Agenda

- Brief History
- Concepts & Basic Usage
- Modifying Units
- Resource Management
- Converting init scripts
- The Journal
- nspawn
- Sneak peek at what's coming in RHEL 7.2
Keep an open mind ... and please refrain from cussing or throwing objects at the speaker.
Brief History
init, Upstart and systemd
Brief History of the Init Process

- **init**
  - Referenced `inittab → rc scripts`
  - Unexpected pauses could cause delays in boot process (serial)
  - Needed to be sequenced for dependencies (manual process)

- **Upstart**
  - Introduced as init replacement
  - Added async service startup, auto restart, event-based start
  - Also referenced `inittab → /etc/init → /etc/rc.d`
  - Used `initctl` for service control

- **systemd**
  - Supersedes its predecessors in terms of speed and capabilities
Life Beyond Init
Concepts & Basic Usage
systemd

- Default init system for most Linux distributions
- 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” for CPU, Memory, and IO
- Properly kill daemons
- Minimal boot times
- Debuggability – no early boot messages are lost
- Easy to learn and backwards compatible
systemd Structure

**systemd Utilities**
- systemctl
- journalctl
- notify
- analyze
- cgl
- cgtop
- loginctl
- nspawn

**systemd Daemons**
- systemd
- journal
- networkd
- login
- user session

**systemd Targets**
- bootmode
- basic
- multi-user
- telephony
- graphical
- user-session
- display service
- tizen service

**systemd Core**
- manager
- systemd
- unit
- service
- timer
- mount
- target
- swap
- login
- multiseat
- inhibit
- session
- pam
- namespace
- cgroup
- log
- dbus

**systemd Libraries**
- dbus-1
- libpam
- libcap
- libcryptsetup
- tcpwrapper
- libaudit
- libnotify

**Linux Kernel**
- cgroups
- autofs
- kdbus
systemd Concepts: Units

- Init scripts have been replaced with service units.
- Units are `systemd` objects used for organizing boot and maintenance tasks.
- Units have a name and type.
- Unit configs are stored in respective config files.
- Service units end with the `.service` file extension and serve a similar purpose as init scripts:
  - To view, start, stop, restart, enable, or disable system services, use the `systemctl` command.
## Available systemd Unit Types

<table>
<thead>
<tr>
<th>Unit Type</th>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Unit</td>
<td>.service</td>
<td>A system service</td>
</tr>
<tr>
<td>Target Unit</td>
<td>.target</td>
<td>A group of systemd units</td>
</tr>
<tr>
<td>Automount Unit</td>
<td>.automount</td>
<td>A filesystem automount point</td>
</tr>
<tr>
<td>Device Unit</td>
<td>.device</td>
<td>A device file recognized by the kernel</td>
</tr>
<tr>
<td>Mount Unit</td>
<td>.mount</td>
<td>A filesystem mount point</td>
</tr>
<tr>
<td>Path Unit</td>
<td>.path</td>
<td>A file or directory in a filesystem</td>
</tr>
<tr>
<td>Scope Unit</td>
<td>.scope</td>
<td>An externally created process</td>
</tr>
<tr>
<td>Slice Unit</td>
<td>.slice</td>
<td>A group of hierarchically organized units that manage system processes</td>
</tr>
<tr>
<td>Snapshot Unit</td>
<td>.snapshot</td>
<td>A saved state of the systemd manager</td>
</tr>
</tbody>
</table>
# Available systemd Unit Types

<table>
<thead>
<tr>
<th>Unit Type</th>
<th>File Extension</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Socket Unit</td>
<td>.socket</td>
<td>An inter-process communications socket</td>
</tr>
<tr>
<td>Swap Unit</td>
<td>.swap</td>
<td>A swap device or swap file</td>
</tr>
<tr>
<td>Timer Unit</td>
<td>.timer</td>
<td>A systemd timer</td>
</tr>
</tbody>
</table>
systemd Units: 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
PrivateTmp=true

[Install]
WantedBy=multi-user.target

*Comments were removed for readability*
# systemd Units: Locations

<table>
<thead>
<tr>
<th>Directory</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>/usr/lib/systemd/system/</td>
<td>Systemd units distributed with RPM installed packages.</td>
</tr>
<tr>
<td>/run/systemd/system/</td>
<td>Systemd units created at runtime. This directory takes precedence over the directory with installed service units. <em>Non-persistent.</em></td>
</tr>
<tr>
<td>/etc/systemd/system/</td>
<td>Systemd units created and managed by the <strong>system administrator</strong>. This directory takes precedence over the directory with runtime units.</td>
</tr>
</tbody>
</table>

**Note:** Unit files in /etc take precedence over /usr
Compatibility Changes

- `systemd` has only limited support for runlevels
- `systemctl` utility does not support custom commands
- `systemctl` utility does not communicate with services not started by `systemd`
- `systemd` stops only running services
- System services are unable to read from the standard input stream
Compatibility Changes (cont.)

• System services do not inherit any context (such as the HOME and PATH environment variables) from the invoking user and their session
• When loading a SysV init script, `systemd` reads dependency information encoded in the Linux Standard Base (LSB) header and interprets it at run time
• All operations on service units are subject to a timeout of 5 minutes to prevent a malfunctioning service from freezing the system
systemd startup flow

- Initiates services concurrently

- Instantaneously creates sockets for enabled services

- Passes to daemon processes to start in parallel

- Maintains sockets and uses them to reconnect services

- Allows systemd to handle service order dependencies; services start without delay

- systemd creates all sockets first, daemons next

- Daemons need not be running, they only need the correct socket to be available

- Daemons not yet running are cached in socket buffer and filled when daemons come online
Managing Services: Start/Stop

Init

service httpd \{start,stop,restart,reload\}

systemd

systemctl httpd.service \{start,stop,restart,reload\}
Managing Services: Start/Stop

• Glob units to work with multiple services
  systemctl restart httpd mariadb

• When the unit “type” isn't specified, .service is assumed.
  systemctl start httpd = systemctl start httpd.service

• Make life easy and install shell completion
  – yum install bash-completion
  – systemctl [tab] [tab]
  – Add bash-completion to your SOE and minimal kickstarts

• Connect directly to remote hosts
  systemctl -H [hostname] restart httpd
Managing Services: Status

Init

- service httpd status

systemd

- systemctl status httpd

**Tip:** pass -l if the logs are cutoff
Managing Services: Status

```
[root@fedora-22 ~]# systemctl status httpd
● httpd.service - The Apache HTTP Server
    Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor preset: disabled)
    Active: active (running) since Wed 2015-08-26 17:07:12 EDT; 11s ago
    Main PID: 7014 (httpd)
    Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec: 0 B/sec"
    CGroup: /system.slice/httpd.service
        └─7014 /usr/sbin/httpd -DFOREGROUND
    Aug 26 17:07:12 fedora-22.example.com systemd[1]: Starting The Apache HTTP Server...
    Aug 26 17:07:12 fedora-22.example.com systemd[1]: Started The Apache HTTP Server.
[root@fedora-22 ~]# ```
Managing Services: Status

• List loaded services:
  – systemctl -t service

• List installed services:
  – systemctl list-unit-files -t service (like chkconfig --list)

• Check for services in failed state:
  – systemctl --state failed
Managing Services: Enable/Disable

Init

- `chkconfig httpd {on,off}

systemd

- `systemctl {enable, disable} httpd`

Tip: Globing units will clean up your kickstarts

- `systemctl enable httpd mariadb ntpd lm_sensors [etc]`
Targets == Runlevels

- “Runlevels” are exposed as target units
- More meaningful names:
  - multi-user.target vs. runlevel3
  - graphical.target vs. runlevel5
- View the default target: `systemctl get-default`
- Set the default target: `systemctl set-default [target]`
- Change at run-time: `systemctl isolate [target]`

Note: `/etc/inittab` is no longer used.
Cockpit is now in RHEL Extras

![Cockpit dashboard](image.png)
systemd Sockets

- Have a .socket extension
- Represent inter-process communication (IPC) sockets
- Used to delay start of a service at boot time and to start less frequently used apps on demand
- Similar in principle to services which use xinetd
Sockets

tftp.socket
[Unit]
Description=Tftp Server
Activation Socket

[Socket]
ListenDatagram=69

[Install]
WantedBy=sockets.target

man systemd.socket

tftp.service
[Unit]
Description=Tftp Server

[Service]
ExecStart=/usr/sbin/initftpd -s /var/lib/tftpboot
StandardInput=socket
Sockets

cockpit.socket

[Unit]
Description=Cockpit Web Server Socket
Documentation=man:cockpit-ws(8)

[Socket]
ListenStream=9090

[Install]
WantedBy=sockets.target

cockpit.service

[Unit]
Description=Cockpit Web Server
Documentation=man:cockpit-ws(8)

[Service]
ExecStartPre=/usr/sbin/remotectl cert --ensure --user=root --group=cockpit-ws
ExecStart=/usr/libexec/cockpit-ws
PermissionsStartOnly=true
User=cockpit-ws
Group=cockpit-ws
Timers

fstrim.timer
[Unit]
Description=Discard unused blocks once a week

[Timer]
OnStartupSec=10min
OnCalendar=weekly
AccuracySec=1h
Persistent=true
[Install]
WantedBy=multi-user.target

fstrim.service
[Unit]
Description=Discard unused blocks

[Service]
Type=oneshot
ExecStart=/usr/sbin/fstrim /

man systemd.timer
Customizing Units
What's Available?

- List a unit's properties:
  - `systemctl show --all httpd`

- Query a single property:
  - `systemctl show -p Restart httpd`
  - `Restart=no`

- Helpful man files: `systemd.exec` and `systemd.service`
  - Restart, Nice, CPUAffinity, OOMScoreAdjust, LimitNOFILE, etc

Disclaimer: just because you can configure something doesn't mean you should!
Customizing Units: Drop-ins

1) Create directory

```bash
mkdir /etc/systemd/system/[name.type.d]/
```

2) Create drop-in

```bash
vi /etc/systemd/system/httpd.service.d/50-httpd.conf
```

```ini
[Service]  
   Restart=always
   CPUAffinity=0 1 2 3
   OOMScoreAdjust=-1000
```

Remember the 'S' is capitalized

3) Notify systemd of the changes

```bash
systemctl daemon-reload
```
Customizing Units: Drop-ins

```
[root@host243 httpd.service.d]# systemctl status httpd
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
   Active: active (running) since Sun 2014-03-16 14:31:08 CDT; 2min 6s ago
   Process: 686 ExecStop=/bin/kill -WINCH ${MAINPID} (code=exited, status=0/SUCCESS)
   Main PID: 689 (httpd)
   Status: "Total requests: 15884; Current requests/sec: 133; Current traffic: 60KB/sec"
   CGroup: /system.slice/httpd.service
   └─689 /usr/sbin/httpd -DFOREGROUND

Mar 16 14:31:08 host243.local systemd[1]: Started The Apache HTTP Server.
```
Customizing Units: Drop-ins

- Safe to apply on running services
  - Note: some options require a service restart to take effect
- Use `systemd-delta` to see what's been altered on a system
- Simple to use with configuration tools like Satellite, Puppet, etc.
- Simply delete the drop-in to revert to defaults.
- Don't forget `systemctl daemon-reload` when modifying units
Resource Management
Slices, scopes, services
Control Groups Made Simple

Resource Management with cgroups can reduce contention and improve throughput and predictability.
Slices, Scopes, Services

- **Slice** – Unit type for creating the cgroup hierarchy for resource management.
- **Scope** – Organizational unit that groups a services' worker processes.
- **Service** – Process or group of processes controlled by systemd
Understanding the Hierarchy

- systemd implements a standard, single-root hierarchy under /sys/fs/cgroup
Understanding the Hierarchy

- Each slice gets equal CPU time on the scheduler.

user.slice  CPUShares=1024
system.slice  CPUShares=1024
machine.slice  CPUShares=1024
Understanding the Hierarchy

- `/`
  - `user.slice`
    - `user-1000.slice`
      - `session-3.scope`
        - `sshd: user`
          - `bash`
    - `user-1001.slice`
  - `system.slice`
  - `machine.slice`

CPUShares=1024
Understanding the Hierarchy

- `/`
  - `user.slice`
    - `user-1000.slice`
      - `session-3.scope`
        - `sshd: user`
          - `bash`
    - `user-1001.slice`
  - `system.slice`
    - `tomcat.service`
    - `sshd.service`
    - `mariadb.service`
    - `httpd.service`
  - `machine.slice`

CPUShares=1024
Understanding the Hierarchy

- /

user.slice

user-1000.slice

session-3.scope

sshd: user

bash

user-1001.slice

system.slice

tomcat.service

sshd.service

mariadb.service

httpd.service

machine.slice

vm1.scope

/usr/bin/qemu

vm2.scope

/usr/bin/qemu
Viewing Resources

- Show top control groups by their resource usage:
  `systemd-cgtop`
- Recursively show control group contents:
  `systemd-cgls`
Resource Management – Configuration

• Configure cgroup attributes:
  ```bash
  systemctl set-property --runtime httpd / CPUShares=2048
  ```

• Drop “--runtime” to persist:
  ```bash
  systemctl set-property httpd CPUShares=2048
  ```

• Or place in the unit file:
  ```ini
  [Service]
  CPUShares=2048
  ```

http://0pointer.de/blog/projects/resources.html
Resource Management – CPU & MEM

- CPUAccounting=1 to enable
- CPUShares – default is 1024.
- Increase to assign more CPU to a service
  - e.g. CPUShares=1600
- MemoryAccounting=1 to enable
- MemoryLimit=
- Use K, M, G, T suffixes
  - MemoryLimit=1G

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

- `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)
- `BlockIOWeight=/var/log 5M`

https://www.kernel.org/doc/Documentation/cgroups/blkio-controller.txt
Converting Init Scripts
You can do it ... it's easy!
Remember init scripts?
/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 "$prog: "
  LANG=$HTTPD_LANG daemon --pidfile=${pidfile} $httpd $OPTIONS
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && touch ${lockfile}
  return $RETVAL
}

stop() {
  echo -n "$prog: "
  killproc -p ${pidfile} -d ${STOP_TIMEOUT} $httpd
  RETVAL=$?
  echo
  [ $RETVAL = 0 ] && rm -f ${lockfile} ${pidfile}
}

From RHEL 6.4; comments removed
/etc/init.d/httpd – continued

```bash
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) $httpd
    RETVAL=$?
    ;;
```
/etc/init.d/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
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
PrivateTmp=true

[Install]
WantedBy=multi-user.target

*Comments were removed for readability
To be clear

- 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
Unit file layout: Custom App Example

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

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

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

- Store containers under `/var/lib/container` to align with `@systemd-nspawn.service`
  - `mkdir /var/lib/container`

- Install a minimal OS with `yum`; 306 rpms ~360MB on disk:
  ```
  yum -y --releasever=7Server
  --installroot=/var/lib/container/rhel7 install systemd passwd
  yum redhat-release vim-minimal
  ```

- Point nspawn at the directory and go
  - `systemd-nspawn -D /var/lib/container/rhel7`

- To “boot with an init system”, we need to, start the container, set the root password, and configure the system if necessary, etc
  - `systemd-nspawn -D /var/lib/container/rhel7 passwd ; systemctl disable kdump postfix firewall tuned`

integration points with container managers, to allow seamless management of services across container boundaries.
systemd-nspawn

- Mini-container manager that is shipped with systemd itself
- A container manager that is as simple to use and "just works"
- An integration points with container managers, to allow seamless management of services across container boundaries
- An init system runs inside the container, and the container hence in most ways appears like an independent system of its own
RHEL 7.2 will likely rebase on systemd 219
219 Highlights

- systemctl – enhancements edit, cat, etc
- CPUQuota – “cap” CPU usage for services
- systemd-socket-proxy – add socket activation to daemons that don't support it natively
- systemd-nspawn
  - Improved networking
  - Ephemeral & template support
  - “-M [container]” option for systemctl, journalctl, etc
  - Import and run Docker containers & raw cloud images
- networkd – DHCP srv/clt, bridge, bond, vlan, vxlan, macvlan, tun
Additional 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
Ohh my pretty little systemd, I love you!
THANK YOU!

Scott Seighman
Solutions Architect
Red Hat

sseighma@redhat.com
CleRHUG

plus.google.com/+RedHat
facebook.com/redhatinc
linkedin.com/company/red-hat
twitter.com/RedHatNews
youtube.com/user/RedHatVideos
Tips & Troubleshooting

• 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

• Interactive boot append: systemd.confirm_spawn=1

• Enable debugging append:
  - debug
  - debug systemd.log_target=kmsg log_buf_len=1M
  - debug systemd.log_target=console console=ttyS0

http://freedesktop.org/wiki/Software/systemd/Debugging/
Tips & Troubleshooting

- `rc.local` is supported, but no longer runs last
  - `chmod +x /etc/rc.d/rc.local`
- `systemd-analyze`
  - Use 'blame', 'plot', or 'critical-chain' for more details
- Append `systemd.unit=[target]` to the kernel
  - Rescue mode: single, s, S, or 1
  - Emergency (similar to `init=/bin/bash`): -b or emergency