SUSE Linux Enterprise 12
Best Practice for systemd

Ralf Dannert
Systems Engineer
rdannert@suse.com
SUSE Linux GmbH

We adapt. You succeed.

Agenda 1

- systemd Basics
  - Units, Services, Targets
- Systemctl, systemd-analyze
- #SLES11SP3 – SLES12, service comparison
- #Presets
- Understanding service dependencies
- Socket activation
- The system journal
- dracut(used during system shutdown)

Agenda 2

- #Security
- Network
- Unit Generators
- #tmpfiles
- Systemd: Testing and debugging
  - Snapshot, isolate
  - systemd-nspawn
- #Convert A SysV Init Script Into A systemd Service File
- Control groups, slice, scope
- Appendix

systemd Basics

Apropos unification

- apropos systemd|wc -l
- 131

The role of systemd

- system- and session manager for Linux
- provides aggressive parallelization capabilities
- uses socket and D-Bus activation for starting services
- keeps track of processes using Linux cgroups
- Started directly by the Kernel
- Resists signal 9
- Replacement for System V init daemon
  - Fully compatible with System V init
  - Will revert to init scripts if no native config is found

Unit File

- File name extension defines Unit type
- Location: `/usr/lib/systemd/system/` and `/etc/systemd/system/`

Unit	Filename
Service	`<service>.service`
Targets	`<target>.target`
Sockets	`<socket>.socket`
Path	`<path>.path`
Timer	`<timer>.timer`
Mountpoint	`<mount>.mount`
Automount Point	`<automount>.automount`
Swap	`<swap>.swap`
Device	`<device>.device`
Scope/Slice	`<scope>.scope` `<slice>.slice`

Unit File Content

Sections:

- `[Unit]`
- `[Service]`
- `[Install]`

Option	Description
`Description`	Description
`After`	Start of the unit is delayed until all listed units have started up
`Before`	Inverse of `After`
`Requires`	The units listed here are activated as well

Service Unit Files

Option	Description
`Type`	Process start-up type Available types: `simple`, `forking`, `oneshot`, `dbus`, `notify`, `idle`
`EnvironmentFile`	File to read environment variables from
`ExecStart`	Command line (absolute path, with arguments) that is executed when this service is started
`ExecStartPre`, `ExecStartPost`	Additional commands that are executed before or after the command in `ExecStart`
`ExecReload`	Commands to trigger a configuration reload in the service
`ExecStop`	Commands to execute to stop the service
`KillMode`	How processes of this service shall be killed Available modes: `control-group`, `process`, `none`
`Restart`	Configures whether the main service process shall be restarted when it exits. Available options: `no`, `on-success`, `on-failure`, `on-abort`, or `always`

systemctl

- `systemctl` is the command to interact with systemd
- Without options, it displays the various units active on the system, with `-a` also the inactive units
- Some are not defined in files in `/usr/lib/systemd/system/`, they are created automatically, such as
  - various .mount units based on entries in `/etc/fstab`
  - other units based on rules in `/usr/lib/udev/rules.d/` that contain rules that contain `TAG+=systemd` entries

Manage Services

	Command
systemd	`systemctl <command> <service>.service`
System V	`rc<service> <command>`

systemd Command	Description	System V
`start`	Start service	`start`
`stop`	Stop service	`stop`
`restart`	Restart service	`restart`
`try-restart`	Restart service if it is running	`try-restart`
`reload`	Reload configuration without interrupting operation	`reload`
`reload-or-restart`	Reload service if it is supported, otherwise restart it	n/a
`reload-or-try-restart`	Reload service if it is supported, otherwise restart it if it is running	n/a
`status`	List detailed status information	`status`
`is-active`	List short status information	`status`

Enabling/Disabling Services

```
systemctl <command> <service>.service   or
systemctl <command> <service>
```

systemd Command	Description	System V
`enable`	Enable service	`insserv`
`disable`	Disable service	`insserv -r`
`is-enabled`	Check if service is enabled	`chkconfig`
`reenable`	Disable service and enable it afterwards	n/a
`mask`	After “disabling” a service, it can still be started manually or through a dependency, after masking it cannot be started at all	n/a
`unmask`	A service that has been masked can only be used again after it has been unmasked	n/a

/etc/sysconfig deprecated(1)

- **What is in /etc/sysconfig anyway?**
- Additional command line parameters for daemon
- Locale settings for daemon
- Shutdown time-out/mode for daemon
- system locale, time zone information, console keyboard
- CPU affinity for daemon
- service should start or not
- Network config
- kernel modules to statically load
- Access modes for device nodes (!)
- user/group ID, umask to run specific daemons as
- Resource limits for daemon
- OOM adjustment for daemon
  - http://0pointer.de/blog/projects/on-etc-sysinit.html

Systemd native alternatives for /etc/sysconfig

- Use Unit files
  - simple, declarative descriptions
  - Easy to modify
  - understand process context settings
- Use new common configuration files on all distros:
  - /etc/hostname, /etc/vconsole.conf, /etc/locale.conf, /etc/modules-load.d/*.conf, /etc/sysctl.d/*.conf, /etc/tmpfiles.d/*.conf, /etc/binfmt.d/*.conf, /etc/os-release, /etc/machine-id, /etc/machine-info
- Turn settings into native daemon settings
- /etc always intended to be the place for "Host-specific system configuration"(FHS)
- Use compatibility option:
  - **`EnvironmentFile=-/etc/sysconfig/foobar`**

    systemd.exec(5), systemd.service(5)

systemd Targets

Target Units

- Targets are synchronization points, similar to runlevels, but more fine-grained
- Each target is named instead of numbered and is intended to serve a specific purpose with the possibility of **having multiple ones active at the same time!**
- Some targets are implemented by inheriting all of the services of another target and adding additional services to it
- Create custom target:
  - Take one of the existing runlevels as a base /etc/systemd/system/yourtarget
  - make a directory /etc/systemd/system/yourtarget.wants, and then symlink the additional services from /usr/lib/systemd/system/ that you wish to enable
- Switching:
  - `systemctl isolate multi-user.target`
  - init 3, init 5, etc. still work
  - systemd.target(5), systemctl(1)

Target Units

Target Unit	Description	System V
default.target	Booted by default	
graphical.target (runlevel5.target)	System with network, multi-user support and a displaymanager	Runlevel 5
multi-user.target (runlevel3.target)	Multi-user system with network	Runlevel 3
multi-user.target (runlevel2.target)	Local multi-user system without network.	Runlevel 2
mail-transfer-agent.target	All services necessary for sending and receiving mails	
rescue.target (runlevel1.target)	Single user system without network	Runlevel 1 Runlevel S
emergency.target	Emergency shell on the console	
reboot.target (runlevel6.target)	Reboot the system	Runlevel 6
halt.target (runlevel0.target, poweroff.target)	Shut down the system	Runlevel 0

Systemctl target units

```
sles12migrated:~ # systemctl list-units --type=target
UNIT                    LOAD   ACTIVE SUB    DESCRIPTION
basic.target            loaded active active Basic System
cryptsetup.target       loaded active active Encrypted Volumes
getty.target            loaded active active Login Prompts
graphical.target        loaded active active Graphical Interface
local-fs-pre.target     loaded active active Local File Systems (Pre)
local-fs.target         loaded active active Local File Systems
multi-user.target       loaded active active Multi-User System
network-online.target   loaded active active Network is Online
network.target          loaded active active Network
nss-lookup.target       loaded active active Host and Network Name Lookups
nss-user-lookup.target  loaded active active User and Group Name Lookups
paths.target            loaded active active Paths
remote-fs-pre.target    loaded active active Remote File Systems (Pre)
remote-fs.target        loaded active active Remote File Systems
rpcbind.target          loaded active active RPC Port Mapper
slices.target           loaded active active Slices
sockets.target          loaded active active Sockets
swap.target             loaded active active Swap
sysinit.target          loaded active active System Initialization
time-sync.target        loaded active active System Time Synchronized
timers.target           loaded active active Timers

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.

21 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

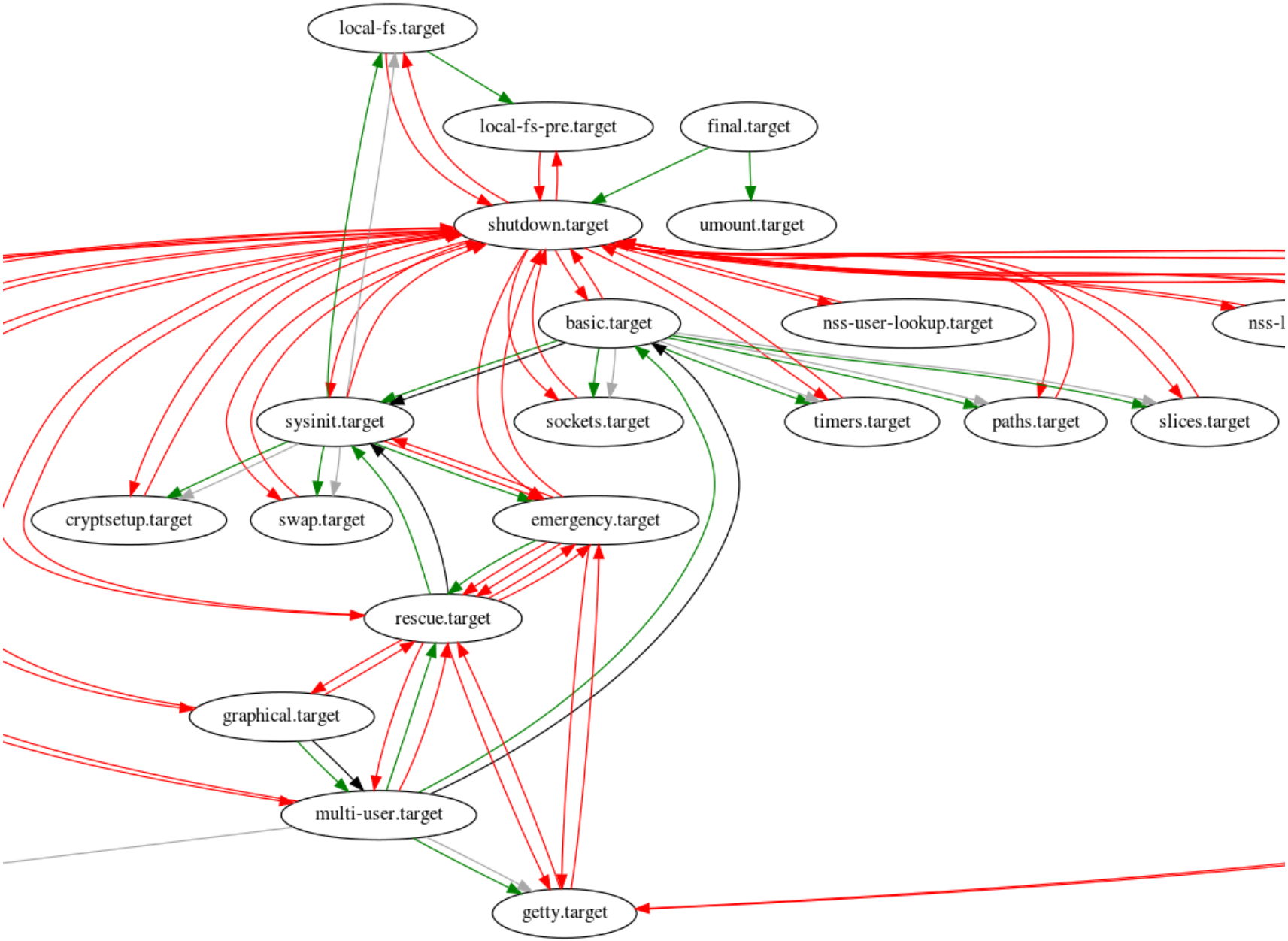
systemd.special(7)

Target dependencies

#Shows required and wanted units of the specified unit

```
systemctl list-dependencies|grep target
```

```
default.target
● └─multi-user.target
●   ├─basic.target
●   │ ├─paths.target
●   │ ├─slices.target
●   │ ├─sockets.target
●   │ ├─sysinit.target
●   │ │ ├─cryptsetup.target
●   │ │ ├─local-fs.target
●   │ │ └─swap.target
●   │ └─timers.target
●   ├─getty.target
●   └─remote-fs.target
●     └─remote-fs-pre.target
```



systemd-analyze dot --to-pattern='*.target' --from-pattern='*.target' | dot -Tsvg >/tmp/targets.svg

Change Targets

- Change the current target:
  `systemctl isolate <target>.target`
- Change to the default target:
  `systemctl default`
- Persistently change the default target:
  `systemctl set-default -f <target>.target`
- Get the current target:
  `systemctl list-units --type=target`
- Change the default target for the current boot process:
  `systemd.unit=<target>.target` at boot prompt
- Show a target's dependencies(which services are pulled in)
  `systemctl show -p "Requires" <target>.target`
  `systemctl show -p "Wants" <target>.target`
- examine what gets started when when booted into a specific target

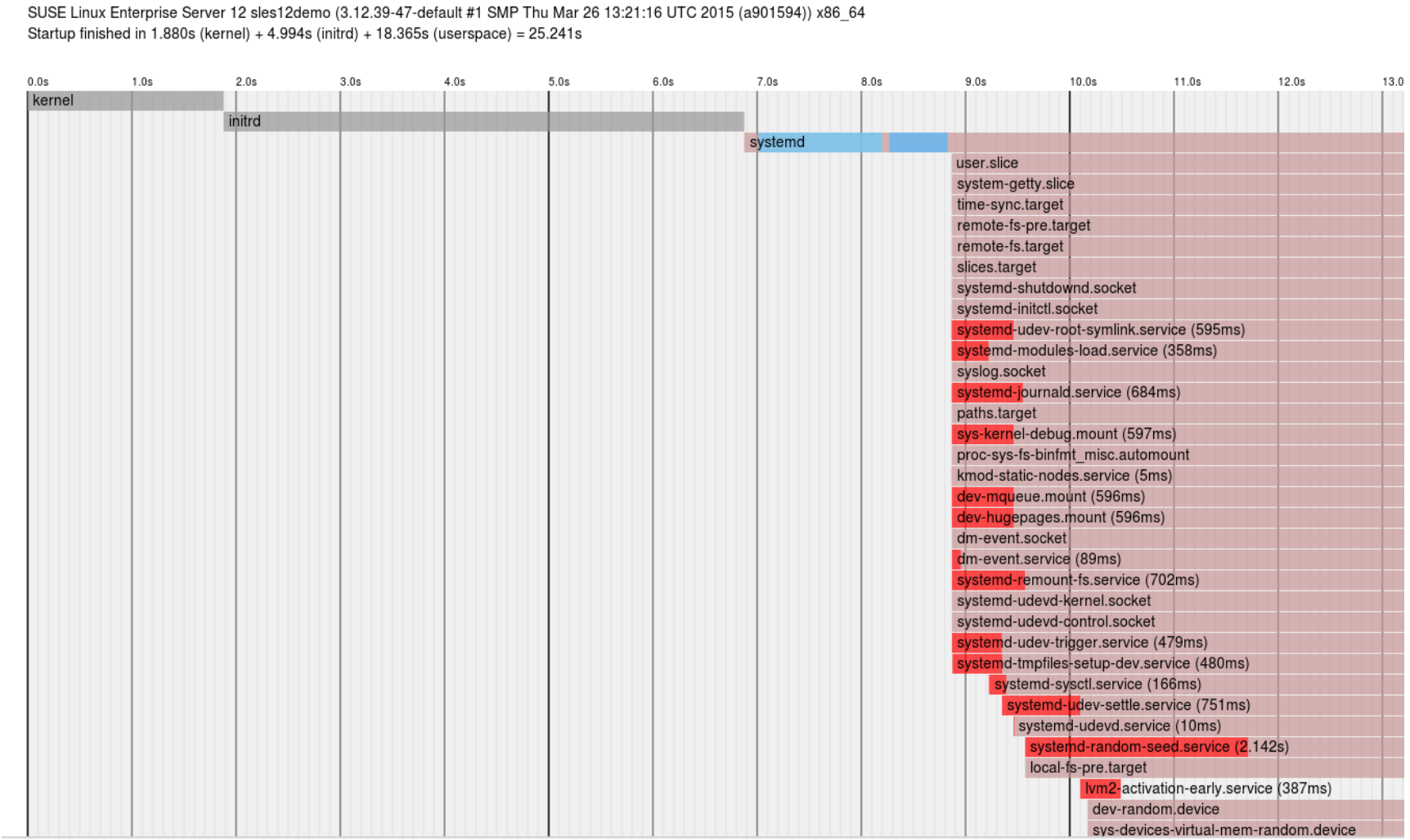
  `systemd --test --system --unit=multi-user.target`

Systemd: was nun?

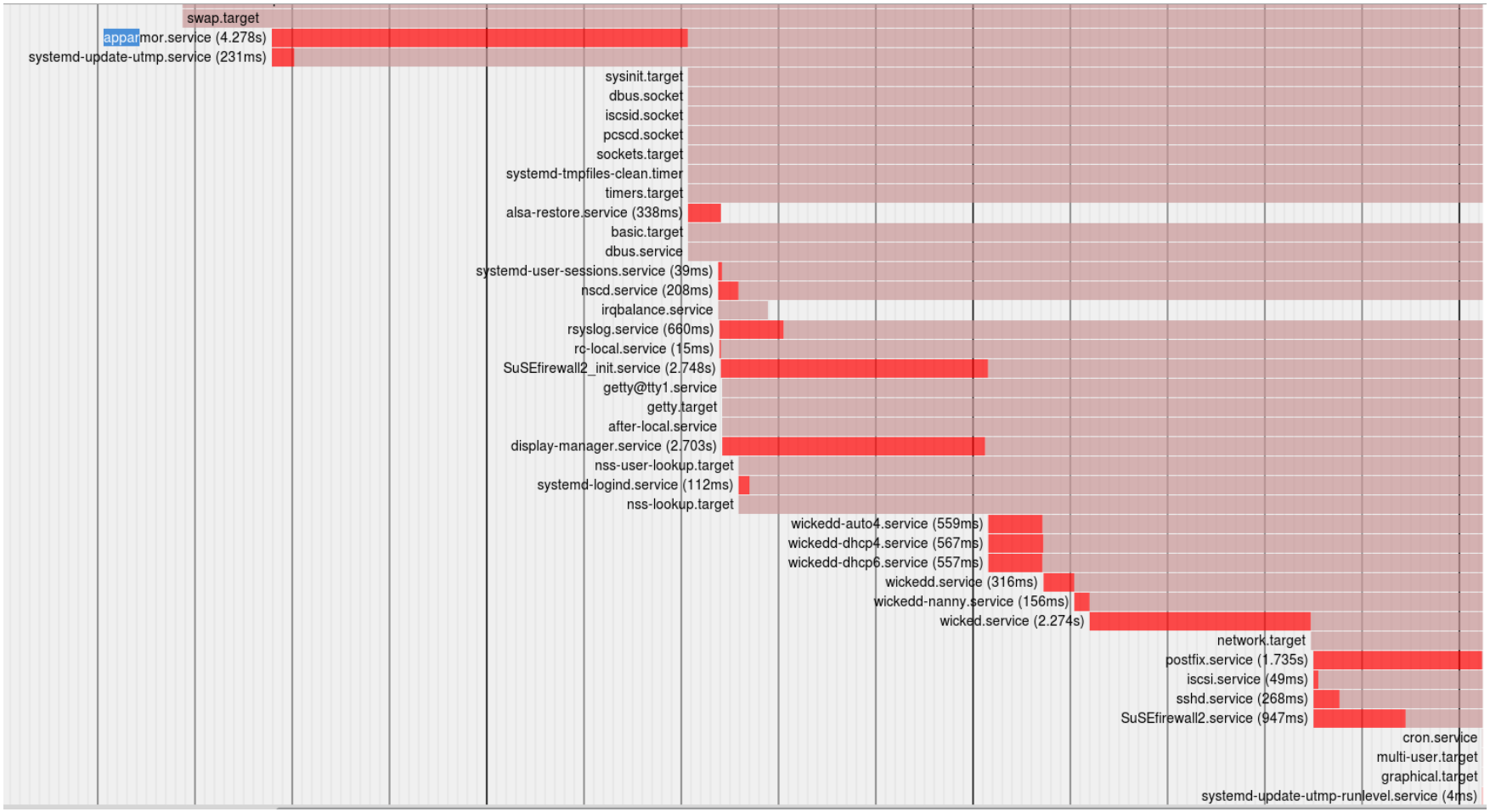
Analyze system boot-up performance systemd-analyze

- systemd-analyze blame
- systemd-analyze critical-chain
- systemd-analyze dump
  - complete server state of all loaded units, regardless of their state, including inactive units
  - systemctl -t service --all

boot-up performance systemd-analyze plot



boot-up performance
systemd-analyze plot



Upgrading to SLES12

search this guide

Deployment Guide > Manual Deployment > Updating SUSE Linux Enterprise > **Upgrading to SUSE Linux Enterprise 12**

Upgrading to SUSE Linux Enterprise 12 7.4

Upgrading from SUSE Linux Enterprise 11 SP3 (or higher) to SUSE Linux Enterprise 12 is supported using one of the following methods:

- Manual upgrade, booting from an installation medium (see Section 7.4.1, Manual Upgrade from SLE 11 SP3 to SLE 12, Using an Installation Source).
- Semi-automated migration, possible via SSH (see Section 7.4.2, Automated Migration from SLE 11 SP3 to SLE 12)

Manual Upgrade from SLE 11 SP3 to SLE 12, Using an Installation Source # 7.4.1

To upgrade your system this way, you need to boot from an installation source, like you would do for a fresh installation. However, when the boot screen appears, you need to select *Upgrade* (instead of *Installation*). The installation source to boot from can be one of the following:

https://www.suse.com/documentation/sles-12/book_sle_deployment/data/sec_update_sle12.html

Using machinery to upgrade to SLES12

- Requirements: Separate machine, where new system is set up and tested
- create a full description of SLES11 SP3
  - `machinery inspect -x $host_original`
  - `machinery copy $host_original $host_target`
- export the generated AutoYaST profile
  - `machinery export-autoyast $host_target –autoyast-dir=/tmp/my_migration/`
- install new system(still SLES11 SP3) using exported AutoYaST profile
- offline in-place upgrade to SLES12
  - `machinery inspect -x $host_target`
- Compare SLES12 with SLES11 SP3
  - `machinery compare --show-all $host_original $host_target`
- **Result:** upgraded SLES 12 on a new machine with same functionality as SLES11 SP3
- Adapt AutoYaST profile(delete unused rpms)
  - https://github.com/SUSE/machinery/wiki/How-to-upgrade-a-SLES-11-SP3-system-to-SLES-12

Different options to upgrade to SLES12

- Compare systems after upgrade from SLES11SP3 with pattern Minimal, base, lamp_server, mail_server, x11 to SLES12
- systemd-analyze plot >/tmp/plot.htm

- Using machinery:
- # create a full description
  - `machinery inspect -x $host_original`
  - `machinery copy $host_original $host_target`
- export the generated AutoYaST profile
  - `machinery export-autoyast $host_target –autoyast-dir=/tmp/my_migration/`
- Adapt autoyast profile(delete unused rpms)
  - https://github.com/SUSE/machinery/wiki/How-to-upgrade-a-SLES-11-SP3-system-to-SLES-12
  -

Systemd presets

Which Services to enable by Default

- What is a Service
  - A daemon or process started using a systemd service unit
  - A daemon or process that is invoked by socket activation, either by using a systemd socket unit, D-BUS activation or similar behavior
  - A systemd timer unit that runs periodically

Criteria:

- Locally running services
  - No manual configuration to be functional
  - does not listen on a network socket
  - Example: Local D-BUS services
- non-persistent services
  - Example: iptables

Welcome to systemd-presets

- Prior SLE12
  - whether a service is enabled or disabled after package installation is encoded in the %post scripts of RPM
- SLE12
  - rpm systemd-presets-branding-SLE-12.0
  - Packages updated to invoke "systemctl preset" in %post script of RPM
  - /usr/lib/systemd/system-preset/90-default-SLE.preset
- Services enabled by default:
    - acpid cron avahi-daemon YaST2-Second-Stage YaST2-Firstboot postfix
    - nscd purge-kernels ModemManager iscsid.socket iscsi libvirtd readonly-root
    - haveged irqbalance vmtoolsd iprdump iprinit iprupdate lvm2-lvmetad.socket
    - rtas_errd wpa_supplicant cio_ignore btrfsmaintenance-refresh
- systemctl preset <service-name>
- Alternatives: Implement policy(Debian style):
  - "echo 'enable *' > /etc/systemd/system-preset/50-foobar.preset"

Example exercise

- After a default installation of SLES12 enable additional service ntp

- Correct answer:
- Check, if RPM is enabled for change
- determine enabled services
- `$ systemctl list-unit-files`
- Drop in "enable ntp" in /etc/systemd/system-preset/50-foobar.preset
  - Check ntp package, if updated to invoke "systemctl preset" in rpm %post script
- `$ rpm -q --scripts ntp|grep preset`

- systemd.preset(5)

Understanding service dependencies

Service Dependencies 1

- 2 types of dependencies
- **Activation of units(Requires/Wants/Conflicts)**
- Show units that the specified unit requires or wants
  - `Systemctl list-dependencies <unit>`
- Show units that require or want the specified unit
  - `Systemctl list-dependencies--reverse <unit>`

- **Order of units(After/Before)**
- List units on which the specified unit has an “After” dependency
- This command show units, that need to be started before the specified unit
  - `Systemctl list-dependencies --after <unit>`
- This command show units, that need to be delayed until the specified unit starts
  - `Systemctl list-dependencies --before <unit>`

Unit dependencies
Example: nfs-server

```
sles12sp1test:~ # systemctl show -p  BindsTo nfs-mountd.service
BindsTo=nfs-server.service
sles12sp1test:~ # systemctl show -p  BindsTo nfs-idmapd.service
BindsTo=nfs-server.service
sles12sp1test:~ # systemctl list-dependencies --after nfs-server.service|grep nfs
nfs-server.service
● ├─nfs-config.service
● ├─nfs-idmapd.service
● ├─nfs-mountd.service
● ├─proc-fs-nfsd.mount
sles12sp1test:~ # systemctl list-dependencies --before nfs-server.service
nfs-server.service
● ├─rpc-statd-notify.service
● └─remote-fs-pre.target
●   └─remote-fs.target
●     ├─display-manager.service
●     └─systemd-user-sessions.service
sles12sp1test:~ # systemctl show -p After remote-fs.target
After=remote-fs-pre.target
sles12sp1test:~ # systemctl show -p Before remote-fs-pre.target
Before=remote-fs.target
```

Service Dependencies 2

- Design unit file correctly
  - Note that **Wants=** and **Requires=** do not imply **After=**, meaning that if **After=** is not specified, the two units will be started in parallel !
- `Systemctl show -p <property>`
  - Shows properties of the specified unit
- `systemctl cat PATTERN`(shell-style globs)
  - Show backing files of one or more units. Prints the "fragment" and "drop-ins"
  - `systemctl cat systemd-poweroff.service`
  - # /usr/lib/systemd/system/systemd-poweroff.service
- Neue Befehle:
  - `systemctl mask` - link these units to /dev/null, making it impossible to start them

systemd.unit(5)

How to hook units into the start-up of other units

- Along with a unit file foo.service, the directory **foo.service.wants/** may exist
- All unit files symlinked from such a directory are implicitly added as dependencies of type **Wants=** to the unit
- useful to hook units into the start-up of other units, without having to modify their unit files
- `systemctl enable NAME`
  - create a number of symlinks as encoded in "[Install]" sections of the unit files
- similar functionality exists for **Requires= type** dependencies as well, the directory suffix is .requires/ in this case.
- systemd.unit(5)

Override single entries in unit file

- Example: **`/usr/lib/systemd/system/tftpd.service`**
  - `[Service]`
  - `ExecStart=/usr/bin/in.tftpd -s /srv/tftp/`
  - don't edit global unit file **`/usr/lib/systemd/system/tftpd.service`**, changes will be lost with updates
  - Users don't want to create a full copy of **`/usr/lib/systemd/system/tftpd.service`** in **`/etc/systemd/system`**, because it will override all settings. Llater updates of **`/usr/lib/systemd/system/tftpd.service`** would be masked completely.
- Solution:
  - Create drop-in directory and a .conf file in **`/etc/systemd/system/tftpd.service.d/tftpdir.conf`**
  - `[Service]`
  - `#empty string assigned = list of commands to start is reset first`
  - `ExecStart=`
  - `ExecStart=/usr/bin/in.tftpd -s /some_other_directory/`

  - **`systemd-delta /etc`** (identify configuration files, that override)

https://bbs.archlinux.org/viewtopic.php?id=152950

Socket activation

Socket activated services and containers

- Mechanism:
  - systemd listens on sockets and activate services (again) next time they are connected to
  - idling after having processed connections process exit on their own
  - Unvisible for clients if service is currently running or not
  - service's IP socket stays continously connectable, no connection attempt ever fails, all connects will be processed promptly
- Example
  - with socket activated OS containers, the host's systemd instance listens on number of ports on behalf of containers, i.e. SSH, web and database
  - as first connection comes in, spawns container and pass to it all 3 sockets
- Benefit:
  - This setup lowers resource usage: services only running when needed, consume resources when required
  - Low profile for not often demanded services, i.e. web site hosters
  - hosting many sites on a single system only activating services as necessary (over-commit)
  - http://0pointer.de/blog/projects/socket-activated-containers.html

Socket activation

- Use cases
  - Socket activation for parallelization, simplicity, robustness
  - On-demand socket activation for singleton services
  - On-demand socket activation for per-connection service instances
- Advantages
  - no explicit dependency configuration necessary
  - sockets always available because initialized at boot before all other services
  - no userspace ordering of service start-up necessary
  - simplification of service development
  - Allows restart of services or upgrades while services stay continously available and responsive without losing any message

  - systemd.socket(5)
  - http://0pointer.de/blog/projects/inetd.html

Socket activation example 1

- In host:
- sles12:~ # **systemctl cat mycontainer**

```
# /etc/systemd/system/mycontainer.service
[Unit]
Description=start nspawn container for sshd
[Service]
ExecStart=/usr/bin/systemd-nspawn -bD
/var/tmp/bootstrap
opensuse132:~ # cat
/etc/systemd/system/mycontainer.socket
[Unit]
Description=The SSH socket of my little container
[Socket]
ListenStream=23
```

Socket activation example 2

sshd service and socket in container

```
nspawn-container:~ # systemctl cat sshd.socket
# /etc/systemd/system/sshd.socket
[Unit]
Description=SSH Socket for Per-Connection Servers
[Socket]
ListenStream=23
Accept=yes
```

```
nspawn-container:~ # systemctl cat sshd
# /usr/lib/systemd/system/sshd.service
[Unit]
Description=OpenSSH Daemon
After=network.target

[Service]
EnvironmentFile=-/etc/sysconfig/ssh
ExecStartPre=/usr/sbin/sshd-gen-keys-start
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always
[Install]
WantedBy=multi-user.target
```

The system journal

systemd-journald.service

- systemd-journald is a system service
- maintains structured, indexed journals
- Received from kernel, user processes(syslog call), standard error of system services, numerous metadata fields
- /run is used when /var/log/journal is not available
- log data is lost at reboot

Journal configuration

- journald.conf - Journal service configuration file
  - where to store journal data:
  - Storage=none #all log data received will be dropped
  - Storage=persistent #below /var/log/journal hierarchy
  - Storage=auto #as persistent only if /var/log/journal exists
  - Storage=volatile #/run/log/journal/
  - ForwardToSyslog=, ForwardToKMsg=, ForwardToConsole=
  - overridden at boot time with the kernel command line options "systemd.journald.forward_to_syslog="
  - systemd.log_level=debug equivalent debug
    - turns on the debug output from both the system manager and the kernel
- in grub2
  - GRUB_CMDLINE_LINUX="systemd.log_level=debug systemd.log_target=console systemd.journald.forward_to_console=yes"
  - journald.conf(5), systemd-journald.service(8), systemd(1)

Systemd boot parameters

- systemd.unit=
  - Overrides the unit to activate on boot, i.e.: rescue.target or emergency.target
- systemd.crash_shell=
  - Spawns a shell when crashes
- systemd.log_target=
  - Argument: console, syslog, kmsg, syslog-or-kmsg, null
- systemd.log_level=
  - Argument: emerg, alert, crit, err, warning, notice, info, debug

dracut(used during system shutdown)

initrd or initramfs

- initial ramdisk: temporary file system used in the boot process of the Linux kernel
- loading this file system into memory.
- used to make preparations before the real root file system can be mounted
  - HW detection
  - Device drivers as loadable modules
  - Tools used: udev, md scans, LVM, mount NFS

ordering of services
systemd used in dracut initramfs

```
                                   systemd-journal.socket
                                             |
                                             v
                                   dracut-cmdline.service
                                             |
                                             v
                                   dracut-pre-udev.service
                                             |
                                             v
                                    systemd-udevd.service
                                             |
                                             v
local-fs-pre.target                dracut-pre-trigger.service
         |                                   |
         v                                   v
 (various mounts)  (various swap  systemd-udev-trigger.service
         |           devices...)             |             (various low-level   (various low-level
         |               |                   |             services: seed,       API VFS mounts:
         v               v                   v             tmpfiles, random     mqueue, configfs,
  local-fs.target   swap.target     dracut-initqueue.service    sysctl, ...)        debugfs, ...)
         |               |                   |                   |                   |
         \_______________|_________________ | _________________|_________________/
                                            \|/
                                             v
                                        sysinit.target
                                             |
                              ______________/|\______________
                             /               |               \
                             |               |               |
                             v               |               v
                        (various             |            rescue.service
                       sockets...)           |               |
                             |               |               v
                             v               |            rescue.target
                     sockets.target          |
                             |               |
                             \_______________ |                              emergency.service
                                            \|                                       |
```

https://www.kernel.org/pub/linux/utils/boot/dracut/dracut.html#dracutbootup7

dracut-shutdown.service

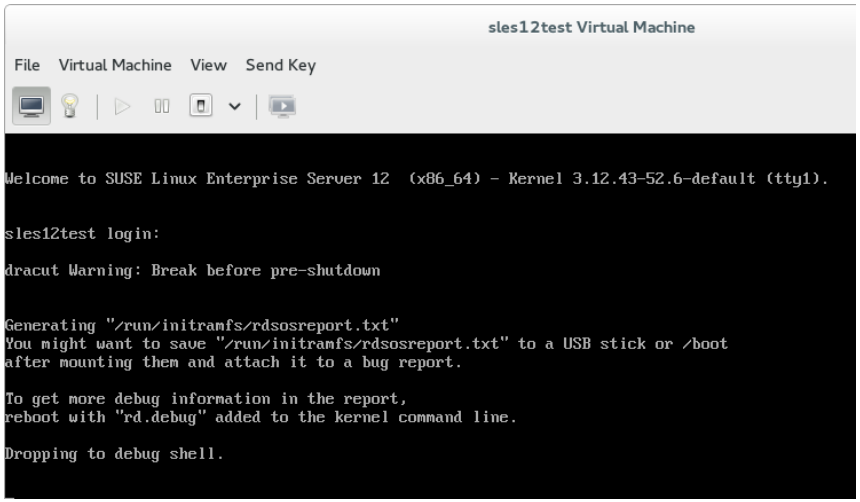
- **On a systemd driven system, the dracut initramfs is also used for the shutdown procedure!**
  - systemd switches to the shutdown.target
  - systemd starts `/usr/lib/systemd/system/shutdown.target.wants/dracut-shutdown.service`
  - dracut-shutdown.service executes `/usr/lib/dracut/dracut-initramfs-restore` which unpacks the initramfs to `/run/initramfs`
  - systemd finishes shutdown.target
  - systemd kills all processes
  - systemd tries to unmount everything and mounts the remaining read-only
  - systemd checks, if there is a `/run/initramfs/shutdown` executable
  - if yes, it does a pivot_root to `/run/initramfs` and executes `./shutdown`
  - The old root is then mounted on `/oldroot`
  - `/usr/lib/dracut/modules.d/99shutdown/shutdown.sh` is the shutdown executable.
  - shutdown will try to umount every `/oldroot` mount and calls the various shutdown hooks from the dracut modules
  - This ensures, that all devices are disassembled and unmounted cleanly.

debug the shutdown process
exercise

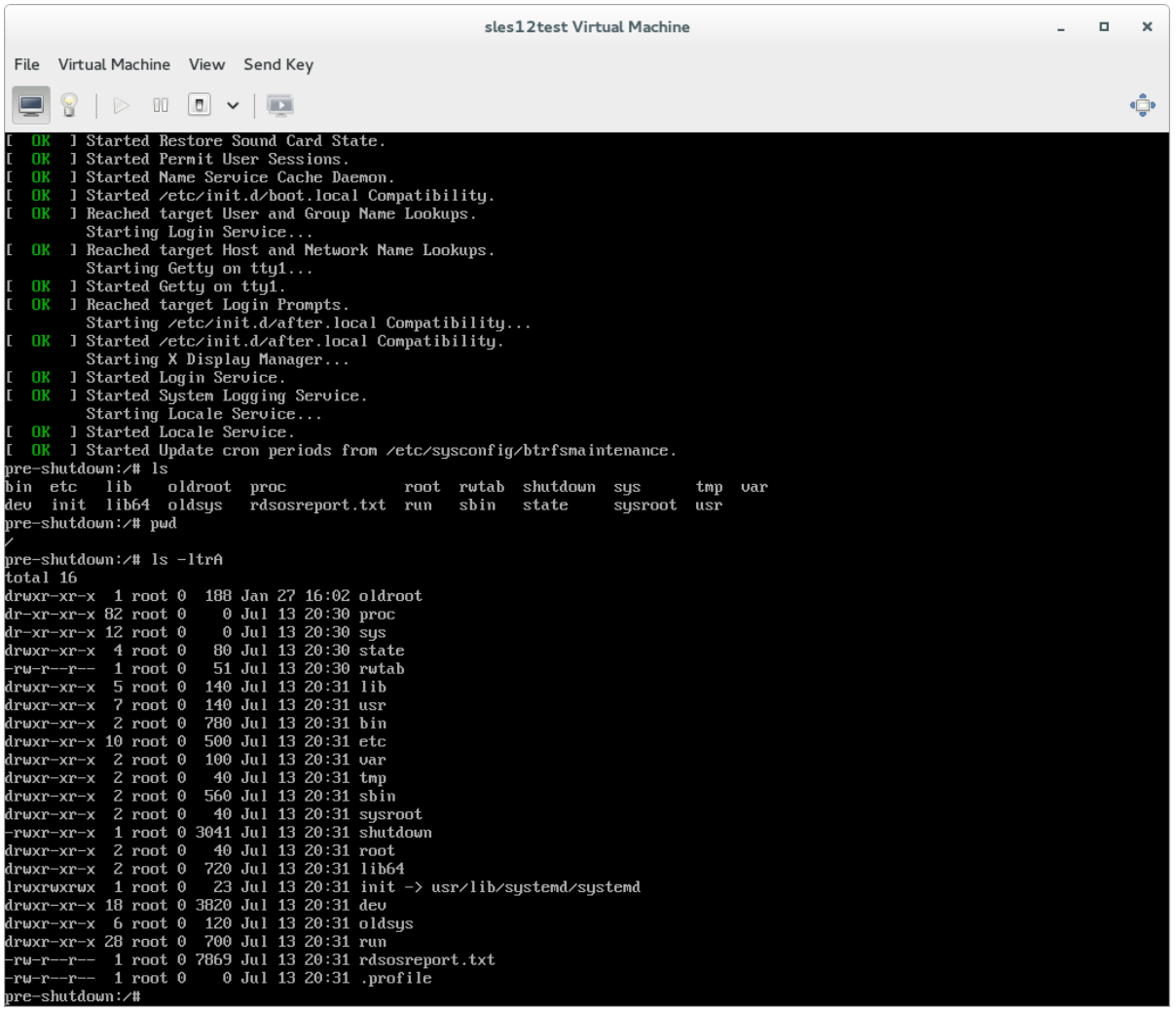
- get a shell in the shutdown procedure
  - `$ mkdir -p /run/initramfs/etc/cmdline.d`
  - `$ echo "rd.break=pre-shutdown rd.shell" > /run/initramfs/etc/cmdline.d/debug.conf`
  - `$ touch /run/initramfs/.need_shutdown`

- And provide the output of all of these commands:
  - `$ systemctl start dracut-shutdown.service`
  - `$ systemctl status dracut-shutdown.service`
  - `$ journalctl -u dracut-shutdown.service`
  - `$ bash -x /usr/lib/dracut/dracut-initramfs-restore`

Late shell during shutdown process



Late shell during shutdown process



- Make note of:
- oldsys
- shutdown
- rdsosreport.txt

Security

Securing Services with systemd

- Isolating services from the network
- Service-private /tmp
  - new file system namespace for the executed processes
  - temporary data created by service will be removed after service is stopped
- Making directories appear read-only or inaccessible to services
  - Taking away capabilities from services
- Disallowing forking, limiting file creation for services
- Controlling device node access of services
- http://0pointer.de/blog/projects/security.html
- systemd.exec(5), capabilities(7)

Unit File Sections: Secure Services

- Applies to [Service], [Socket], [Mount], [Swap] sections / unit types
  - Limit network access (namespace)
    - PrivateNetwork=yes
  - Private /tmp
    - PrivateTmp=yes
  - Restrict access to directories
    - InaccessibleDirectories=/home
    - ReadOnlyDirectories=/var
  - Restrict capabilities
    - CapabilitiesBoundingSet=CAP_CHOWN CAP_KILL
    - CapabilitiesBoundingSet=-CAP_PTRACE (all but this one)

Network

Network Concepts In Systemd

- network.target
  - guarantees that the network service has been started
  - Undefined if network interfaces are already configured when reached
  - primary purpose: ordering things properly at shutdown
- network-online.target
  - target that actively waits until the nework is "up"
- network-pre.target

- systemd.special(7)

network-online.target

- network-online.target is a target that actively waits until the nework is "up", where the definition of "up" is defined by the network management software.
- Definition: configured, routable IP address
- Purpose: actively delay activation of services until the network is set up
- **active target**: pulled in by the services requiring the network to be up
  - but not pulled in by the network management service itself
  - remote mounts defined in /etc/fstab pull this service in
  - make sure network is up before it is attempted to connect to a network share.
  - http://www.freedesktop.org/wiki/Software/systemd/NetworkTarget/

network-pre.target

- pulled in by services that want to run before any network is set up, i.e.: setting up a firewall(shorewall)
- passive unit(cannot start directly)
- not pulled in by network management service, but by the service that wants to run before it
- Network management services should set **After=network-pre.target**
- Services that want to be run before the network is configured should place **Before=network-pre.target** and also set **Wants=network-pre.target** to pull it in
- Result: avoid unnecessary synchronization points

Systemd unit generators

Systemd generators

- should only be used to generate unit files
- execute binaries early at bootup and at configuration reload time before unit files are loaded
- dynamically generate unit files
- create symbolic links to unit files to add additional dependencies
  - extending or overriding existing definitions
  - main purpose: convert configuration files that are not native unit files dynamically into native unit files.
  - **/usr/lib/systemd/system-generators/**
- `systemctl daemon-reload`
  - systemd.generator
  - systemd-fstab-generator (8)
  - systemd.mount(5)

http://www.freedesktop.org/software/systemd/man/systemd.generator.html

Mount unit configuration

- /etc/fstab mount options understood by systemd
- systemd will create a dependency of type Wants from either local-fs.target or remote-fs.target, depending whether the file system is local or remote
- **nofail**
  - this mount will be only wanted, not required, by the local-fs.target
  - boot will continue even if this mount point is not mounted successfully
- **fail**
  - opposite meaning and is default
- **noauto**
  - mount will not be added as a dependency for local-fs.target
  - will not be mounted automatically during boot, unless pulled in by some other unit
- **auto**
  - stricter handling of failing "auto" mounts during boot
  - **failing to mount an "auto" mount (without "nofail" option), systemd will drop to an emergency shell rather than continuing the boot!**
- systemd.mount(5) systemd.mount(5)

lvm2-activation-generator(1)

- The **lvm2 activation generator** generates systemd units conditionally based on the global/use_lvmetad **lvm.conf** setting
- If **use_lvmetad=0**, the lvm2-activation-early.service and lvm2-activation.service units will be generated
- These units are responsible for direct volume activation by calling "`vgchange -aay --sysinit`" (this is actually the original on-boot activation as it was used before)
- If **use_lvmetad=1**, no units will be generated as we're relying on autoactivation

- Important thing to note is that the lvm2-activation units normally bring in the udev-settle ("storage-wait") service that waits for udev to settle (with block devices)
- We don't need this if lvmetad is used in conjunction with autoactivation feature... but systemd units can't be enabled or disabled (or dependencies added/removed) dynamically based on external configuration
- Therefore, we need the unit generator which adds support for such situations: the units as a whole either exist or not based on the external configuration

lvm2-activation-generator(2)

- systemd units generated by lvm2-activation-generator:
- lvm2-activation-early.service
  - responsible for direct activation of LVM2 logical volumes if lvmetad daemon is not used
  - Direct LVM2 activation requires udev to be settled
- lvm2-activation.service
  - activation of LVM2 volumes ordered after cryptsetup.target(layered on top of encrypted devices)
- lvm2-activation-net.service
  - activation of LVM2 volumes ordered after remote-fs.target(layered on attached remote devices)

lvm2-activation-generator(8)

Lvm activation

- exclude LVs or VGs from being started automtically
  - https://www.suse.com/support/kb/doc.php?id=7016683
- Do not activate other VGs or LVs, that are not listed in the activation filter:
- in /etc/lvm/lvm.conf define _both_ parameters with the VGs or LVs you want activated on boot

```
activation {
  volume_list = [ "vgtest" ]
  auto_activation_volume_list = [ "vgtest" ]
}
```

- # systemctl status lvm2-activation-early.service
- Oct 22 21:19:26 sles12 lvm[657]: 0 logical volume(s) in volume group "vg1" now active
- Oct 22 21:19:26 sles12 lvm[657]: 1 logical volume(s) in volume group "vgtest" now active

tmpfiles

Clear tmp directories separately

- **systemd-tmpfiles** creates, deletes, and cleans up volatile and temporary files and directories
- **systemd-delta**
  - [OVERRIDDEN] /etc/tmpfiles.d/tmp.conf → /usr/lib/tmpfiles.d/tmp.conf
  - # Clear tmp directories separately, to make them easier to override
  - # SUSE policy: we don't clean those directories
  - `d /tmp 1777 root root -`
  - `d /var/tmp 1777 root root -`
  - # Exclude namespace mountpoints created with PrivateTmp=yes
  - `x /tmp/systemd-private-%b-*`
  - `X /tmp/systemd-private-%b-*/tmp`
  - `x /var/tmp/systemd-private-%b-*`
  - `X /var/tmp/systemd-private-%b-*/tmp`
- #OWNER_TO_KEEP_IN_TMP="root"
- tmpfiles.d(5), systemd-tmpfiles(8)

https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_boot_systemd_advanced.html

Systemd: Testing and debugging

Debugging with systemd

- systemctl snapshot
  - saved state of the systemd manager
  - useful for testing various targets
- systemctl isolate
  - changing runlevel in a traditional init system
  - isolate command will immediately stop processes that are not enabled in the new unit
- Fix problem in emergency target and return to previous target
  - `$ systemctl snapshot saved.snapshot`
  - `$ systemctl|grep "loaded units" #49` loaded units listed.
  - `$ systemctl isolate emergency.target`
  - `$ systemctl|grep "loaded units" #13` loaded units listed.
  - `$ systemctl isolate saved.snapshot`
  - `$ systemctl|grep "loaded units" #50` loaded units listed.
- systemctl(1)

Systemd-nspawn

- spawn a namespace container for debugging, testing and building
- not suitable for secure container setups
- set up minimal OS directory tree
  - `mkdir -p /var/tmp/bootstrap/etc/repos.d`
  - `cp /etc/zypp/repos.d/SLES12-12-0.repo /var/tmp/bootstrap/etc/repos.d/`
  - `zypper --root /var/tmp/bootstrap/ install --no-recommends systemd syslinux perl-Bootloader-YAML zypper`
  - `rm /var/tmp/bootstrap/etc/securetty`
  - `systemd-nspawn -bD /var/tmp/bootstrap 3`
- From host:
  - `systemd-analyze -M <nspawn_container_name>`
  - Startup finished in 675ms (userspace) = 675ms
- Test your services in container

Understand boot process with systemd-nspawn using debug

- $ `/usr/bin/systemd-nspawn -D <rootfs_dir> /bin/systemd systemd.log_level=debug`
  - spawned /usr/lib/systemd/system-generators/systemd-fstab-generator
  - Priority of unit files
  - load configuration failures
  - Creation of private D-Bus server
  - Activating default.target
  - Ignoring failed dependency jobs
  - Installing targets, service, sockets, mounts,..

Systemd general debugging guidelines

- discover journals of local containers and interleave them on display
  - `$ journalctl -m`
  - Useful for debugging of cluster or combined web/DB setups
- Change current log level of systemd daemon
  - `$ systemd-analyze set-log-level debug`
- Booting into Emergency Target
  - #remount root filesystem read-write
  - `systemctl isolate emergency.target`
  - `mount -o remount,rw /`
- Run a copy of the host system in a btrfs snapshot
  - `btrfs subvolume snapshot / /.tmp`
  - `systemd-nspawn --private-network -D /.tmp -b`

http://freedesktop.org/wiki/Software/systemd/Debugging/
http://www.it3.be/2015/05/15/systemd-spawn-rear/

Systemd general debugging guidelines

- test rear rescue image
  - -d option in rear keeps temporary directories intact
  - `$ rear -vd mkrescue`
  - rootfs directory contains a full populated rescue image
  - `$ systemd-nspawn -bD /tmp/rear.6PzAIYKf22D4uOL/rootfs/`

http://www.it3.be/2015/05/15/systemd-spawn-rear/

Journal: Filtering output

- Show all messages from this boot
  - `$ journalctl -b`
  - `$ journalctl -b -1` #from the previous boot
  - `$ journalctl --since="2015-05-30 12:10:11"`
- Show all messages by a specific executable:
  - `$ journalctl /usr/lib/systemd/systemd`
- Show all messages by a specific process:
  - `$ journalctl _PID=1`
- Show all messages by a specific unit:
  - `$ journalctl -u postfix`
- Show kernel ring buffer aka dmesg
  - `$ journalctl -k`
- journal options based on metadata: trusted journal fields
- systemd.journal-fields(7)

Early Debug Shell

- enable shell access to be available very early in the startup process to fall back on and diagnose systemd related boot up issues
  - `$ systemctl enable debug-shell.service`

- enable service manually
  - `$ cd $PATH_TO_YOUR_ROOT_FS/etc/systemd/system`
  - `mkdir -p sysinit.target.wants`
  - `ln -s /usr/lib/systemd/system/debug-shell.service sysinit.target.wants/`
- On next boot switch to tty9 using CTRL+ALT+F9
- Have root shell
- **Warning:**
- After debugging is finished
  - `$ systemctl disable debug-shell.service`

- https://freedesktop.org/wiki/Software/systemd/Debugging/

systemd-halt.service

- pulled in by halt.target
- responsible for the actual system halt
  - PID 1 is replaced by `/usr/lib/systemd/systemd-shutdown`
    - unmount all remaining file systems
    - disable all remaining swap devices
    - detach all remaining storage devices
    - kill all remaining processes
    - run all executables in `/usr/lib/systemd/system-shutdown/`
    - executables in this directory are executed in parallel
    - execution of the action is not continued before all executables finished

Convert A SysV Init Script Into A systemd Service File

Adapting SysV Init script

- Example: /etc/init.d/named starts
  - `/usr/sbin/named -t /var/lib/named -u named`
- Although not a native systemd service, standard adaptations apply, here: convert to use IPv4 only
  - cat /etc/systemd/system/named.service.d/named.conf (drop-in directory)
  - ExecStart=
  - ExecStart=/usr/sbin/named -4 -t /var/lib/named -u named

  - Last line will override default after
  - `systemctl daemon-reload`
  - `systemctl restart named.service`

- How Do I Convert A SysV Init Script Into A systemd Service File?
- http://0pointer.de/blog/projects/systemd-for-admins-3.html
- daemon(7) - Writing and packaging system daemons

systemd conversion
Use case: upgrade from SLES-11-SP3

- automatic conversion of sysv boot init scripts is too dangerous
- Macros provided when a package installs systemd unit files
- Save and Restore SysV Service Runlevel Information
- service migration uses helper script in %pre and %post section
  - `/usr/sbin/systemd-sysv-convert --save $services_to_migrate`

- if initial install is under systemd, migration is disabled
- `touch /var/lib/systemd/migrated/$sysv_service`
- sles12test:~ # ls /var/lib/systemd/migrated/|wc -l
- 69
- systemd reloads its unit files on installation

Example: rpm -q --scripts ipvsadm
preinstall scriptlet

- test -n "$FIRST_ARG" || FIRST_ARG=$1
- **# disable migration if initial install under systemd**
- [-d /var/lib/systemd/migrated] || mkdir -p /var/lib/systemd/migrated || :
- if [$FIRST_ARG -eq 1]; then
- for service in ipvsadm.service ; do
- sysv_service=${service%.*}
- **touch "/var/lib/systemd/migrated/$sysv_service"** ||:
- done
- else
- if [$FIRST_ARG -gt 1]; then
- for service in ipvsadm.service ; do
- if [! -e "/usr/lib/systemd/system/$service"]; then
- touch "/run/rpm-ipvsadm-update-$service-new-in-upgrade"
- fi
- done
- fi
- for service in ipvsadm.service ; do
- sysv_service=${service%.*}
- if [! -e "/var/lib/systemd/migrated/$sysv_service"]; then
- services_to_migrate="$services_to_migrate $sysv_service"
- fi
- done
- if [-n "$services_to_migrate"]; then
- /usr/sbin/systemd-sysv-convert --save $services_to_migrate >/dev/null 2>&1 || :
- fi
- fi

New-Style Daemons
Writing and packaging system daemons

- Provide correct exit code
  - LSB recommendations for SysV init scripts
- integration in systemd
  - provide a .service unit file(start, stop, maintain)
  - systemd.service(5)
- rely on systemd's resource limit control
  - systemd.exec(5)
- D-Bus service activation guarantees:
  - on-demand starting of services
  - may be started in parallel
  - Restarts on failure without losing any bus requests
    - all requests are queued while the daemon cannot process them
    - socket stays bound and accessible during restart
  - Daemon(7), systemd-notify(1)

init scripts: legacy actions

- **What:** init scripts that have historically defined custom actions
- Init scripts sometimes implemented additional actions besides the usual start/stop/status etc. For systemd service files that extra feature doesn't exist!
- **actiondir="/usr/lib/initscripts/legacy-actions"**

- Example:
  - **Old:** init script "foo" had an action "frob"
  - **New:** service file is now called "foo.service"
  - To support “frob” action implement feature here:
  - /usr/lib/initscripts/legacy-actions/foo/frob
  - legacy actions are called by /usr/sbin/service
- best practice:
  - Don't package a legacy action for new scripts or actions that were not supported by the prior init script
  - intended for compatibility with existing scripts only
  - https://en.opensuse.org/openSUSE:Systemd_packaging_guidelines

Control group, slice, scope

Systemd and control groups

- **Service, scope and slice units directly map to objects in the cgroup tree!**

- **Slice unit**
  - organize a hierarchy in which scopes and services are placed
  - set defaults for the whole tree
- **Scope unit**
  - manage externally created processes
  - started and stopped by arbitrary processes via fork()
  - registered by systemd at runtime using bus interfaces

- **system.slice**
  - service and scope units
- **machine.slice**
  - All virtual machines and containers registered with systemd-machined
- **user.slice**
  - all user processes and services started on behalf of the user, including the per-user systemd instance
  - user sessions handled by systemd-logind

systemd.special(5), systemd.slice(5) /usr/src/linux/Documentation/cgroups/cgroups.txt

Transient units - systemd-run

- wrapper around StartTransientUnit()
- run process as a transient service in the background invoked from PID 1
- alternatively run as a scope unit in foreground - run from the systemd-run process itself

```
- sles12:~ # systemctl cat limits.slice
- # /etc/systemd/system/limits.slice
- [Unit]
- Description=Limited resources Slice
- DefaultDependencies=no
- Before=slices.target
- [Slice]
- CPUShares=512
- MemoryLimit=256M
- # /etc/systemd/system/limits.slice.d/90-MemoryLimit.conf
- [Slice]
- MemoryLimit=104857600

- sles12:~ # systemctl show -p MemoryLimit limits.slice
- MemoryLimit=104857600
```

Systemd-run example

```
- systemd-run --unit=mem_hung --scope --slice=limits python

- sles12:~ # systemd-cgls
- ├─limits.slice
- │ └─mem_hung.scope
- │   └─3461 /usr/bin/python
```

systemd.slice(2)

- **machinectl**
  - -M, --machine=
  - Execute operation on a local container
  - machinectl may be used to introspect and control the state of the systemd(1) virtual machine and container registration manager systemd-machined.service(8).
- **systemctl**
  - -M, --machine=
  - Execute operation on a local container. Specify a container name to connect to.
  - -H, --host=
  - Execute the operation remotely. Specify a hostname, or username and hostname separated by "@", to connect to. This will use SSH to talk to the remote machine manager instance.
- **journalctl**
  - -M, --machine=
  - Show messages from a running, local container. Specify a container name to connect to.

control group hierarchy in systemd-nspawn container

```
linux:~ # systemd-cgls
|-1 /usr/lib/systemd/systemd 3
|-system.slice
| |-dbus.service
| | `-127 /bin/dbus-daemon --system --address=systemd:
| |-systemd-journald.service
| | `-92 /usr/lib/systemd/systemd-journald
| |-systemd-logind.service
| | `-128 /usr/lib/systemd/systemd-logind
| `-dm-event.service
|   `-120 /sbin/dmeventd
`-user.slice
  `-user-0.slice
    |-session-33.scope
    | |-130 login -- root
    | |-135 -bash
    | |-180 systemd-cgls
    | `-181 systemd-cgls
    `-user@0.service
      |-129 /usr/lib/systemd/systemd --user
      `-131 (sd-pam)
linux:~ #
```

Appendix

Systemd man-pages shortlist

- bootup (7) - System bootup process
- init (1) - systemd system and service manager
- journalctl (1) - Query the systemd journal
- systemctl (1) - Control the systemd system and service manager
- systemd (1) - systemd system and service manager
- systemd-analyze (1) - Analyze system boot-up performance
- systemd-delta (1) - Find overridden configuration files
- systemd-fstab-generator (8) - Unit generator for /etc/fstab
- systemd-journalctl (1) - Query the systemd journal
- systemd-nspawn (1) - Spawn a namespace container for debugging, testing and building
- systemd-tmpfiles (8) - Creates, deletes and cleans up volatile and temporary files and directories
- **systemd.directives (7) - Index of configuration directives**
- systemd.exec (5) - Execution environment configuration
- systemd.mount (5) - Mount unit configuration
- systemd.preset (5) - Service enablement presets
- systemd.resource-control (5) - Resource control unit settings
- systemd-run (1) - Run programs in transient scope or service units
- systemd.service (5) - Service unit configuration
- systemd.special (7) - Special systemd units
- systemd.target (5) - Target unit configuration
- systemd.unit (5) - Unit configuration
- dracut-shutdown.service (8) - unpack the initramfs to /run/initramfs

Additional material

- https://github.com/rdannert/systemd/
- some cmdline tools to explore systemd
- will be updated irregularly

Links

https://www.suse.com/de-de/documentation/sled-12/book_sle_admin/data/cha_systemd.html

systemd in SUSE Linux Enterprise 12
https://www.suse.com/docrep/documents/huz0a6bf9a/systemd_in_suse_linux_enterprise_12_white_paper.pdf

SUSECon: Systemd Intro by Olaf Kirch
www.susecon.com/doc/2014/sessions/TUT7563.pdf

multipath integration in systemd:
https://www.suse.com/documentation/sles-12/stor_admin/data/sec_multipath_trouble.html

Systemd Generators:
http://www.freedesktop.org/software/systemd/man/systemd.generator.html (since systemd 220)
https://www.mankier.com/8/lvm2-activation-generator

Systemd Presets:
http://lists.freedesktop.org/archives/systemd-devel/2011-July/002830.html

Systemd network targets
http://www.freedesktop.org/wiki/Software/systemd/NetworkTarget/
http://freedesktop.org/wiki/Software/systemd/Optimizations/

How Do I Convert A SysV Init Script Into A systemd Service File?
http://0pointer.de/blog/projects/systemd-for-admins-3.html

Systemd blog articles by Lennart Poettering
http://lmgtfy.com/?q=systemd+site%3A0pointer.net%2Fblog%2F

We adapt. You succeed.

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org