DEMYSTIFYING SYSTEMD

Strengthening service concepts and management in Red Hat Enterprise Linux 8

Ben Breard, RHCA
Principal product manager

Herr Lennart Poettering
Sr. consulting engineer

AGENDA

- Concepts and unit files
- Security and sandboxing
- Resource management
- Unprivileged units
- Miscellaneous awesome stuff

PROJECT STATS

39,776

1,129

181

20

PROJECT STATS

39,776	Commits
1,129	Contributors
181	systemd releases
20	Releases since RHEL 7

PROJECT STATS

39,776	Commits
1,129	Contributors
181	systemd releases
20	Releases since RHEL 7
10	Years of systemd

(Nov 18th 2009)

A BIG YEAR FOR RED HAT

redhat®

Red Hat

A BIG YEAR FOR SYSTEMD

systemd

A BIG YEAR FOR SYSTEMD

?

A BIG YEAR FOR SYSTEMD

?

A BIG YEAR FOR SYSTEMD

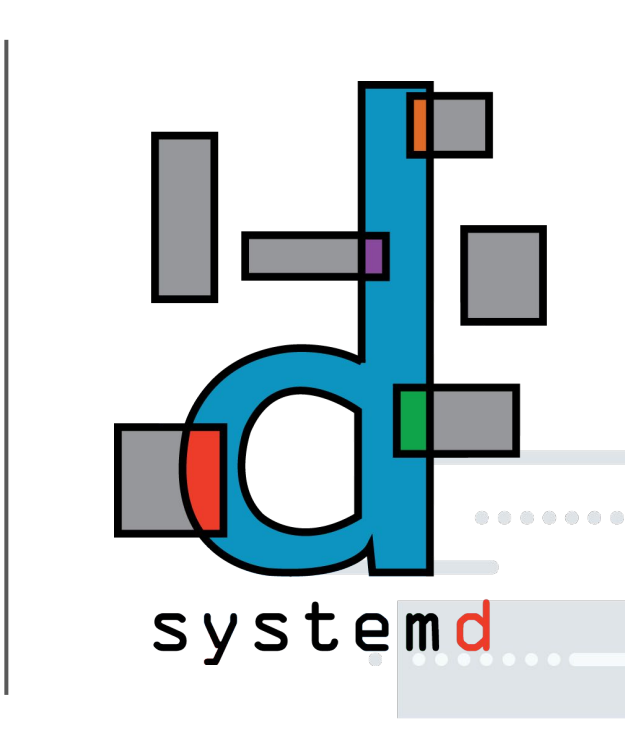
Here's what an AI/ML Logo Service

systemd

POLL: VOTE FOR YOUR FAVORITE

2nd

1st



SYSTEMD HIGHLIGHTS

Red Hat Enterprise Linux 8

Security

- Improved sandboxing and isolation options for services
- Unprivileged unit files (systemd --user)
- Additional hardening of systemd services
- Dynamic users

Usability

- Many improvements to systemctl, journalctl, etc.
- Additional service & unit files settings
- Resource management using cgroups v2 (tech preview)
- Better journal compression and performance

New technology Previews

- IP accounting and filtering
- Portable system services

CONCEPTS AND UNIT FILES

UNIT FILES

```
[Unit]
Description=The Apache HTTP Server
Wants=httpd-init.service
After=network.target remote-fs.target nss-lookup.target
httpd-init.service
Documentation=man:httpd.service(8)

[Service]
Type=notify
Environment=LANG=C

ExecStart=/usr/sbin/httpd $OPTIONS -DFOREGROUND
ExecReload=/usr/sbin/httpd $OPTIONS -k graceful
KillSignal=SIGWINCH
KillMode=mixed
PrivateTmp=true

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

UNIT TYPES

- foo.**service**
- bar.**socket**
- baz.**device**
- qux.**mount**
- waldo.**automount**
- thud.**swap**
- grunt.**target**
- snork.**timer**
- grault.**path**
- pizza.**slice**
- tele.**scope**

UNIT FILE LOCATIONS

Maintainer:

```
/usr/lib/systemd/system
```

Administrator:

```
/etc/systemd/system
```

Non-persistent, runtime:

```
/run/systemd/system
```

Identify and compare overriding unit files:

```
systemd-delta
```

Note:

unit files in `/etc` take precedence over `/run`, and `/run` over `/usr`

BASIC USAGE

`systemctl` - primary command for interacting with systemd

- e.g. start, stop, reload, restart, enable, disable, status
- `systemctl enable --now httpd`
- `systemctl set-property --runtime CPUShares=2048 httpd`

`journalctl` - view and filter the system journal

- `journalctl -fu chronyd`

`systemd-run` - use transient unit files

`systemd-analyze` - analyze and debugging systemd

`systemd-cgls` - view cgroup hierarchy

`systemd-cgtop` - view cgroup accounting

BASIC USAGE

Limit the CPU usage of a task to 15% of 1 core

```
systemd-run -p CPUQuota=15% /usr/bin/cpuhog
```

Wait for the task to complete and provide stats and exit code

```
systemd-run -p CPUAccounting=1 --wait /usr/bin/long-job
    Running as unit: run-u1573.service
    Finished with result: success
    Main processes terminated with: code=exited/status=0
    Service runtime: 30.004s
    CPU time consumed: 2ms
```

Schedule a timer

```
systemd-run --on-calendar=18:55 /usr/bin/dinner-is-ready
```

Start a shell under an automatically picked, unused UID w/ read-only fs access

```
systemd-run -p DynamicUser=1 -t /bin/bash
```

SECURITY SANDBOXING AND CAPABILITIES

SECURING UNITS

- Reduce system attack surface per unit
  - Namespace isolation
  - Syscall filters
  - Linux capabilities (breakdown of root perms)
- Provides container-style isolation for traditional services
- Simple to apply as another layer of security for systems

SECURING UNITS

`PrivateTmp=`

- File system namespace: `/tmp & /var/tmp`
- Files under: `/tmp/systemd-private-*-[unit]-*/tmp`

`PrivateNetwork=`

- Creates a network namespace with a single loopback device, private 127.0.0.1

`JoinsNamespaceOf=`

- Enables multiple units to share `PrivateTmp= & PrivateNetwork=`

`SELinuxContext=`

- Specify an SELinux security context for the process/service

https://www.freedesktop.org/software/systemd/man/systemd.exec.html

SECURING UNITS

`ProtectSystem=`

- If enabled, `/usr` and `/boot` directories are mounted read-only
- If “full”, `/etc` is also read-only
- **New:** **strict** - whole system tree is read-only except `/dev`, `/proc`, `/sys`

`ProtectHome=`

- If enabled, `/home`, `/root`, `/run/user` will appear empty
- Alternatively can set to “read-only”
- **New:** **tmpfs** - masks w/ `tmpfs` mount

`PrivateDevices=`

- If enabled, creates a private `/dev` namespace.
- Includes pseudo devices like `/dev/null`, `/dev/zero`, etc
- Disables `CAP_MKNOD`

https://www.freedesktop.org/software/systemd/man/systemd.exec.html

SECURING UNITS

`ReadWriteDirectories=, ReadOnlyDirectories=, InaccessibleDirectories=`

- Configure file system namespaces

`NoNewPrivileges=`

- Ensure a process & children cannot elevate privileges

`CapabilityBoundingSet=`

- CAP_SYS_ADMIN
- ~CAP_NET_ADMIN
- man:capabilities(7) for details

`RestrictAddressFamilies=`

- AF_INET AF_INET6 AF_UNIX
- ~AF_PACKET

https://www.freedesktop.org/software/systemd/man/systemd.exec.html

SECURING UNITS

New in Red Hat Enterprise Linux 8

`ProtectKernelTuneables=`

- Disable modification to `/proc & /sys`

`ProtectKernelModules=`

- Prohibit load/unload of modules.
- Masks `/usr/lib/modules`

`ProtectControlGroups=`

- Disable write access to `/sys/fs/cgroup`

`RestrictNamespaces=`

- Boolean to restrict all or a subset of namespaces
  - `cgroup ipc net mnt pid user uts`

`ConditionSecurity=`

- `uefi-secureboot selinux`

SECURING UNITS

New in Red Hat Enterprise Linux 8

`MemoryDenyWriteExecute=`

- Disable memory mapping that is simultaneously writable & executable

`DynamicUsers=`

- (Restrictions apply to stateful data)
- Dynamically allocated UID/GID (61184 - 65519)
- `/etc/[passwd, group]` are not altered and users are removed when the service stops

`PrivateMounts=`

- Service is run in a private mount namespace

`RestrictRealtime=`

- Prohibit real-time scheduling

`RemoveIPC=`

- Remove semaphores, shared memory, & message queues

SECURING UNITS

New in Red Hat Enterprise Linux 8

```
SystemCallFilter=
```

- seccomp filtering to whitelist/blacklist individual or groups of syscalls

@aio	@file-system	@mount	@reboot	@system-service
@basic-io	@io-event	@network-io	@resources	@timer
@chown	@ipc	@obsolete	@setuid	
@clock	@keyring	@privileged	@signal	
@cpu-emulation	@memlock	@process	@swap	
@debug	@module	@raw-io	@sync	

https://www.freedesktop.org/software/systemd/man/systemd.exec.html

SECURITY MADE SIMPLE

1. systemctl edit [unit.service]

2. Use `$EDITOR` to insert the following:

```
[Service]
ProtectSystem=strict
ProtectHome=1
PrivateDevices=1
ProtectKernelTunables=1
ProtectKernelModules=1
ProtectControlGroups=1
SystemCallFilter=@system-service
SystemCallErrorNumber=EPERM
NoNewPrivileges=1
PrivateTmp=1
```

3. :wq

4. systemctl restart [unit]

https://www.freedesktop.org/software/systemd/man/systemd.exec.html

SYSTEMD-ANALYSE SECURITY (8.1)

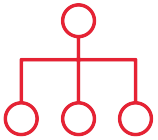
	NAME	DESCRIPTION	EXPOSURE
✗	PrivateNetwork=	Service has access to the host's network	0.5
✗	User=/DynamicUser=	Service runs as root user	0.4
✗	RestrictNamespaces=~CLONE_NEWUSER	Service may create user namespaces	0.3
✗	RestrictAddressFamilies=~…	Service has network configuration privileges	0.3
✗	CapabilityBoundingSet=~CAP_NET_ADMIN	Service may allocate exotic sockets	0.2
✓	CapabilityBoundingSet=~CAP_RAWIO	Service has no raw I/O access	
✗	CapabilityBoundingSet=~CAP_SYS_MODULE	Service may load kernel modules	0.2
✗	CapabilityBoundingSet=~CAP_SYS_TIME	Service processes may change the system clock	0.2
✓	DeviceAllow=	Service has a minimal device ACL	
✗	IPAddressDeny=	Service does not define an IP address whitelist	0.2
✓	KeyringMode=	Service doesn't share key material with other services	
✗	NoNewPrivileges=	Service processes may acquire new privileges	0.2
✓	NotifyAccess=	Service child processes cannot alter service state	
✓	PrivateDevices=	Service has no access to hardware devices	
✓	PrivateMounts=	Service cannot install system mounts	
✓	PrivateTmp=	Service has no access to other software's temporary files	0.2
✗	PrivateUsers=	Service has access to other users	0.2
✗	ProtectControlGroups=	Service may modify to the control group file system	
✓	ProtectHome=	Service has no access to home directories	0.2
✗	ProtectKernelModules=	Service may load or read kernel modules	0.2
✗	ProtectKernelTunables=	Service may alter kernel tunables	0.1
✗	ProtectSystem=	Service has very limited write access to the OS file hierarchy	
✓	SystemCallFilter=~@clock	System call whitelist defined for service, and @clock is not included	
✓	SystemCallFilter=~@debug	System call whitelist defined for service, and @debug is not included	
✓	SystemCallFilter=~@module	System call whitelist defined for service, and @module is not included	
✓	SystemCallFilter=~@mount	System call whitelist defined for service, and @mount is not included	
✓	SystemCallFilter=~@raw-io	System call whitelist defined for service, and @raw-io is not included	
✓	SystemCallFilter=~@reboot	System call whitelist defined for service, and @reboot is not included	
✓	SystemCallFilter=~@swap	System call whitelist defined for service, and @swap is not included	
✗	SystemCallFilter=~@privileged	System call whitelist defined for service, and @privileged is included	0.2

----truncated----

→ **Overall exposure level for httpd.service: 6.7 MEDIUM**

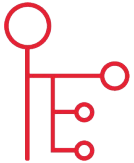
RESOURCE MANAGEMENT

SLICE, SCOPES, AND SERVICES



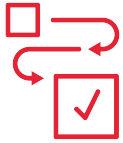
Slice

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

RESOURCE MANAGEMENT

Configuration

Configure cgroup attributes:

```
systemctl set-property --runtime httpd CPUShares=2048
```

Drop “--runtime” to persist (will create a drop-in):

```
systemctl set-property httpd CPUShares=2048
```

Or place in the unit file:

```
[Service]
CPUShares=2048
```

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

CONTROL GROUPS

Red Hat Enterprise Linux 8

cgroups v1—the default

- Well supported in the Linux ecosystem for over a decade
- Same basic behavior as Red Hat Enterprise Linux 7
  - systemd uses cgroups labels by default
  - Accounting is opt-in for CPU & BlockIO
- Memory and Tasks accounting is now enabled by default

- Same unit file options available: (*=deprecated)
  - `CPUAccounting=`, `*CPUShares=`, `CPUQuota=`
  - `MemoryAccounting=`, `*MemoryLimit=`
  - `*BlockIOAccounting=`, `*BlockIOWeight=`, `*BlockIODeviceWeight=`
  - `TasksAccounting=`, `TasksMax=`

https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt

CONTROL GROUPS

Red Hat Enterprise Linux 8

cgroups v2—tech preview

- Unified hierarchy with **vastly improved controllers**
  - Delivers more coherent and holistic resource management
- Perfectly integrated with systemd
  - Ecosystem in-progress (virt & containers work remains)
  - Fedora 31 may default to v2
  - Support planned for 8.1 or 8.2

- Append `systemd.unified_cgroup_hierarchy` to kernel
- Best effort translation for relevant controllers:
  - `CPUWeight=` replaces `CPUShares=`
  - `MemoryMax=` replaces `MemoryLimit=`
  - `IO*=` replaces `BlockIO*=`

https://www.kernel.org/doc/Documentation/cgroup-v2.txt

V2 CHEAT SHEET

v1	Min...**Default**...Max	v2	Min...**Default**...Max
`CPUShares=`	2...**1024**...262144	`CPUWeight=`	10...**100**...10000
`StartupCPUShares=`	2...**1024**...262144	`StartupCPUWeight=`	10...**100**...10000
`MemoryLimit=`	N/A	`MemoryMax=`	N/A
`BlockIOWeight=`	10...**500**...1000	`IOWeight=`	10...**100**...10000

CGROUP v2 CONTROLS

Red Hat Enterprise Linux 8

```
CPUWeight= CPUStartupWeight=
CPUQuota=

MemoryMin=
MemoryLow=
MemoryHigh=
MemoryMax=
MemorySwapMax=

IODeviceLatencyTargetSec=
IOWeight=
IODeviceWeight=
IOReadBandwidthMax= IOWriteBandwidthMax=
IOReadIOPSMax= IOWriteIOPSMax=
```

https://www.freedesktop.org/software/systemd/man/systemd.resource-control.html

UNPRIVILEGED UNITS

SYSTEMD -- USER

```
$ systemctl --user status
● localhost.localdomain
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Sat 2019-03-09 15:29:52 CST; 31min ago
   CGroup:
/user.slice/user-1000.slice/user@1000.service
           └─init.scope
             ├─1420 /usr/lib/systemd/systemd --user
             └─1427 (sd-pam)
```

SYSTEMD --USER

User units:

```
~/.config/systemd/user
```

Maintainer user units:

```
/usr/lib/systemd/user &
~/.local/share/systemd/user
```

Global user units (all users):

```
/etc/systemd/user
```

Note:

.bashrc & .bash_profile are not sourced by systemd

```
~/.config/environment.d
```

```
systemctl --user import-environment
```

```
systemctl --user show-environment
```

SYSTEMD --USER

- Interact with the systemd user instance
  - `systemctl --user`
  - `e.g. start, stop, restart, enable, disable, status`
  - `systemctl --user enable --now foo.service`
- Filter the journal by user unit(s)
  - `journalctl --user-unit=foo.service`
- Enable/disable systemd user outside of sessions (start on boot)
  - `loginctl enable-linger $USER`
  - `loginctl disable-linger $USER`
- “Shame back” view of user’s disgusting use of system resources
  - `loginctl user-status`

MISCELLANEOUS AWESOME STUFF

IP ACCOUNTING AND FILTERING

Technology Preview

`IPAccounting=`

- Ingress and egress IP traffic is counted for associated processes
- Applies to services, sockets, and slices
- Requires cgroup v2

`IPAddressAllow=`

- Filtering via cgroups eBPF hooks independent from iptables/nft
- IP/netmask for allowed traffic

`IPAddressDeny=`

- IP/netmask deny list

```
systemd-run -p
IPAddressAllow=10.0.0.5 -p
IPAddressDeny=any -t
mysqladm …
```

```
System-wide Example:

systemctl set-property
system.slice
IPAddressDeny=any
IPAddressAllow=localhost
```

http://0pointer.net/blog/ip-accounting-and-access-lists-with-systemd.html

MISCELLANEOUS AWESOME STUFF

Journal

- Better compression & performance
- Familiar filtering options
  - journalctl --grep=
- Additional color coding for log levels
  - Debug entries are light grey

Mount Options in /etc/fstab

- `x-systemd.growfs`
- `x-systemd.makefs`

systemctl

- Restart counter for units (Restart=)
  - `systemctl show -p NRestarts --value`
- Create a new unit file:
  - `systemctl edit --force foo.service`
- Reboot into UEFI Firmware setup:
  - `systemctl reboot --firmware-setup`

MISCELLANEOUS AWESOME STUFF

systemd-run

- `--pipe` use STDIN/STDOUT/STERR w/ transient units
- `--wait` for it to exit code

Unit files:

- `ExecStart` accepts a relative path
- Improved drop-in prefixes
- Clickable links with `--no-pager`

Parse, normalize, and calculate next occurance

```
systemd-analyze calendar '2019-05-8 11:45:00'
  Original form: 2019-05-8 11:45:00
Normalized form: 2019-05-08 11:45:00
    Next elapse: Wed 2019-05-08 11:45:00 EDT
       (in UTC): Wed 2019-05-08 15:45:00 UTC
       From now: 4 days left
```

Concatenate config files w/ drop-ins

```
systemd-tmpfiles --cat-config
```

```
systemd-analyze cat-config
/etc/systemd/journald.conf
```

HELPFUL RESOURCES

- RHEL documentation:
  https://access.redhat.com/site/documentation/Red_Hat_Enterprise_Linux/
- Demystifying systemd 2018: https://www.youtube.com/watch?v=tY9GYsoxeLg
- systemd project page: http://www.freedesktop.org/wiki/Software/systemd/
- Lennart Poettering's systemd blog series: (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
- Cgroups v2: https://www.kernel.org/doc/Documentation/cgroup-v2.txt
- Cgroups v2 @ facebook: https://facebookmicrosites.github.io/cgroup2/docs/overview

RED HAT SUMMIT

THANK YOU


linkedin.com/company/Red-Hat


facebook.com/RedHatinc

youtube.com/user/RedHatVideos


twitter.com/RedHat

PORTABLE SYSTEM SERVICES

Technology Preview

- Similar concepts as containers, except the focus is on the integration of units/apps as part of the system.
- When images or chroots are attached, the relevant units are copied to the system:

```
/usr/lib/systemd/portablectl attach
example.raw
```

- An alternative to system containers (atomic install in RHEL 7)
- Can use disk images (QCOW, RAW, etc) or chroots

- Services are managed as any other local unit file:

```
systemctl enable --now example.service
```

- Detaching images cleans up the units and restores the default state of the system

```
/usr/lib/systemd/portablectl detach
example.raw
```

http://0pointer.net/blog/walkthrough-for-portable-services.html