Understanding systemd

Patrick Ladd
Technical Account Manager
Red Hat Software
pladd@redhat.com
What is systemd?

- Replaces init
  - Literally!

  ```bash
  [root@rhel7 ~]# ls -al /sbin/init
  lrwxrwxrwx. 1 root root 22 Jan 27 13:43 /sbin/init -> ../../../lib/systemd/systemd
  [root@rhel7 ~]#  
  ```

- First process to start and last to stop
- Parent process of all other processes
- Manages services and other resources
What was init again?

- init – System V UNIX origins in 1970s
- Process for starting system:
  - BIOS/UEFI → Bootloader → Kernel → init
- init is the parent of all processes
- Creates processes from scripts stored in /etc/inittab
- “Modern” init scripts are stored in /etc/init.d and called from /etc/rc*
Why replace System V init?

- init scripts!
  - Old, poorly maintained
  - Lack of standardization
  - Difficult / impossible to analyze (by humans and/or computers)

- Single threaded

- Unable to represent complex relationships
. /etc/rc.d/init.d/functions
if [ -f /etc/sysconfig/httpd ]; then
  . /etc/sysconfig/httpd
fi
HTTPD_LANG=${HTTPD_LANG-"C"}
INITLOG_ARGS=""
apachectl=/usr/sbin/apachectl
httpd=${HTTPD-/usr/sbin/httpd}
prog=httpd
pidfile=${PIDFILE-/var/run/httpd/httpd.pid}
lockfile=${LOCKFILE-/var/lock/subsys/httpd}
RETVAL=0
STOP_TIMEOUT=${STOP_TIMEOUT-10}

start() {
  echo -n "$\text{Starting }$$prog:  "
  LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && touch ${lockfile}
  return $RETVAL
}

stop() {
  echo -n "$\text{Stopping }$$prog:  "
  killproc -p ${pidfile} -d ${STOP_TIMEOUT} $httpd
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
}
reload() {
    echo -n "$prog: "
    if ! LANG=$HTTPD_LANG $httpd $OPTIONS -t >&/dev/null; then
        RETVAL=6
        echo "$not reloading due to configuration syntax error"
        failure "$not reloading $httpd due to configuration syntax error"
    else
        LSB=1 killproc -p ${pidfile} $httpd -HUP
        RETVAL=$?
        if [ $RETVAL -eq 7 ]; then
            failure "$httpd shutdown"
        fi
    fi
    echo
}

case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    status)
        status -p ${pidfile} $httpd
        RETVAL=$?
        ;;

    *)
        echo "Invalid command: $1"
        ;;
esac
start
condrestart|try-restart)
    if status -p ${pidfile} $httpd >/dev/null; then
        stop
        start
    fi
force-reload|reload)
    reload
graceful|help|configtest|fullstatus)
    $apachectl $@
    RETVAL=$?
*)
    echo "$Usage: $prog {start|stop|restart|condrestart|try-restart|force-reload|reload|status|fullstatus|graceful|help|configtest}"
    RETVAL=2
esac
exit $RETVAL
systemd: httpd.service

[Unit]
Description=The Apache HTTP Server
After=remote-fs.target nss-lookup.target

[Service]
Type=notify
EnvironmentFile=/etc/sysconfig/httpd
ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
ExecStop=/usr/sbin/httpd $OPTIONS -k graceful-stop

KillSignal=SIGCONT
PrivateTmp=true

[Install]
WantedBy=multi-user.target
So long, and thanks for all the fish

RIP
Sys V
init
1970-2011
systemd Overview

- Controls More than Services
- Dependency Control
- Tracks and Restarts Services
- Service Activation
- Faster Start Up and Shutdown
- Improved Resource Management
- Better Logging, Debugging and Profiling
- Backwards compatible
- Easier to learn
systemd Units

Controls more than services, it controls all resources on the system - referred to as units.

Examples of Units:

<table>
<thead>
<tr>
<th>Services</th>
<th>Sockets</th>
<th>Mounts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Targets</td>
<td>Swap</td>
<td>and more...</td>
</tr>
</tbody>
</table>

Units are defined using Unit Files

- Naming convention is name.unit_type
systemd Unit Files

• Maintainer files: /usr/lib/systemd/system
• Administrator files: /etc/systemd/system
• Non-persistent, runtime data: /run/systemd
• Drop-ins: /etc/systemd/system/[name.type].d/*.conf

Note: unit files under /etc will take precedence over /usr

Don't forget `systemctl daemon-reload` when modifying units.
Common Unit File Options

- Description=Unit description
- Documentation=Documentation links
- Requires=Additional units required
- Before/After=Unit must start Before/After
- Wants=Weaker Requires
- Conflicts=Units cannot co-exist
- WantedBy/RequiredBy=Set other units requirement

- Lots of great detail in the RHEL 7 System Administrator's Guide
Service Activation

• Start up services when needed
  – Save resources
  – Increased reliability
  – Transparent to client

• Activation by Socket, Device, Path, Bus, and Timer

• Recommended to convert xinetd services to units
Improved Resource Management

- Services labeled and isolated with Cgroups
- More control than nice alone
- Can properly kill/restart entire service chain
- Can configure multiple instances for a single service
- Can balance by shares or by hard limits
Kill/Restart Cleanly

- Tracked in the kernel
- Knows all children
- Don’t need to rely on a potentially misbehaving process to hopefully kill its children
Auto-Restarting

• It’s paying attention!
• Reality: software does crash occasionally
• Reduces need for manual intervention
• Socket stays open, only lose that single transaction
With init:
$ service unit {start,stop,restart,reload}

With systemd:
$ systemctl {start,stop,restart,reload} unit1 [unit2 ...]

- Allows multiple services to be acted on simultaneously
- Assumes .service as unit type
- Tab completion works great with systemctl
  • Install bash-completion
systemctl vs service

```
[root@rhel6 ~]# service hellod start
Starting Hello Server: [ OK ]
[root@rhel6 ~]# service hellod status
hellod (pid 2889) is running...
[root@rhel6 ~]# service hellod stop
Stopping Hello Server: [ OK ]
[root@rhel6 ~]#
```

```
[root@rhel7 ~]# systemctl start hellod.service
[root@rhel7 ~]# systemctl status hellod.service
hellod.service - Hello Service
   Loaded: loaded (/etc/systemd/system/hellod.service; enabled)
   Active: **active (running)** since Tue 2015-02-17 18:29:25 EST; 5s ago
   Process: 3941 ExecStart=/usr/local/sbin/hellod (code=exited, status=0/SUCCESS)
   Main PID: 3943 (hellod)
   CGroup: /system.slice/hellod.service
          └─3943 /usr/local/sbin/hellod

Feb 17 18:29:25 rhel7.mruzicka systemd[1]: Starting Hello Service...
[root@rhel7 ~]# systemctl stop hellod.service
[root@rhel7 ~]#
```
systemctl vs service

- List services:

[root@rhel6 ~]# service --status-all
abrt-ccpp hook is installed
abrtd (pid 1652) is running...
abrt-dump-oops is stopped
acpid (pid 1440) is running...
atd (pid 1675) is running...
auditd (pid 1106) is running...
automount (pid 1518) is running...
certmonger (pid 1704) is running...
Stopped
cgred is stopped

[root@rhel7 ~]# systemctl --type service --state active
<table>
<thead>
<tr>
<th>UNIT</th>
<th>LOAD</th>
<th>ACTIVE</th>
<th>SUB</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>abrt-ccpp.service</td>
<td>loaded</td>
<td>active</td>
<td>exited</td>
<td>Install ABRT coredump hook</td>
</tr>
<tr>
<td>abrt-oops.service</td>
<td>loaded</td>
<td>active</td>
<td>running</td>
<td>ABRT kernel log watcher</td>
</tr>
<tr>
<td>abrt-xorg.service</td>
<td>loaded</td>
<td>active</td>
<td>running</td>
<td>ABRT Xorg log watcher</td>
</tr>
<tr>
<td>abrtd.service</td>
<td>loaded</td>
<td>active</td>
<td>running</td>
<td>ABRT Automated Bug Reporting</td>
</tr>
<tr>
<td>accounts-daemon.service</td>
<td>loaded</td>
<td>active</td>
<td>running</td>
<td>Accounts Service</td>
</tr>
<tr>
<td>alsal-state.service</td>
<td>loaded</td>
<td>active</td>
<td>running</td>
<td>Manage Sound Card State (res</td>
</tr>
</tbody>
</table>
Managing Services: Enable / Disable

With init:
$ chkconfig unit {on,off}

With systemctl:
$ systemctl {enable, disable, mask, unmask} unit [unit...]

mask — “This will link these units to /dev/null, making it impossible to start them. This is a stronger version of disable, since it prohibits all kinds of activation of the unit, including manual activation. Use this option with care.”
# Systemctl vs chkconfig

**List all services:**

```
[root@rhel6 ~]# chkconfig --list
abrt-ccpp       0:off 1:off 2:off 3:on 4:off 5:on 6:off
abrtd           0:off 1:off 2:off 3:on 4:off 5:on 6:off
acpid           0:off 1:off 2:on  3:on 4:on 5:on 6:off
atd             0:off 1:off 2:off 3:on 4:on 5:on 6:off
auditd          0:off 1:off 2:on  3:on 4:on 5:on 6:off
autofs          0:off 1:off 2:off 3:on 4:on 5:on 6:off
blk-availability 0:off 1:on 2:off 3:on 4:on 5:on 6:off
certmonger      0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

```
[root@rhel7 ~]# systemctl list-unit-files --type=service
UNIT FILE            STATE
abrt-ccpp.service    enabled
abrt-oops.service    enabled
abrt-pstoreoops.service disabled
abrt-vmcore.service  enabled
abrt-xorg.service    enabled
abrtd.service        enabled
accounts-daemon.service enabled
alsa-restore.service static
alsa-state.service   static
```
### systemctl

Lots of options...

<table>
<thead>
<tr>
<th>Command</th>
<th>Option</th>
<th>Command</th>
<th>Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>cancel</td>
<td>is-active</td>
<td>reload-or-restart</td>
<td></td>
</tr>
<tr>
<td>condreload</td>
<td>is-enabled</td>
<td>reload-or-try-restart</td>
<td></td>
</tr>
<tr>
<td>condrestart</td>
<td>is-failed</td>
<td>rescue</td>
<td></td>
</tr>
<tr>
<td>condstop</td>
<td>isolate</td>
<td>reset-failed</td>
<td></td>
</tr>
<tr>
<td>daemon-reexec</td>
<td>kexec</td>
<td>restart</td>
<td></td>
</tr>
<tr>
<td>daemon-reload</td>
<td>kill</td>
<td>set-default</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>link</td>
<td>set-environment</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td>list-dependencies</td>
<td>set-property</td>
<td></td>
</tr>
<tr>
<td>disable</td>
<td>list-jobs</td>
<td>show</td>
<td></td>
</tr>
<tr>
<td>emergency</td>
<td>list-sockets</td>
<td>show-environment</td>
<td></td>
</tr>
<tr>
<td>enable</td>
<td>list-unit-files</td>
<td>snapshot</td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td>list-units</td>
<td>start</td>
<td></td>
</tr>
<tr>
<td>force-reload</td>
<td>mask</td>
<td>status</td>
<td></td>
</tr>
<tr>
<td>get-default</td>
<td>poweroff</td>
<td>stop</td>
<td></td>
</tr>
<tr>
<td>halt</td>
<td>preset</td>
<td>suspend</td>
<td></td>
</tr>
<tr>
<td>help</td>
<td>reboot</td>
<td>try-restart</td>
<td></td>
</tr>
<tr>
<td>hibernate</td>
<td>reenable</td>
<td>unmask</td>
<td></td>
</tr>
<tr>
<td>hybrid-sleep</td>
<td>reload</td>
<td>unset-environment</td>
<td></td>
</tr>
</tbody>
</table>
Lots of new commands...

```
[root@rhel7 ~]# systemctl
systemd-analyze       systemd-loginctl
systemd-ask-password  systemd-machine-id-setup
systemd-cat           systemd-notify
systemd-cgls          systemd-nspawn
systemd-cgtop         systemd-run
systemd-coredumpctl   systemd-stdio-bridge
systemd-delta         systemd-sysv-convert
systemd-detect-virt   systemd-tmpfiles
systemd-inhibit       systemd-tty-ask-password-agent
```
systemd Dependencies

• Define order and requirements for each unit

• Example: nfs-lock.service
  Requires=rpcbind.service network.target
  After=network.target named.service rpcbind.service
  Before=remote-fs-pre.target

• No more semi-arbitrary 00-99 ASCII order loading
Parallel, Not Serial

- Allows for Faster Start Up and Shutdown
- Efficiently Use System Resources

```
[root@rhel7 ~]# systemctl-analyze plot > /tmp/boot_plot.svg
```
Boot Process

- Boot path determined by default.target

Let’s track it backwards!

[root@rhel7 ~]# systemctl get-default
graphical.target

[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/graphical.target

[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
After=multi-user.target
Conflicts=rescue.target
Wants=display-manager.service
AllowIsolate=yes
Boot Process

• graphical.target requires multi-user.target...

[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/multi-user.target

[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=systemd.special
Conflicts=systemd.lease.target
After=systemd.lease.target
AllowIsolate=yes
Boot Process

• Which requires basic.target...

[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/basic.target

[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target slices.target
After=sysinit.target sockets.target timers.target paths.target slices.target

• Which requires sysinit.target...

[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/sysinit.target

[Unit]
Description=System Initialization
Documentation=man:systemd.special(7)
Conflicts=emergency.service emergency.target
Wants=local-fs.target swap.target
After=local-fs.target swap.target emergency.service emergency.target
Boot Process

Which wants local-fs-pre.target and swap.target...

```
[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/local-fs-pre.target
[Unit]
Description=Local File Systems (Pre)
Documentation=man:systemd.special(7)
RefuseManualStart=yes
[root@rhel7 ~]# grep -v '^#' /usr/lib/systemd/system/swap.target
[Unit]
Description=Swap
Documentation=man:systemd.special(7)
```

- End of the line!
Targets then loaded from the beginning..

But, how does this work for starting individual services?
Boot Process – Services/Units

- Target “Wants” Directories:
  /usr/lib/systemd/system/<name>.target.wants/
  /etc/systemd/system/<name>.target.wants/

- Files are symlinks to actual unit files
- Empty target wants directories are placeholders
Example for multi-user.target.wants:

```bash
[root@rhel7 ~]# ls /usr/lib/systemd/system/multi-user.target.wants
brandbot.path  Plymouth-quit.service  systemd-logind.service
dbus.service   Plymouth-quit-wait.service  systemd-user-sessions.service
getty.target   Systemd-ask-password-wall.path
[root@rhel7 ~]# ls /etc/systemd/system/multi-user.target.wants
abrt-ccpp.service  hypervkvpd.service  postfix.service
abrt.service       hypervvssd.service  remote-fs.target
abrt-oops.service  irqbalance.service  rhsmcertd.service
abrt-vmcore.service kdump.service     rngd.service
abrt-xorg.service  ksm.service       rsyslog.service
atd.service        ksmtuned.service   smartd.service
auditd.service     libstoragemgmt.service sshd.service
avahi-daemon.service libvirtd.service   sysstat.service
chronyd.service    mariadb.service    tuned.service
cron.service       mdmonitor.service  vmtoolsd.service
cups.path          ModemManager.service
httpd.service      NetworkManager.service
```
Exploring dependencies

List all services by target:

```
[root@rhel7 ~]# systemctl list-dependencies multi-user.target --no-pager
multi-user.target
├── abrt-ccpp.service
├── abrt-oops.service
├── abrt-vmcore.service
├── basic.target
│   ├── alsa-restore.service
│   ├── alsa-state.service
│   ├── paths.target
│   │   ├── slices.target
│   │   │   └── .slice
│   │   │       └── system.slice
│   │   ├── sockets.target
│   │   │   └── avahi-daemon.socket
│   │   └── cups.socket
│   └── timers.target
│       └── systemd-tmpfiles-clean.timer
├── getty.target
│   └── getty@tty1.service
└── remote-fs.target
```
Analyzing Boot

- Each unit is tracked during start up

```
[root@rhel7 ~]# systemd-analyze blame --no-pager
  2.598s  mariadb.service
  1.459s  kdump.service
  868ms  Plymouth-quit-wait.service
  867ms  postfix.service
  510ms  firewalld.service
  397ms  network.service
  380ms  httpd.service
  347ms  boot.mount
  311ms  tuned.service
  245ms  lvm2-monitor.service
  237ms  libvirtd.service
  232ms  accounts-daemon.service
  203ms  systemd-vconsole-setup.service
  203ms  ModemManager.service
  168ms  avahi-daemon.service
  167ms  systemd-logind.service
  156ms  rtkit-daemon.service
  127ms  chronyd.service
```
Targets are the new Runlevels

Targets != Runlevels – some equivalency

<table>
<thead>
<tr>
<th>Traditional Runlevel</th>
<th>Equivalent Target</th>
<th>Symlink Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runlevel 0</td>
<td>poweroff.target</td>
<td>runlevel0.target</td>
</tr>
<tr>
<td>Runlevel 1</td>
<td>rescue.target</td>
<td>runlevel1.target</td>
</tr>
<tr>
<td>Runlevel 2</td>
<td>multi-user.target</td>
<td>runlevel2.target</td>
</tr>
<tr>
<td>Runlevel 3</td>
<td>multi-user.target</td>
<td>runlevel3.target</td>
</tr>
<tr>
<td>Runlevel 4</td>
<td>multi-user.target</td>
<td>runlevel4.target</td>
</tr>
<tr>
<td>Runlevel 5</td>
<td>graphical.target</td>
<td>runlevel5.target</td>
</tr>
<tr>
<td>Runlevel 6</td>
<td>reboot.target</td>
<td>runlevel6.target</td>
</tr>
</tbody>
</table>

- Targets can and will contain other targets
## Common Targets

<table>
<thead>
<tr>
<th>Target</th>
<th>Purpose</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>graphical.target</td>
<td>Supports multiple users, graphical and text-based logins</td>
<td></td>
</tr>
<tr>
<td>multi-user.target</td>
<td>Supports multiple users, text-based logins only</td>
<td></td>
</tr>
<tr>
<td>rescue.target</td>
<td>Single user, local file systems mounted and basic system initialization completed, networking is not activated</td>
<td></td>
</tr>
<tr>
<td>emergency.target</td>
<td>Single user, root file system is mounted read-only, only a few essential services are started, networking is not activated</td>
<td></td>
</tr>
</tbody>
</table>

- Rescue and Emergency require root password!
Working with Targets

Viewing the default target:

```
[root@rhel7 ~]# systemctl get-default
multi-user.target
```

Setting default target:

```
[root@rhel7 ~]# systemctl set-default graphical.target
rm '/etc/systemd/system/default.target'
ln -s '/usr/lib/systemd/system/graphical.target' '/etc/systemd/system/default.target'
```

Default target is just a symlink:

```
[root@rhel7 ~]# ls -al /etc/systemd/system/default.target
lrwxrwxrwx. 1 root root 40 Feb 22 21:17 /etc/systemd/system/default.target -> /usr/lib/systemd/system/graphical.target
```
Working with Targets

Changing currently loaded target:

```
[root@rhel7 ~]# systemctl isolate graphical.target
[root@rhel7 ~]#
```

Changing to rescue mode:

```
[root@rhel7 ~]# systemctl rescue
Broadcast message from mruzicka@rhel7.mruzicka on pts/0 (Sat 2015-02-14 19:48:43 EST):
The system is going down to rescue mode NOW!
```

Changing to emergency mode without sending message:

```
[root@rhel7 ~]# systemctl --no-wall emergency
```
Working with Targets

View list of currently loaded targets:

```
[root@rhel7 ~]# systemctl list-units --type=targets
```

Results pipe to less by default: (can use --no-pager)

<table>
<thead>
<tr>
<th>UNIT</th>
<th>LOAD</th>
<th>ACTIVE</th>
<th>SUB</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>basic.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Basic System</td>
</tr>
<tr>
<td>cryptsetup.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Encrypted Volumes</td>
</tr>
<tr>
<td>getty.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Login Prompts</td>
</tr>
<tr>
<td>local-fs-pre.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Local File Systems (Pre)</td>
</tr>
<tr>
<td>local-fs.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Local File Systems</td>
</tr>
<tr>
<td>multi-user.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Multi-User System</td>
</tr>
<tr>
<td>network.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Network</td>
</tr>
<tr>
<td>paths.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Paths</td>
</tr>
<tr>
<td>remote-fs.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Remote File Systems</td>
</tr>
<tr>
<td>slices.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Slices</td>
</tr>
<tr>
<td>sockets.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Sockets</td>
</tr>
<tr>
<td>sound.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Sound Card</td>
</tr>
<tr>
<td>swap.target</td>
<td>loaded</td>
<td>active</td>
<td>active</td>
<td>Swap</td>
</tr>
</tbody>
</table>

Lines 1-14/23 61%
### Shutting Down, Suspending, Etc.

<table>
<thead>
<tr>
<th>Old Command</th>
<th>New Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>halt</td>
<td>systemctl halt</td>
<td>Halts the system</td>
</tr>
<tr>
<td>poweroff</td>
<td>systemctl poweroff</td>
<td>Powers off the system</td>
</tr>
<tr>
<td>reboot</td>
<td>systemctl reboot</td>
<td>Restarts the system</td>
</tr>
<tr>
<td>pm-suspend</td>
<td>systemctl suspend</td>
<td>Suspends the system</td>
</tr>
<tr>
<td>pm-hibernate</td>
<td>systemctl hibernate</td>
<td>Hibernates the system</td>
</tr>
<tr>
<td>pm-suspend-hybrid</td>
<td>systemctl hybrid-sleep</td>
<td>Hibernates and suspends the system</td>
</tr>
</tbody>
</table>

[root@rhel7 ~]# ls -al /usr/sbin/shutdown
lrwxrwxrwx. 1 root root 16 Feb 13 17:00 /usr/sbin/shutdown -> ../bin/systemctl
[root@rhel7 ~]#
Resetting Root Password

Step 1: Reboot your system

Step 2: When the GRUB2 boot loader menu appears, press any key to interrupt, and press ‘e’ to edit selection
Resetting Root Password

Step 3: Using the cursor keys, navigate to the end of the line that starts with linux16

Step 4: Append ‘rd.break’ to the end of the line and press Ctrl-X to boot modified config

```
inmod part_msdos
inmod xfs
set root='hd0,msdos1'
if [ x$feature_platform_search_hint = xy ]; then
    search --no-floppy --fs-uuid --set=root --hint='hd0,msdos1' dfa2f4c
e-ab99-4e78-bc96-ebb73fa2922c
else
    search --no-floppy --fs-uuid --set=root dfa2f4ce-ab99-4e78-bc96-ebb7
3fa2922c

fi
linux16 /vmlinux-3.10.0-229.el7.x86_64 root=/dev/mapper/rhel_rhel7-root
trocrashkernel=auto rd.lvm.lv=rhel_rhel7/root rd.lvm.lv=rhel_rhel7/swap rhgb
quiet LANG=en_US.UTF-8 rd.break
initrd16 /initramfs-3.10.0-229.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.
Resetting Root Password

Step 6: At the switch_root prompt, remount the /sysroot file system read-write, then use chroot to enter the jail.

Step 7: Reset the root password with passwd.

```
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:
```
Resetting Root Password

IMPORTANT! (Because you’re using SELinux, right?)

Step 8: Instruct SELinux to relabel all files upon reboot.
- System may not boot properly if skipped!

Step 9: Exit twice to initiate relabel and reboot

```
sh-4.2# touch /.autorelabel
sh-4.2# exit
exit
switch_root:/># exit
logout
```

NOTE: The relabel may take quite a while to run before the system becomes available.
systemd-cgtop

Show top control groups by their resource usage:

```
[root@rhel7 ~]# systemd-cgtop
```

<table>
<thead>
<tr>
<th>Path</th>
<th>Tasks</th>
<th>%CPU</th>
<th>Memory</th>
<th>Input/s</th>
<th>Output/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>/</td>
<td>453</td>
<td>20.9</td>
<td>19.3G</td>
<td>0B</td>
<td>11.8K</td>
</tr>
<tr>
<td>/machine.slice</td>
<td>-</td>
<td>2.7</td>
<td>132.1M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/machine.slice...tance\x2d00000017.scope</td>
<td>2</td>
<td>2.7</td>
<td>132.1M</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/machine.slice...0000017.scope/emulator</td>
<td>2</td>
<td>2.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/machine.slice...x2d00000017.scope/vcpu0</td>
<td>1</td>
<td>0.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/system.slice/auditd.service</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>/system.slice/avahi-daemon.service</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

May need to enable accounting – perfect drop-
```

```
[root@rhel7 ~]# vi /etc/systemd/system/mariadb.service.d/accounting.conf
[Service]
CPUAccounting=1
MemoryAccounting=1
BlockAccounting=1
```
systemd-cgls

Recursively show control group contents:

```
[root@rhel7 ~]# systemd-cgls

└─1 /usr/lib/systemd/systemd --switched-root --system --deserialize 23
   └─user.slice
       └─user-1000.slice
           └─session-2.scope
               └─311 -bash
                   └─2830 sshd: mruzicka [priv
                       └─2866 sshd: mruzicka@pts/1
                           └─2867 -bash

└─system.slice
   └─systemd-localed.service
       └─1810 /usr/lib/systemd/systemd-localed
   └─colord.service
       └─1644 /usr/libexec/colord
   └─upower.service
       └─1145 /usr/libexec/upowerd
   └─polkit.service
       └─680 /usr/lib/polkit-1/polkitd --no-debug
```
systemd Logging: journalctl
Improved Logging

- Don’t need to wait for syslog to start
- No More Losing STDERR and STDOUT
- More detail than classic syslog alone
- Logging with metadata
- Improved debugging and profiling
journalctl

- Does not replace rsyslog in RHEL 7
  - rsyslog is enabled by default
- The journal is not persistent by default.
  - Enable persistence: `mkdir /var/log/journal`
- Stored in key-value pairs
  - journalctl [tab] [tab]
  - Man 7 systemd.journal-fields
- Collects event metadata along with the message
- Simple to filter
  - Interleave units, binaries, etc.
Using the Journal

- Tail the journal: `journalctl -f`
- Show X number of lines: `journalctl -n 50`
- View from boot: `journalctl -b`
- Filter by priority: `journalctl -p [level]`

<table>
<thead>
<tr>
<th></th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>emerg</td>
</tr>
<tr>
<td>1</td>
<td>alert</td>
</tr>
<tr>
<td>2</td>
<td>crit</td>
</tr>
<tr>
<td>3</td>
<td>err</td>
</tr>
<tr>
<td>4</td>
<td>warning</td>
</tr>
<tr>
<td>5</td>
<td>notice</td>
</tr>
<tr>
<td>6</td>
<td>debug</td>
</tr>
</tbody>
</table>
journalctl

View basic logs:

[root@rhel7 ~]# journalctl

- Logs begin at Tue 2015-02-17 17:56:24 EST, end at Tue 2015-02-17 22:01:01 EST
Feb 17 17:56:24 rhel7.mruzicka systemd-journald[90]: Runtime journal is using 6.2
Feb 17 17:56:24 rhel7.mruzicka systemd-journald[90]: Runtime journal is using 6.2
Feb 17 17:56:24 rhel7.mruzicka kernel: Initializing cgroup subsys cpuset
Feb 17 17:56:24 rhel7.mruzicka kernel: Initializing cgroup subsys cpu
Feb 17 17:56:24 rhel7.mruzicka kernel: Initializing cgroup subsys cpuacct
Feb 17 17:56:24 rhel7.mruzicka kernel: Linux version 3.10.0-229.el7.x86_64 (mock
Feb 17 17:56:24 rhel7.mruzicka kernel: Command line: BOOT_IMAGE=/vmlinuz-3.10.0-

- Time stamps converted to system local time zone
- All logged data is shown, including rotated logs
- Non-persistent by default, can be preserved
journalctl

View most recent logs: (\texttt{-f} to follow)

\begin{verbatim}
[root@rhel7 ~]# journalctl -n 10
-- Logs begin at Tue 2015-02-17 17:56:24 EST, end at Tue 2015-02-17 22:05:37 EST
Feb 17 22:00:21 rhel7.mruzicka dbus[623]: [system] Successfully activated service
Feb 17 22:01:01 rhel7.mruzicka systemd[1]: Created slice user-0.slice.
Feb 17 22:01:01 rhel7.mruzicka systemd[1]: Starting Session 37 of user root.
Feb 17 22:01:01 rhel7.mruzicka systemd[1]: Started Session 37 of user root.
Feb 17 22:01:01 rhel7.mruzicka CROND[24501]: (root) CMD (run-parts /etc/cron.hourly)
Feb 17 22:01:01 rhel7.mruzicka run-parts(/etc/cron.hourly)[24507]: starting Oana
Feb 17 22:01:01 rhel7.mruzicka run-parts(/etc/cron.hourly)[24513]: finished Oana
Feb 17 22:01:01 rhel7.mruzicka run-parts(/etc/cron.hourly)[24515]: starting Yum
Feb 17 22:01:01 rhel7.mruzicka run-parts(/etc/cron.hourly)[24519]: finished Yum
Feb 17 22:05:37 rhel7.mruzicka [24590]: blah blah blah
\end{verbatim}

• Can force stdout/stderr to write to journal with \texttt{systemd-cat} if wanted

\begin{verbatim}
[root@rhel7 ~]# systemctl-cmd echo 'blah blah blah'
\end{verbatim}
Filter by priority:

```
[root@rhel7 ~]# journalctl -p err
-- Logs begin at Tue 2015-02-17 17:56:24 EST, end at Tue 2015-02-17 22:10:01 EST
Feb 17 17:56:24 rhel7.mruzicka kernel: Failed to access perfctr msr (MSR cl is 0
Feb 17 17:56:25 rhel7.mruzicka systemd[1]: /usr/lib/systemd/system-generators/a
```

Filter by time and priority:

```
[root@rhel7 ~]# journalctl -p err --since "2015-2-17 18:00:00" --until "2015-2-1
7 18:10:00"
-- Logs begin at Tue 2015-02-17 17:56:24 EST, end at Tue 2015-02-17 22:10:01 EST
Feb 17 18:00:57 rhel7.mruzicka systemd[3163]: Failed at step EXEC spawning /usr/
Feb 17 18:00:57 rhel7.mruzicka systemd[1]: Failed to start Hello Service.
```

- Advanced filtering by field, UID, unit, etc.
Using journalctl

- Other useful filters:
  - -r reverse order
  - -u [unit]
  - binary e.g. /usr/sbin/dnsmasq [additional binaries]
  - --since=yesterday or YYYY-MM-DD (HH:MM:SS)
  - --until=YYYY-MM-DD

- View entire journal
  - journalctl -o verbose (useful for grep)
Systemd Journal

How to enable persistent logging for the systemd journal

- https://access.redhat.com/solutions/696893

System Administrator's Guide


Lennart Poettering - The systemd Journal

- https://www.youtube.com/watch?v=i4CACB7paLc
systemd - Review
Review: systemd

- Replaces init and does much more
- It is here and it’s powerful
- New boot and root password reset process
- New commands and functionality
- Plenty of great information and resources available
Start using the new commands

Bash Completion is your friend!
  - # yum install bash-completion

systemd Cheat Sheet for Red Hat Enterprise Linux 7
  • https://access.redhat.com/articles/systemd-cheat-sheet

Common Administrative Commands in RHEL 5, 6, & 7
  • https://access.redhat.com/articles/1189123
Compatibility

- Systemd maintains 99% backwards compatibility with LSB compatible initscripts and the exceptions are well documented.
- While we do encourage everyone to convert legacy scripts to service unit files, it's not a requirement.
- Incompatibilities are listed here:
  http://www.freedesktop.org/wiki/Software/systemd/Incompatibilities/
- Converting SysV Init Scripts:
  http://0pointer.de/blog/projects/systemd-for-admins-3.html
Lots of great info at https://access.redhat.com and http://www.freedesktop.org/wiki/Software/systemd/

Overview of systemd for RHEL 7
• https://access.redhat.com/articles/754933

Red Hat Summit 2013 - Getting Ready for systemd
• https://access.redhat.com/videos/403833

RHEL 7 - systemd Service & Resource Management
• https://access.redhat.com/videos/898503

Rethinking PID 1 by Lennart Poettering
• http://0pointer.de/blog/projects/systemd.html
More Resources

- RHEL 7 documentation: https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/
- Systemd project page: http://www.freedesktop.org/wiki/Software/systemd/
- Lennart Poettering's systemd blog entries: (read them all) http://0pointer.de/blog/projects/systemd-for-admins-1.html
- Red Hat System Administration II & III (RH134/RH254) http://redhat.com/training/
- Systemd FAQ
- Tips & Tricks
Systemd Resources

- RHEL 7 documentation: https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/
- Systemd project page: http://www.freedesktop.org/wiki/Software/systemd/
- Lennart Poettering's systemd blog entries: (read them all) http://0pointer.de/blog/projects/systemd-for-admins-1.html
- Red Hat System Administration II & III (RH134/RH254) http://redhat.com/training/
- Systemd FAQ
- Tips & Tricks