Systemd – Easy as 1, 2, 3

Ben Breard, RHCA
Solutions Architect, Red Hat
bbreard@redhat.com

Agenda

- Systemd functionality
- Coming to terms
- Learning the basics
- More advanced topics
- Learning the journal
- Available resources

Systemd is more than a SysVinit replacement

Systemd is a system and service manager

Systemd Overview

- Controls “units” rather than just daemons
- Handles dependency between units.
- Tracks processes with service information
  - Services are owned by a cgroup.
  - Simple to configure “SLAs” based on CPU, Memory, and IO.
- Properly kill daemons
- Minimal boot times
- Debuggability – no early boot messages are lost
- Easy to learn and backwards compatible.

Closer look at Units

Systemd - Units

- Naming convention is: name.type
  - httpd.service, sshd.socket, or dev-hugepages.mount
- **Service** – Describe a daemon's type, execution, environment, and how it's monitored.
- **Socket** – Endpoint for interprocess communication. File, network, or Unix sockets.
- **Target** – Logical grouping of units. Replacement for runlevels.
- **Device** – Automatically created by the kernel. Can be provided to services as dependents.
- **Mounts**, **automounts**, **swap** – Monitor the mounting/unmounting of file systems.

Systemd – Units Continued

- **Snapshots** – save the state of units – useful for testing
- **Timers** – Timer-based activation
- **Paths** – Uses inotify to monitor a path
- **Slices** – For resource management.
  - system.slice – services started by systemd
  - user.slice – user processes
  - machine.slice – VMs or containers registered with systemd

Systemd – Dependency Resolution

- Example:
  - Wait for block device
  - Check file system for device
  - Mount file system

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

That's all great ....but

Replace Init scripts!?
Are you crazy?!

We're not crazy, I promise

- SysVinit had a good run, but leaves a lot to be desired.
- Often we work around init more than we realize.
  - One-node clusters
  - Daemon Monitoring with utilities such as monit
  - rc.local hacks
  - Tweaking symlinks under /etc/rc.d/rcX.d/S* to effect execution order
- Systemd encourages better standardization across distributions
  - LSB helped in this effort, but.....
  - Distribution standards benefit us all.

Fine, but isn't this just change for change's sake?

Not Really

- Systemd enables much “smarter” and easier to manage systems.
- PID 1 now handles dependency resolution.
  - No more adding things like `sleep 60; service [daemon] restart` to rc.local
- Services can be configured to autospawn and respawn
- Cgroup integration makes cgroups much easier to leverage.
- Most of us like Init because it's familiar and well understood.
- Systemd is simple to learn, and is easier for noobs

...but I just got used to Upstart in RHEL6.

...well, remember [deprecated technology]

- One of the best things about open source is that the *best* technology wins.
- Albeit, it can be frustrating to keep up, but **comfort should not hinder innovation**
- Upstart was a huge step forward from SysVinit, and was a great addition in RHEL 6.
- Upstart added the ability to respawn services and enabled some parallelization at boot.
- The downside is it failed to handle dependencies, and left it to the user/maintainer.
- Systemd solves that problem and many others.

....but I love System-V init scripts!!!

You're in luck!

- systemd maintains 99% backwards compatibility with 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.
  - ***hint: we'll show you how to do this in a few minutes.
- 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

Isn't systemd just about fast boot times?
I don't care about that on my servers!

You sure about that?

- Lennart Poettering says that “Fast booting isn't the goal of systemd, it's a result of a well designed system.”
- As virt/cloud demand continues, the desire for light-weight, reliable/resilient, and *fast* images grows.
  - A striped down image can boot in ~2 seconds.
  - Less CPU cycles burned during the boot process
  - Important for highly dense and dynamic environments.
  - Even more important for containers.

I don't like change.
It makes me
uncomfortable.

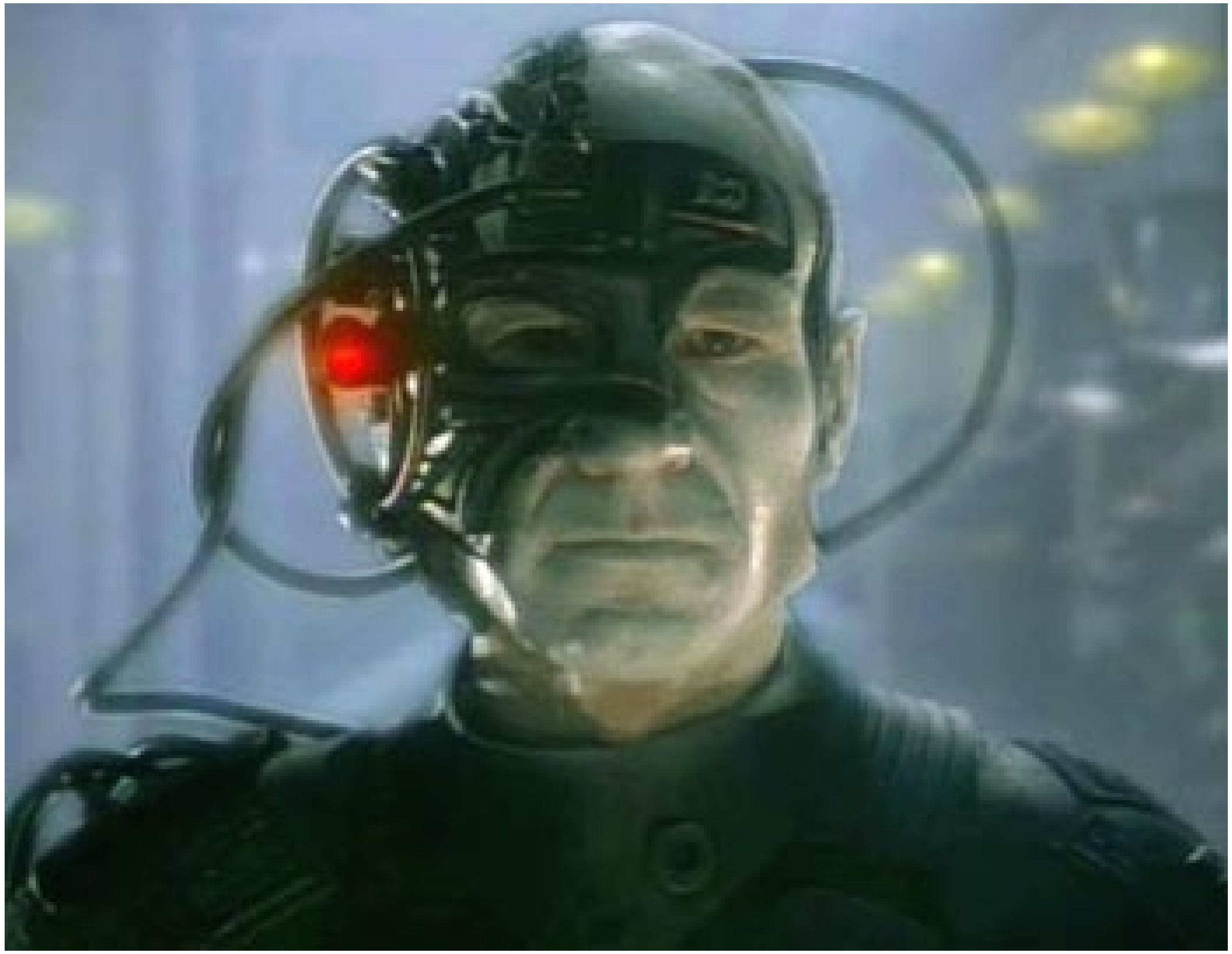
-Alf (R.I.P.)

Dude, seriously!?

Change is constant. Embrace rather than resist.

...in other words.

Resistance is futile!



Captain Jean Luc Picard as Locutus

The Basics:
Managing Services

Managing Services – Unit files

Via Init:

Init scripts are stored in /etc/init.d & called from /etc/rc*

Via systemd:

Maintainer unit files: /usr/lib/systemd/system

User unit files: /etc/systemd/system

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

Managing Services – Start/Stop

Via Init:

```
$ service httpd {start,stop,restart,reload}
```

Via systemctl:

```
$ systemctl {start,stop,restart,reload} httpd.service
```

Managing Services – Start/Stop

Note that:

- systemctl places the “action” before the service name.
- If a unit isn't specified, .service is assumed.
    - systemctl start httpd == systemctl start httpd.service
- Tab completion works great with systemctl
    - Install bash-completion

Managing Services – Status

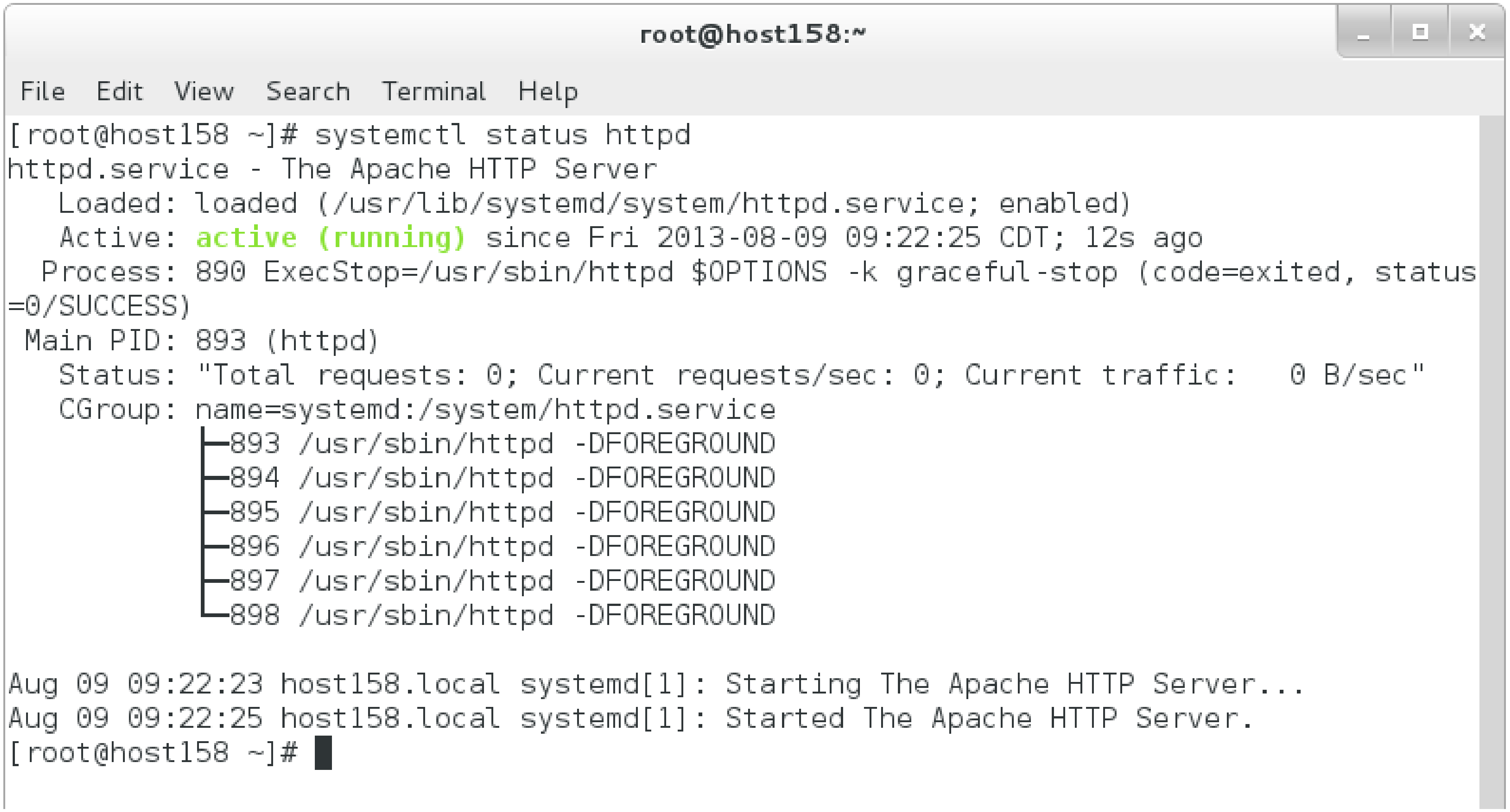
Via Init:

```
$ service httpd status
```

Via systemctl:

```
$ systemctl status httpd.service
```

Managing Services – Status

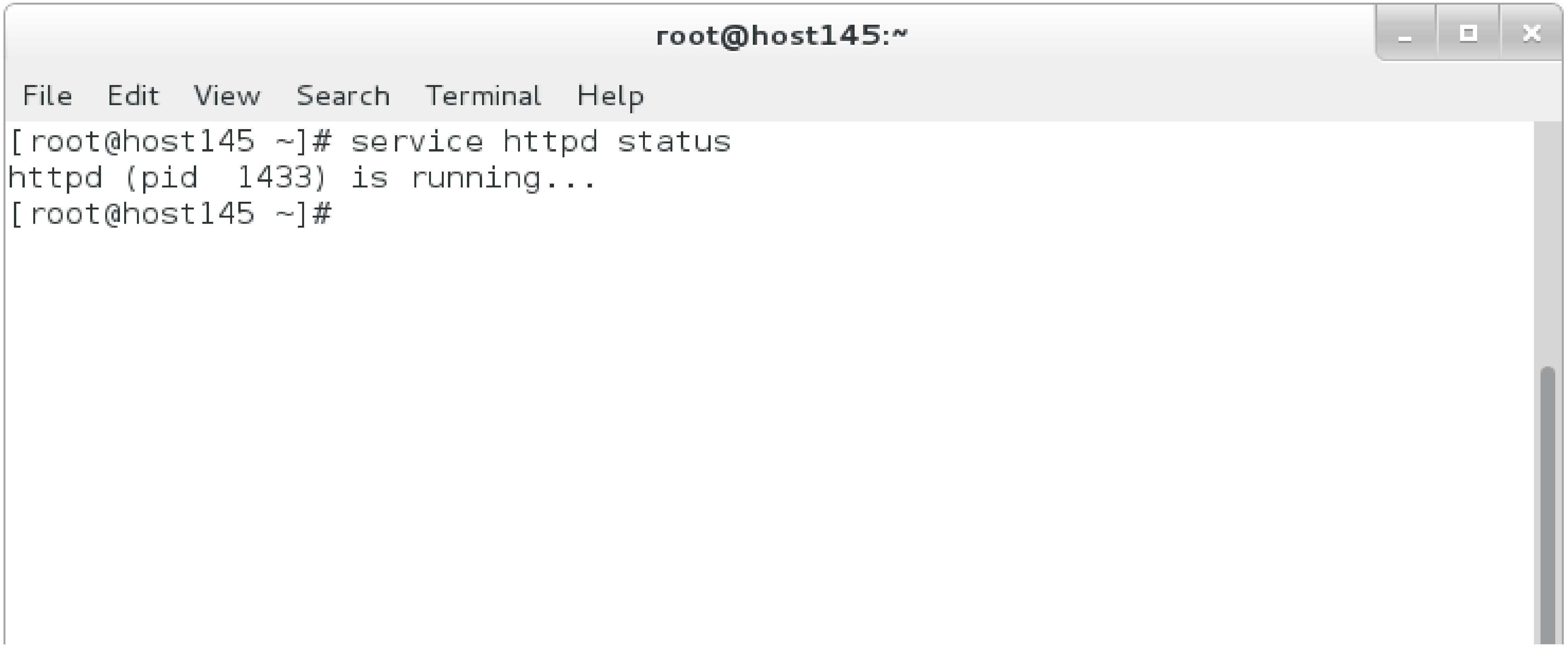


```
[root@host158 ~]# systemctl status httpd
httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
   Active: active (running) since Fri 2013-08-09 09:22:25 CDT; 12s ago
  Process: 890 ExecStop=/usr/sbin/httpd $OPTIONS -k graceful-stop (code=exited, status
=0/SUCCESS)
 Main PID: 893 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 B/sec"
   CGroup: name=systemd:/system/httpd.service
           ├─893 /usr/sbin/httpd -DFOREGROUND
           ├─894 /usr/sbin/httpd -DFOREGROUND
           ├─895 /usr/sbin/httpd -DFOREGROUND
           ├─896 /usr/sbin/httpd -DFOREGROUND
           ├─897 /usr/sbin/httpd -DFOREGROUND
           └─898 /usr/sbin/httpd -DFOREGROUND

Aug 09 09:22:23 host158.local systemd[1]: Starting The Apache HTTP Server...
Aug 09 09:22:25 host158.local systemd[1]: Started The Apache HTTP Server.
[root@host158 ~]# 
```

Managing Services – Status

- That's a little more helpful than:



```
[root@host145 ~]# service httpd status
httpd (pid  1433) is running...
[root@host145 ~]#
```

Managing Services – Status

- List running services:
  - systemctl -t service (similar to chkconfig --list)
- View cgroup tree:
  - Systemd-cgls

- *tip* systemctl can connect to remote hosts over SSH using “-H”

Managing Services – Enable/Disable

Via Init:

```
$ chkconfig httpd {on,off}
```

Via systemctl:

```
$ systemctl {enable,disable,mask} httpd.service
```

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.”

Runlevels

~~Runlevels~~

Runlevels == Targets

- "Runlevels" are exposed via target units
- /etc/inittab is no longer used
- Target names are more relevant:
  - multi-user.target vs. runlevel3
  - graphical.target vs. runlevel5
- Set the default via: `systemctl enable graphical.target --force`
- Change at run-time via: `systemctl isolate [target]`
- Change at boot time by appending:
  - systemd.unit=[target]
    - Rescue append '1', 's', or systemd.unit=rescue.target
    - Emergency append emergency, or systemd.unit=emergency.target

Runlevel Names

Runlevel	Systemd Target	Description
0	poweroff.target, runlevel0.target	System halt
1	rescue.target, runlevel1.target	Single user mode
3 (2,4)	multi-user.target, runlevel3.target	Multi-user, non graphical
5	graphical.target, runlevel5.target	Multi-user, graphical
6	reboot.target, runlevel6.target	System reboot

```
ls /lib/systemd/system/runlevel*target -l
lrwxrwxrwx. 1 root root 15 Jul  3 21:37 /lib/systemd/system/runlevel0.target -> poweroff.target
lrwxrwxrwx. 1 root root 13 Jul  3 21:37 /lib/systemd/system/runlevel1.target -> rescue.target
lrwxrwxrwx. 1 root root 17 Jul  3 21:37 /lib/systemd/system/runlevel2.target -> multi-user.target
lrwxrwxrwx. 1 root root 17 Jul  3 21:37 /lib/systemd/system/runlevel3.target -> multi-user.target
lrwxrwxrwx. 1 root root 17 Jul  3 21:37 /lib/systemd/system/runlevel4.target -> multi-user.target
lrwxrwxrwx. 1 root root 16 Jul  3 21:37 /lib/systemd/system/runlevel5.target -> graphical.target
lrwxrwxrwx. 1 root root 13 Jul  3 21:37 /lib/systemd/system/runlevel6.target -> reboot.target
```

getty

getty

- Append: console=ttyS0
  - Will enable first detected serial port
- Simply start additional getty's via:
  - systemctl start serial-getty@USB0.service
  - Started using template file: /usr/lib/systemd/system/serial-getty@.service
- To customize serial device configuration:
  - cp /usr/lib/systemd/system/serial-getty@.service /etc/systemd/system/serial-getty@ttyS2.service
  - Edit config
  - systemctl enable serial-getty@ttyS2.service
  - systemctl start serial-getty@ttyS2.service

http://0pointer.de/blog/projects/serial-console.html

Troubleshooting the Boot Process

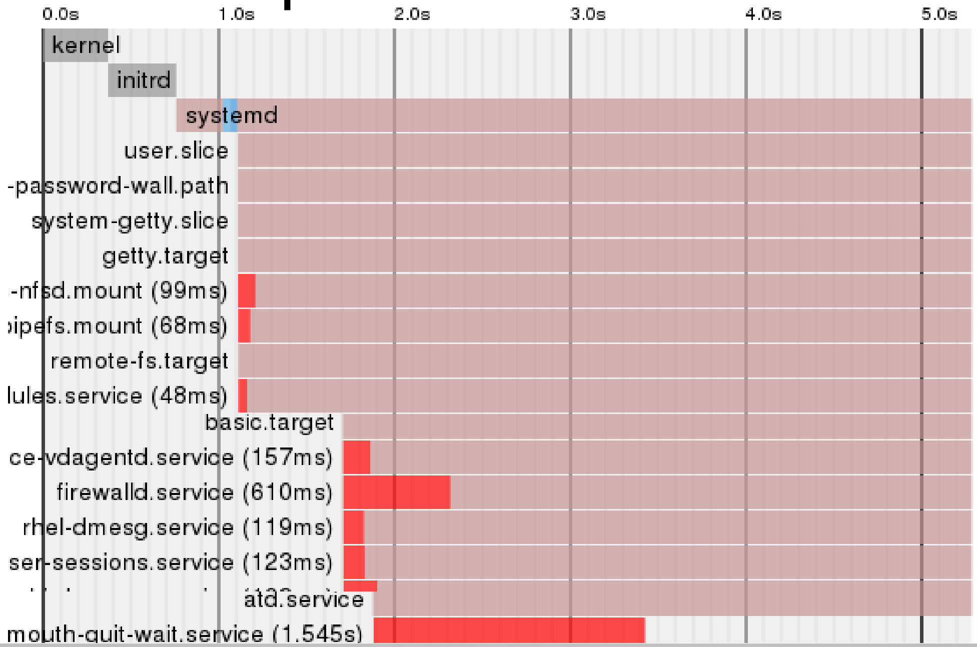
Booting

- Boot process is too fast to watch
- Interactive boot append: systemd.confirm_spawn=1
- /var/log/boot.log – still works the same
- Enable debugging from grub by appending:
  - systemd.log_level=debug systemd.log_target=kmsg log_buf_len=1M
  - Or send dbug info to a serial console:
    systemd.log_level=debug systemd.log_target=console console=ttyS0
- Enable early boot shell (can troubleshoot with systemctl command)
  - systemctl enable debug-shell.service
- systemctl list-jobs

http://freedesktop.org/wiki/Software/systemd/Debugging/

Booting

- rc.local
  - touch /etc/rc.d/rc.local ; chmod +x /etc/rc.d/rc.local
    - Don't forget to add #!/bin/bash
- systemd-analyze – stats
  - Use blame and/or plot for more details



Customizing Service Unit Files

Service Unit Files

- Changes under /usr/lib/systemd/system will be overwritten by rpms
- It is recommended to either:
  - copy unit files to /etc/systemd/system/
  - or add an include statement to new unit file.
    - .include /usr/lib/systemd/system/httpd.service
- /etc service files will take precedence over /usr
- Simply delete the modified service file to revert to defaults
- systemd-delta – will show what is overridden

Service Files – httpd Example

- First edit the new service file:
  - vim /etc/systemd/system/httpd.service

```
.include /usr/lib/systemd/system/httpd.service

[Service]
Restart=always
StartLimitInterval=10
StartLimitBurst=5
StartLimitAction=reboot
Nice=-5
WatchdogSec=1
```

- Reload services files: systemctl daemon-reload
- Restart httpd: systemctl restart httpd

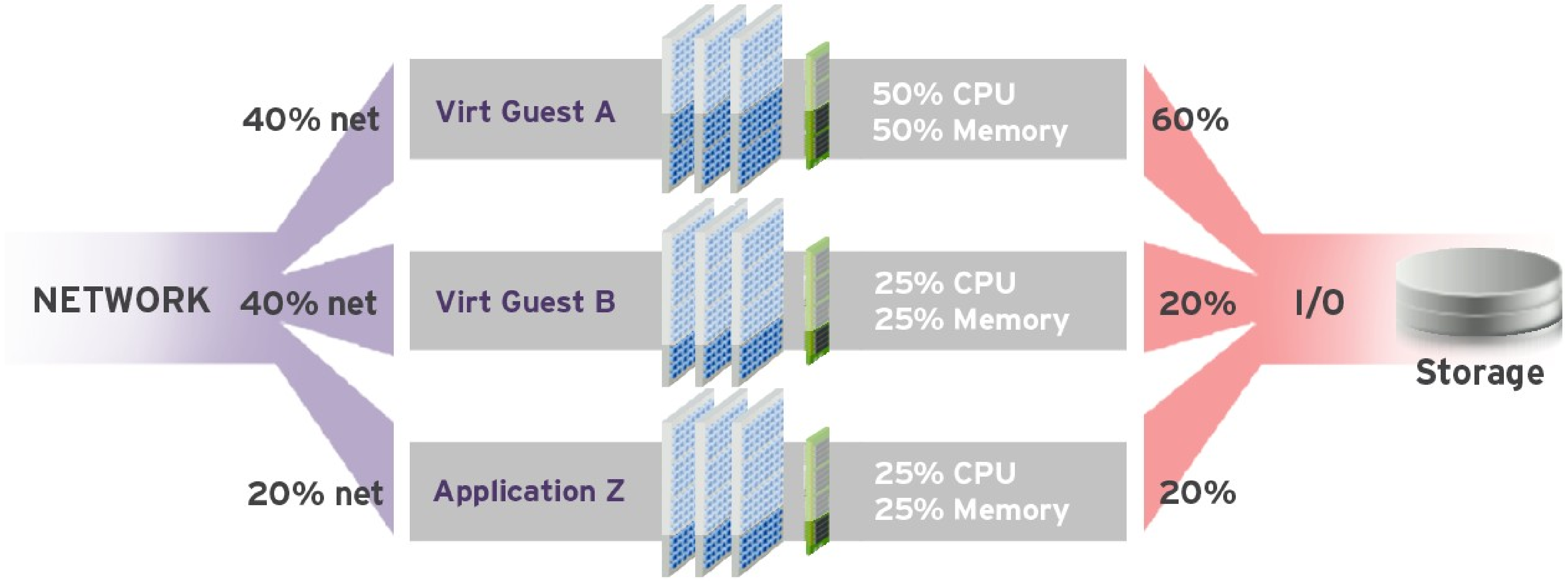
Service Files – httpd Example

- Nice, CPUAffinity, CPUSchedulingPolicy, CPUSchedulingPriority, LimitCPU, IOSchedulingPriority, OOMScoreAdjust, IOSchedulingClass, etc
- For details see:
  - man 5 systemd.service
  - man 5 systemd.exec

Resource Management

Control Groups made simple

- Resource Management with cgroups can reduce application or VM contention and improve throughput



Resource Management

- cgroups are configured in /etc/systemd/system.conf
- CPU enabled by default
- Alter DefaultControllers for additional controllers.
  - e.g. DefaultControllers=cpu,memory,blkio
- Each service is run in it's own cgroup
- Cgroup settings are per service not process
- View usage via systemd-cgtop

Resource Management - CPU

- CPUShares – default is 1024.
- Increase to assign more CPU to a service
  - e.g. CPUShares=1600

Resource Management - Memory

- Expose MemoryLimit and MemorySoftLimit
- Use K, M, G, T suffixes
  - MemoryLimit=1G

Resource Management - BlkIO

- BlockIOWeight= assigns an IO weight to a specific service
  - Similar to CPU shares
  - Default is 1000
  - Range 10 – 1000
  - Can be defined per device (or mount point)
- BlockIOReadBandwidth & BlockIOWriteBandwidth
  - BlockIOWriteBandwith=/var/log 5M

Resource Management – additional attributes

- Not all cgroup attributes are exposed in systemd.
- Additional attributes are available via: ControlGroupAttribute
  - e.g. ControlGroupAttribute=memory.swappiness 70
- Configure runtime (will not persist) via get-cgroup-attr & set-cgroup-attr
  - systemctl get-cgroup-attr httpd.service cpu.shares
  - systemctl set-cgroup-attr httpd.service cpu.shares 2048
- Remember to monitor with systemd-cgtop

http://0pointer.de/blog/projects/resources.html

Converting Init Scripts

But first, remember what init scripts look like?

/etc/init.d/httpd

```
. /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 $"Starting $prog: "
        LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && touch ${lockfile}
        return $RETVAL
}
stop() {
        echo -n $"Stopping $prog: "
        killproc -p ${pidfile} -d ${STOP_TIMEOUT} $httpd
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
}
```

From RHEL 6.4; comments removed

Init – httpd continued

```
reload() {
    echo -n $"Reloading $prog: "
    if ! LANG=$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} $httpd
        RETVAL=$?
        ;;
```

Init – httpd continued

```
 restart)
        stop
        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
```

Contrast that with a systemd unit file syntax

Unit file layout – httpd.service

```
[Unit]
Description=The Apache HTTP Server
After=network.target 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
```

*Comments were removed for readability

Unit file layout – Custom application

```
[Unit]
Description=Something generic
After=syslog.target network.target

[Service]
ExecStart=/usr/sbin/[myapp] -D
Type=forking
PIDFile=/var/run/myapp.pid

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

Unit file layout – Test your unit file

- Copy the unit file
  - cp myapp.service /etc/systemd/system/
- Alert systemd of the changes:
  - systemctl daemon-reload
- Start service
  - systemctl start myapp.service
- View status
  - systemctl status myapp.service

http://0pointer.de/blog/projects/systemd-for-admins-3.html

The Journal

Journal

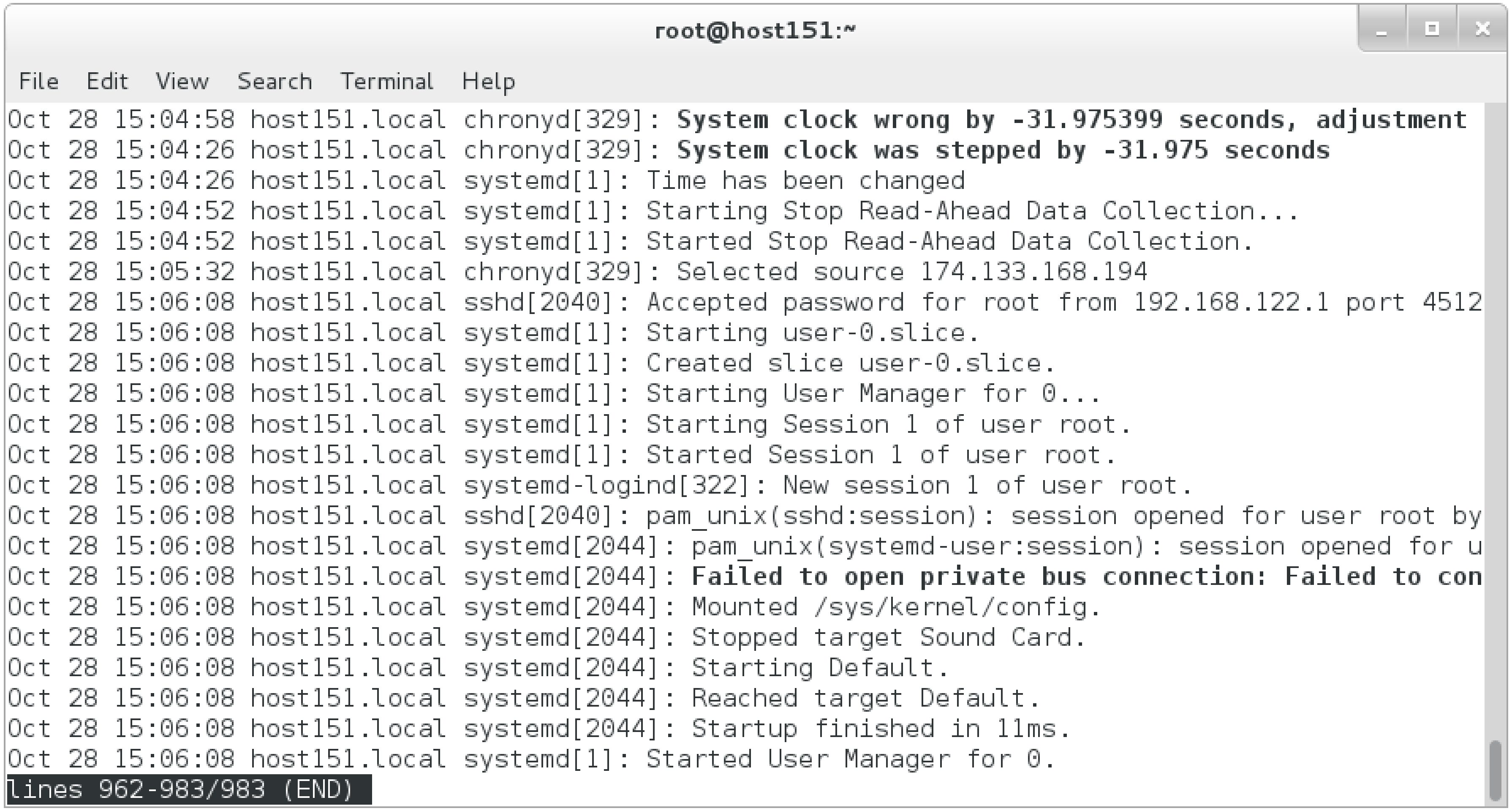
- Indexed
- Formatted
  - Errors in red
  - Warnings in bold
- Security
- Reliability
- Intelligently rotated

Journal

- Does not replace rsyslog in RHEL 7
  - rsyslog is enabled by default
- Use rsyslog for traditional logging w/ enterprise features
- The journal is not persistent by default.
- Collects event metadata
- Stored in key-value pairs
  - man page: systemd.journal-fields(7)
- journalctl - utility for to viewing the journal.
  - Simple (or complex) filtering
  - Interleave units, binaries, etc

Using the Journal

- journalctl


root@host151:~
File Edit View Search Terminal Help

```
Oct 28 15:04:58 host151.local chronyd[329]: System clock wrong by -31.975399 seconds, adjustment
Oct 28 15:04:26 host151.local chronyd[329]: System clock was stepped by -31.975 seconds
Oct 28 15:04:26 host151.local systemd[1]: Time has been changed
Oct 28 15:04:52 host151.local systemd[1]: Starting Stop Read-Ahead Data Collection...
Oct 28 15:04:52 host151.local systemd[1]: Started Stop Read-Ahead Data Collection.
Oct 28 15:05:32 host151.local chronyd[329]: Selected source 174.133.168.194
Oct 28 15:06:08 host151.local sshd[2040]: Accepted password for root from 192.168.122.1 port 4512
Oct 28 15:06:08 host151.local systemd[1]: Starting user-0.slice.
Oct 28 15:06:08 host151.local systemd[1]: Created slice user-0.slice.
Oct 28 15:06:08 host151.local systemd[1]: Starting User Manager for 0...
Oct 28 15:06:08 host151.local systemd[1]: Starting Session 1 of user root.
Oct 28 15:06:08 host151.local systemd[1]: Started Session 1 of user root.
Oct 28 15:06:08 host151.local systemd-logind[322]: New session 1 of user root.
Oct 28 15:06:08 host151.local sshd[2040]: pam_unix(sshd:session): session opened for user root by
Oct 28 15:06:08 host151.local systemd[2044]: pam_unix(systemd-user:session): session opened for u
Oct 28 15:06:08 host151.local systemd[2044]: Failed to open private bus connection: Failed to con
Oct 28 15:06:08 host151.local systemd[2044]: Mounted /sys/kernel/config.
Oct 28 15:06:08 host151.local systemd[2044]: Stopped target Sound Card.
Oct 28 15:06:08 host151.local systemd[2044]: Starting Default.
Oct 28 15:06:08 host151.local systemd[2044]: Reached target Default.
Oct 28 15:06:08 host151.local systemd[2044]: Startup finished in 11ms.
Oct 28 15:06:08 host151.local systemd[1]: Started User Manager for 0.
lines 962-983/983 (END)
```

Using the Journal

- Enable persistence: `mkdir /var/log/journal`
- View from boot: `journalctl -b`
- Tail -f and -n work as expected:
  - journalctl -f ; journalctl -n 50
- Filter by priority: `journalctl -p [level]`

0	emerg
1	alert
2	crit
3	err
4	warning
5	notice
6	debug

Using the Journal

- Other useful filters:
  - --since=yesterday or YYYY-MM-DD (HH:MM:SS)
  - --until=YYYY-MM-DD
  - -u [unit]
  - Pass binary e.g. /usr/sbin/dnsmasq
- View journal fields
  - journalctl [tab] [tab]   ←bash-completion rocks!!
- Entire journal
  - journal -o verbose (useful for grep)

Systemd Resources

- RHEL 7 documentation placeholder: 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)
- FAQ
- Tips & Tricks