb5-libs

scp krb5.conf to /etc/krb5.conf  ## verify this


# Create principal for workstation
kadmin -p dbaplus/admin


kadmin:  addprinc -randkey host/wkstn01.lab.dbaplus.ca
kadmin:  ktadd host/wkstn01.lab.dbaplus.ca

kadmin:  q

# create user
useradd user01

# now from mac, login to msirtx4050
user01@ msirtx4050

# do kinit
kinit
klist

# loginto aap01 to get the service, no password needed
ssh aap01

Bingo!!
```