Demystifying Systemd

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 SysV init 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
- Debggability – 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** – cgroup hierarchy for resource management.
- **Scopes** – Organizational units that groups services' worker processes.
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.
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 lightweight, reliable/resilient, and fast images grows.
  - A stripped 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.
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 files: /usr/lib/systemd/system/
User modifications: /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
- Multiple services can be passed in one command.
  - systemctl start httpd mariadb
- 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...
[root@host158 ~]#  ```
Managing Services – Status

- That's a little more helpful than:

```bash
[root@host145 ~]# service httpd status
httpd (pid 1433) is running...
[root@host145 ~]#
```
Managing Services – Status

- List loaded services:
  - systemctl -t service

- List installed services:
  - systemctl list-unit-files -t service (similar to chkconfig --list)

- View state:
  - systemctl --state failed

*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, unmask} 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 == 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`
- View the default target via: `systemctl get-default`
- Set the default target via: `systemctl set-default [target]`
- Change at run-time via: `systemctl isolate [target]`
- Change at boot time by appending `systemd.unit=[target]` to the kernel
  - Rescue mode: append `single, s, S, or 1`
  - Emergency (similar to `init=/bin/bash`): append `-b or emergency`
## Runlevel Names

<table>
<thead>
<tr>
<th>Runlevel</th>
<th>Systemd Target</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>poweroff.target, runlevel0.target</td>
<td>System halt</td>
</tr>
<tr>
<td>1</td>
<td>rescue.target, runlevel1.target</td>
<td>Single user mode</td>
</tr>
<tr>
<td>3 (2,4)</td>
<td>multi-user.target, runlevel3.target</td>
<td>Multi-user, non graphical</td>
</tr>
<tr>
<td>5</td>
<td>graphical.target, runlevel5.target</td>
<td>Multi-user, graphical</td>
</tr>
<tr>
<td>6</td>
<td>reboot.target, runlevel6.target</td>
<td>System reboot</td>
</tr>
</tbody>
</table>

```
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 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:
  - debug systemd.log_target=kmsg log_buf_len=1M
  - Or send debug info to a serial console:
    - debug systemd.log_target=console console=ttyS0
- Enable early boot shell on tty9
  - systemctl enable debug-shell.service
  - ln -s /usr/lib/systemd/system/debug-shell.service /etc/systemd/system/sysinit.target.wants/
- 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
  - Use 'blame', 'plot', or 'critical-chain' for more details
Customizing Service Unit Files
Customizing Service Unit Files

- Unit files can be altered or extended by placing “drop-ins” under: /etc/systemd/system/foobar.service.d/*.conf

  # cat /etc/systemd/system/httpd.service.d/50-httpd.conf

  [Service]
  Restart=always
  StartLimitInterval=10
  StartLimitBurst=5
  StartLimitAction=reboot
  CPUShares=2048
  Nice=-10
  OOMScoreAdjust=-1000

- Changes are applied on top of maintainer unit files.
Customizing Service Unit Files

- Run `systemctl daemon-reload` after making changes to notify systemd
- Drop-ins will be shown from `systemctl status`

```bash
# systemctl status httpd.service
httpd.service - The Apache HTTP Server
 Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled)
 Drop-In: /etc/systemd/system/httpd.service.d
         └─50-httpd.conf
```
Customizing Service Unit Files – Tips!

- Changes to unit files under /usr/lib/systemd/system/ could be **overwritten** by updates. **DON'T DO IT!**
- /etc service files will take precedence over /usr
- Simply delete the drop-in to revert to defaults. Don't forget to run `systemctl daemon-reload`
- systemd-delta – will show what is overridden and extended between /usr & /etc.
Customizing Service Unit Files

- 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 and predictability.
Resource Management

- View cgroup hierarchy via systemd-cgls
- View usage stats via systemd-cgtop (use for tuning)
- Default hierarchy
  - system.slice – contains system services
  - user.slice – contains user sessions
  - machine.slice – contains virtual machines and containers
- Services can be promoted to their own slice if necessary.
Resource Management – systemd-cgls

```
1 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
machine.slice
  machine-qemu\x2drhel7.scope
  17307 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name rhel7 -S -machine qemu\x2dEAP6.scope
  15290 /usr/bin/qemu-system-x86_64 -machine accel=kvm -name EAP6 -S -machine
user.slice
  user-0.slice
    user@0.service
      3289 /usr/lib/systemd/systemd --user
      3299 (sd-pam)
user-1000.slice
  session-7.scope
    13655 gdm-session-worker [pam/gdm-password]
    13665 /usr/bin/gnome-keyring-daemon --daemonize --login
    13710 gnome-session
    13718 dbus-launch --sh-syntax --exit-with-session
    13719 /bin/dbus-daemon --fork --print-pid 4 --print-address 6 --session
    13784 /usr/libexec/gvfsd
    13788 /usr/libexec/gvfsd-fuse /run/user/1000/gvfs -f -o big_writes
    13879 /usr/libexec/at-spi-bus-launcher
    13883 /bin/dbus-daemon --config-file=/etc/at-spi2/accessibility.conf --n
    13887 /usr/libexec/at-spi2-registryd --use-gnome-session
```
Resource Management – configuration

- `systemctl` can configure and persist cg.group attributes
  - `systemctl set-property httpd.service CPUShares=2048`
- Add `--runtime` to **not** persist the settings:
  - `systemctl set-property --runtime httpd.service \ CPUShares=2048`
- Alternatively settings can be placed in unit files
  - `[Service]`
  - `CPUShares=2048`

http://0pointer.de/blog/projects/resources.html
Resource Management - CPU

- CPUAccounting=1 to enable
- CPUShares – default is 1024.
- Increase to assign more CPU to a service
  - e.g. CPUShares=1600

https://www.kernel.org/doc/Documentation/scheduler/sched-design-CFS.txt
Resource Management - Memory

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

The idea behind soft limits is to allow control groups to use as much of the memory as needed, provided:

a. There is no memory contention

b. They do not exceed their hard limit

https://www.kernel.org/doc/Documentation/cgroups/memory.txt
Resource Management - BlkIO

- BlockIOAccounting=1
- BlockIOWeight= assigns an IO weight to a specific service (requires CFQ)
  - Similar to CPU shares
  - Default is 1000
  - Range 10 – 1000
  - Can be defined per device (or mount point)
- BlockIOWriteBandwidth & BlockIOWriteBandwidth
  - BlockIOWriteBandwith=/var/log 5M

https://www.kernel.org/doc/Documentation/cgroups/blkio-controller.txt
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}
RETV AL=0
STOP_TIMEOUT=${STOP_TIMEOUT-10}
start() {
  echo -n "$prog: "
  LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
  RETVAL=$?
  echo
  [ $RETV AL = 0 ] && touch ${lockfile}
  return $RETV AL
}
stop() {
  echo -n "$prog: "
  killproc -p ${pidfile} -d ${STOP_TIMEOUT} $httpd
  RETVAL=$?
  echo
  [ $RETV AL = 0 ] && rm -f ${lockfile} ${pidfile}
}

From RHEL 6.4; comments removed
Init – httpd continued

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} $httpd
        RETVAL=$?
        ;;
### Init – httpd continued

```bash
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 example

[Unit]
Description=Describe the daemon
After=syslog.target network.target

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

[Install]
WantedBy=multi-user.target
EAP Example

[Unit]
Description=JBoss Enterprise Application Platform
After=syslog.target network.target

[Service]
User=jboss-as
Environment=JBOSS_USER=jboss-as
Environment=JBOSS_HOME=/usr/local/EAP-6.1.1/jboss-eap-6.1
Environment=JBOSS_CONSOLE_LOG=/var/log/jbossas/console.log
ExecStart=/usr/local/EAP-6.1.1/jboss-eap-6.1/bin/standalone.sh
PIDFile=/var/run/jboss-as/jboss-as-standalone.pid
SyslogIdentifier=jboss-as
LimitNOFILE=102642
CPUShares=1600
Restart=always
Slice=jboss.slice

[Install]
WantedBy=multi-user.target
EAP Example

[root@host204 ~]# systemctl status jboss-as
jboss-as.service - JBoss Enterprise Application Platform
    Loaded: loaded (/etc/systemd/system/jboss-as.service; enabled)
    Active: active (running) since Fri 2014-01-10 11:31:20 CST; 45s ago
Main PID: 692 (standalone.sh)
    CGroupe: /jboss.slice/jboss-as.service
    692 /bin/sh /usr/local/EAP-6.1.1/jboss-eap-6.1/bin/standalone.sh
    1095 java -D[Standalone] -server -XX:+UseCompressedOops -Xms1303...
EAP Example
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`
The Journal
Journal

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

http://0pointer.de/blog/projects/journalctl.html
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`
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]`

<table>
<thead>
<tr>
<th>Level</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>
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
  - `journalctl -o verbose` (useful for grep)
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)
- FAQ
- Tips & Tricks
Questions?