systemd

Lucas Nussbaum
lucas.nussbaum@univ-lorraine.fr

Licence professionnelle ASRALL

Administration de systèmes, réseaux et applications à base de logiciels libres

Índice

Sistema de arranque

- El primer proceso iniciado por el núcleo (pid 1)
- Responsable de levantar el resto del espacio de usuario
  - Montar sistemas de archivos
  - Iniciar servicios
  - ...
- Es también el padre de los procesos huérfanos
- Sistema de arranque tradicional en Linux: sysVinit
  - Heredado de Unix System V
  - Con herramientas adicionales (insserv, startpar) para el manejo de dependencias e inicialización en paralelo

systemd

- Escrito (a partir de 2010) por Lennart Poettering (Red Hat) y otros
- Hoy en día, el default en la mayor parte de las distribuciones de Linux
- Cambia el enfoque de *iniciar todos los servicios* (sysVinit) a administrar el sistema y todos sus servicios
- Características clave:
  - Basado en cgroups para
    - Supervisión de servicios
    - Control del ambiente de ejecución de servicios
  - Sintaxis declarativa para los archivos de unidad $\rightsquigarrow$ más eficiente/robusto
  - Activación por sockets para iniciar los servicios en paralelo
  - Interfaz usuario más agradable (systemctl & amigos)
- Características adicionales: Bitácora, unidades de temporizador (como cron), manejo de sesiones de usuario, administración de

Detrás de bambalinas: cgroups

- Abreviado de *grupos de control*
- Característica del núcelo de Linux
- Limita, contabiliza y aísla a los procesos y su uso de recursos (CPU, memoria, disco, E/S, red, etc.)
- Relacionado con el aislamiento del espacio de nombres:
  - Aísla a los procesos del resto del sistema
  - *Chroots en esteroides*
  - PID, red, UTS, montajes, usuario, etc.
- LXC, Docker $\approx$ cgroups + espacios de nombres (+ herramientas de administración)

cgroups y systemd

- Cada servicio corre dentro de su propio cgroup
- Permite:
  - Monitorear y matar a todo proceso creado por cada servicio
  - Contabilidad y asignación/límite de recursos por servicio
- Antes, con sysVinit:
  - No se daba seguimiento a qué servicio inició cuáles procesos
    - Archivos PID, o hacks en scripts: `pidof` / `killall` / `pgrep`
    - Dificil terminar por completo a un servicio (CGIs remanentes tras matar a Apache)
  - Sin límites de recursos (o utilizando `setrlimit` (= `ulimit`), que es por proceso, no por servicio
- También aislar sesiones de usuario $\rightsquigarrow$ matar a todos los procesos de un usuario (no por default)
- Más información: Control Groups vs. Control Groups y Which Service Owns Which Processes?

`systemd-cgls`: Visualizando la jerarquía de cgroups

```
├─1 /sbin/init
├─system.slice
│ ├─apache2.service
│ │ ├─1242 /usr/sbin/apache2 -k start
│ │ ├─9880 /usr/sbin/apache2 -k start
│ │ └─9881 /usr/sbin/apache2 -k start
│ ├─system-getty.slice
│ │ ├─getty@tty1.service
│ │ │ └─1190 /sbin/agetty --noclear tty1 linux
│ │ └─getty@tty2.service
│ │   └─24696 /sbin/agetty --noclear tty2 linux
│ ├─system-postgresql.slice
│ │ └─postgresql@9.4-main.service
│ │   ├─1218 /usr/lib/postgresql/9.4/bin/postgres -D /var/lib/postgresql/9.4/main -c
│ │   ├─1356 postgres: checkpointer process
│ │   ├─1357 postgres: writer process
│ │   ├─1358 postgres: wal writer process
│ │   ├─1359 postgres: autovacuum launcher process
│ │   └─1360 postgres: stats collector process
│ ├─gdm.service
│ │ ├─1209 /usr/sbin/gdm3
│ │ └─1238 /usr/bin/Xorg :0 -novtswitch -background none -noreset -verbose 3 -auth /
└─user.slice
  └─user-1000.slice
    ├─session-1.scope
    │ ├─ 1908 gdm-session-worker [pam/gdm-password]
    │ ├─ 1917 /usr/bin/gnome-keyring-daemon --daemonize --login
```

`systemd-cgtop`: Uso de recursos por servicio

```
Path                                                                 Tasks   %CPU   Memory  Input/s Output/s

/                                                                       92   68.8        -       0B   243.9K
/system.slice                                                            -   65.8        -        -        -
/system.slice/ModemManager.service                                       1      -        -        -        -
/system.slice/NetworkManager.service                                     2      -        -        -        -
/system.slice/accounts-daemon.service                                    1      -        -        -        -
/system.slice/apache2.service                                            3    0.1        -        -        -
/system.slice/atd.service                                                1      -        -        -        -
/system.slice/avahi-daemon.service                                       2    0.0        -        -        -
/system.slice/colord.service                                             1      -        -        -        -
/system.slice/system-postgresql.slice                                    8   66.0        -   340.4K   112.4M
/system.slice/system-postgresql.slice/postgresql@9.4-main.service        8      -        -        -        -
/system.slice/systemd-journald.service                                   1      -        -        -        -
/system.slice/systemd-logind.service                                     1    0.0        -        -        -
/user.slice                                                             13    1.6        -        -        -
/user.slice/user-1001.slice                                              -    1.6        -        -        -
/user.slice/user-1001.slice/session-2.scope                              4      -        -        -        -
/user.slice/user-1001.slice/session-4.scope                              6    1.6        -        -        -
/user.slice/user-1001.slice/session-6.scope                              5      -        -        -        -
```

Requiere habilitar `CPUAccounting`, `BlockIOAccounting`, `MemoryAccounting`

Administración de servicios con `systemctl`

- Lo que se maneja se llama una *unit*: servicios (.service), puntos de montaje (.mount), dispositivos (.device), sockets (.socket), etc.
- Comandos básicos:

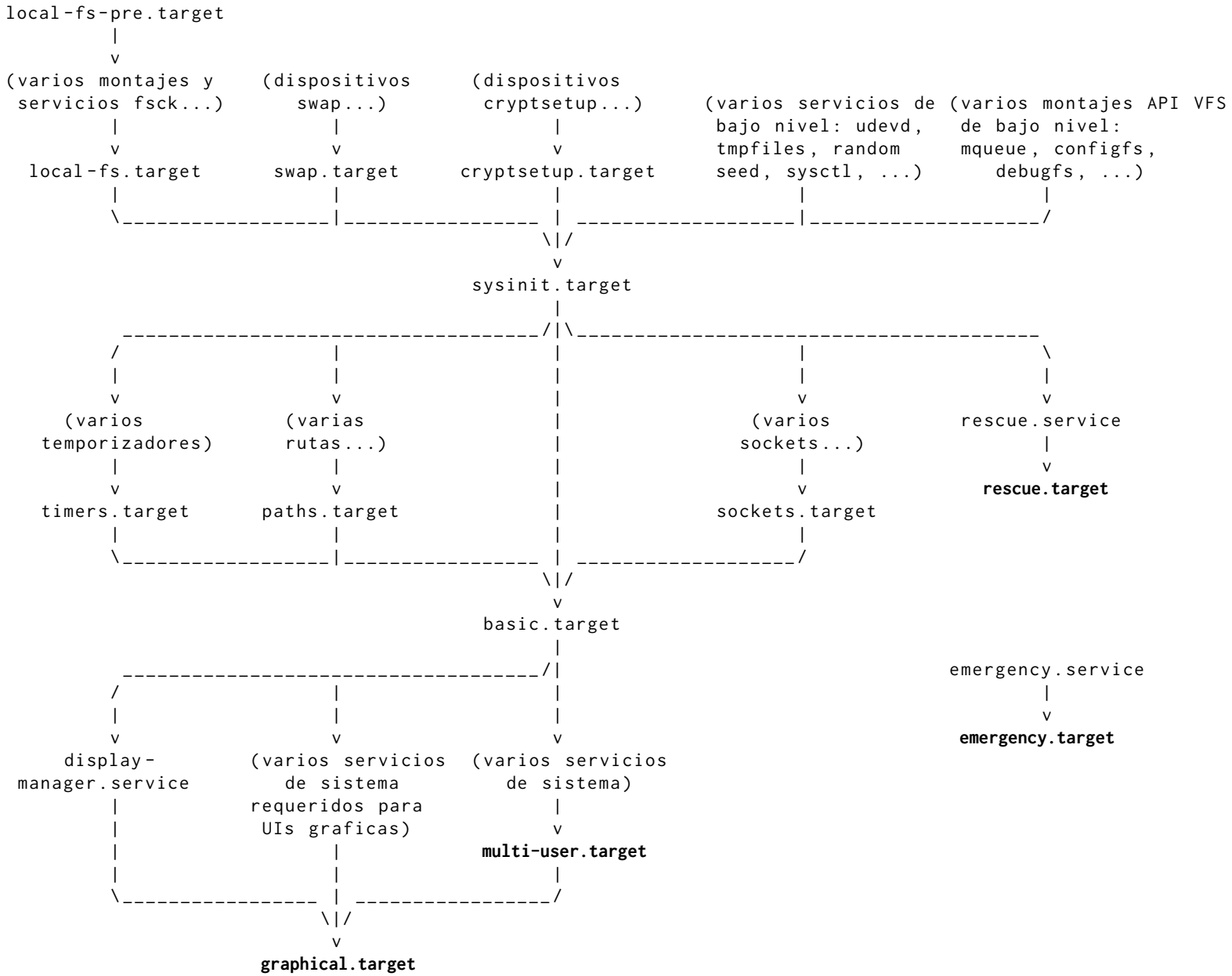
sysVinit	systemd	notas
service foo start	systemctl start foo	
service foo stop	systemctl stop foo	
service foo restart	systemctl restart foo	
service foo reload	systemctl reload foo	
service foo condrestart	systemctl condrestart foo	reinicia si ya está en ejecución
update-rc.d foo enable	systemctl enable foo	auto-inicia al siguiente arranque
update-rc.d foo disable	systemctl disable foo	deshabilita auto-start
	systemctl is-enabled foo	

- Hay auto-completado (funcionan tanto `apache2` como `apache2.service`)
- Pueden especificarse varios servicios: `systemctl restart apache2 postgresql`

systemd y los niveles de ejecución (runlevels)

- Con sysVinit, los niveles de ejecución controlan qué servicios son iniciados automáticamente
  - 0 = apagar; 1 = monousuario / mínimo; 6 = reiniciar
  - Debian: Sin diferencia entre los niveles 2, 3, 4, 5
  - RHEL: 3 = texto multi-usuario, 5 = gráfico multi-usuario
- systemd reemplaza a los runlevels con objetivos:
  - Configurados usando grupos de ligas simbólicas en `/etc/systemd/system/`**`target`**`.wants/`
  - Manipulados por `systemctl enable/disable`
  - `systemctl mask` deshabilita el servicio y evita que sea iniciado manualmente
  - El objetivo default puede configurarse con `systemctl get-default/set-default`
  - Más información: The Three Levels of "Off"

Objetivos default (bootup(7))



Analizando el tiempo de arranque

- Un arranque rápido es importante para algunos casos de uso:
  - Virtualización, nube:
    - Prácticamente sin BIOS / verificaciones de hardware $\rightsquigarrow$ sólo se inicializa el software
    - Requisito para la elasticidad de infraestructura
  - Mundo embebido
- `systemd-analyze time`: resumen

```
Startup finished in 4.883s (kernel) + 5.229s (userspace) = 10.112s
```

- `systemd-analyze blame`: Los más tardados

```
2.417s systemd-udev-settle.
2.386s postgresql@9.4-main.
1.507s apache2.service
240ms NetworkManager.servic
236ms ModemManager.service
194ms accounts-daemon.servi
```

systemd-analyze plot

- Similar a bootchartd, pero no requiere reiniciar con una línea específica `init=` para el núcleo

systemd-analyze critical-chain

- Muestra los servicios en la ruta crítica

```
graphical.target @5.226s
└─multi-user.target @5.226s
  └─exim4.service @5.144s +81ms
    └─postgresql.service @5.142s +1ms
      └─postgresql@9.4-main.service @2.755s +2.386s
        └─basic.target @2.743s
          └─timers.target @2.743s
            └─systemd-tmpfiles-clean.timer @2.743s
              └─sysinit.target @2.742s
                └─networking.service @2.589s +153ms
                  └─local-fs.target @2.587s
                    └─run-user-117.mount @3.877s
                      └─local-fs-pre.target @223ms
                        └─systemd-remount-fs.service @218ms +4ms
                          └─keyboard-setup.service @157ms +61ms
                            └─systemd-udevd.service @154ms +2ms
                              └─systemd-tmpfiles-setup-dev.service @113ms +33ms
                                └─kmod-static-nodes.service @102ms +10ms
                                  └─system.slice @96ms
                                    └─-.slice @94ms
```

Explorando el estado del sistema

- Listar unidades con `systemctl list-units` (o solo `systemctl`):
  - Unidades activas: `systemctl`
  - Mostrar sólo los servicios: `systemctl -t service`
  - Mostrar las unidades en falla: `systemctl --failed`
- Vistazo general del sistema: `systemctl status`
- GUI disponible: `systemadm`

systemctl status service

```
● avahi-daemon.service - Avahi mDNS/DNS-SD Stack
   Loaded: loaded (/lib/systemd/system/avahi-daemon.service; enabled)
   Active: active (running) since Wed 2015-04-01 21:49:28 CEST; 27s ago
 Main PID: 2858 (avahi-daemon)
   Status: "avahi-daemon 0.6.31 starting up."
   CGroup: /system.slice/avahi-daemon.service
           ├─2858 avahi-daemon: running [grep.local]
           └─2859 avahi-daemon: chroot helper

Apr 01 21:49:28 grep avahi-daemon[2858]: No service file found in /etc/avahi/services.
Apr 01 21:49:28 grep avahi-daemon[2858]: Joining mDNS multicast group on interface eth0.IPv
Apr 01 21:49:28 grep avahi-daemon[2858]: New relevant interface eth0.IPv6 for mDNS.
Apr 01 21:49:28 grep avahi-daemon[2858]: Joining mDNS multicast group on interface eth0.IPv
Apr 01 21:49:28 grep avahi-daemon[2858]: New relevant interface eth0.IPv4 for mDNS.
Apr 01 21:49:28 grep avahi-daemon[2858]: Network interface enumeration completed.
Apr 01 21:49:28 grep avahi-daemon[2858]: Registering new address record for fe80::d6be:d9f
Apr 01 21:49:28 grep avahi-daemon[2858]: Registering new address record for 152.81.5.183 on
Apr 01 21:49:28 grep avahi-daemon[2858]: Registering HINFO record with values 'X86_64'/'LIN
Apr 01 21:49:29 grep avahi-daemon[2858]: Server startup complete. Host name is grep.local.
```

Incluye:

- Nombre del servicio y su descripción, estado, PID
- Línea de estado de forma libre de `systemd-notify(1)` o `sd_notify(3)`
- El árbol de procesos dentro del cgroup
- Las últimas líneas de journald (mensajes syslog y stdout/stderr)

Configurar servicios con archivos de unidad

- Con sysVinit: scripts de shell en `/etc/init.d/`
  - Largos y difíciles de escribir
  - Código redundante entre servicios
  - Lento (muchas llamadas a `fork()`)
- Con systemd: sintaxis declarativa (tipo .desktop)
  - Mueve la inteligencia de los scripts a systemd
  - Cubre la mayor parte de las necesidades, aunque pueden usarse los scripts
  - Puede usar includes y overrides (`systemd-delta`)
  - Ver el archivo de configuración para una unidad: `systemctl cat atd.service`
  - O encontrar el archivo bajo `/lib/systemd/system/` (defaults de la distribución) o `/etc/systemd/system` (modificaciones locales)

Ejemplo simple: atd

```
[Unit]
Description=Planificador de ejecucion diferida
# Apunta a la documentación mostrada en systemctl status
Documentation=man:atd(8)

[Service]
# Comando para iniciar el servicio
ExecStart=/usr/sbin/atd -f
IgnoreSIGPIPE=false # El default es true

[Install]
# Donde "systemctl enable" crea la liga simbólica
WantedBy=multi-user.target
```

Opciones comunes

- Documentado en `systemd.unit(5)` ([Unidad]), `systemd.service(5)` ([Servicio]), `systemd.exec(5)` (ambiente de ejecución)
- Muestra todas las opciones para un servicio dado: `systemctl show atd`
- Incluir un archivo de configuración:
  ```
  EnvironmentFile=-/etc/default/ssh
  ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
  ```
- Utilizando la variable mágica `$MAINPID`:
  ```
  ExecReload=/bin/kill -HUP $MAINPID
  ```
- Auto-reiniciar un servicio cuando se cae: ($\approx$ runit / monit)
  ```
  Restart=on-failure
  ```
- Inicio condicional:
  ```
  ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
  ```
  Condicionado por arquitectura, virtualización, cmdline del núcleo,

Opciones de aislamiento y seguridad

- Utilizar un espacio de nombres de red para aislar el servicio de la red:
  `PrivateNetwork=yes`
- Utilizar un espacio de nombres por sistema de archivo:
  - Para proveer un directorio `/tmp` específico al servicio:
    `PrivateTmp=yes`
  - Para hacer algunos directorios inaccesibles o de sólo lectura:
    `InaccessibleDirectories=/home`
    `ReadOnlyDirectories=/var`
- Especifica la lista de `capabilities(7)` para un servicio:
  `CapabilityBoundingSet=CAP_CHOWN CAP_KILL`
  O únicamente elimina una:
  `CapabilityBoundingSet=~CAP_SYS_PTRACE`
- Evita que haga fork:
  `LimitNPROC=1`

Opciones de aislamiento y seguridad (2)

- Ejecutar como usuario/grupo: `User=`, `Group=`
- Ejecutar dentro de un chroot:

```
RootDirectory=/srv/chroot/foobar
ExecStartPre=/usr/local/bin/setup-foobar-chroot.sh
ExecStart=/usr/bin/foobard
RootDirectoryStartOnly=yes
```

- Controlar porción de CPU, límites de memoria, E/S de bloques, nivel de swap:

```
CPUShares=1500
MemoryLimit=1G
BlockIOWeight=500
BlockIOReadBandwith=/var/log 5M
ControlGroupAttribute=memory.swappiness 70
```

- Más información: Converting sysV init scripts to systemd service files, Securing your services, Changing roots, Managing resources

Unidades de temporizador

- Similar a cron, pero con todo el poder de systemd (dependencias, ejecución de la configuración de ambiente, etc.
- Temporizadores de tiempo real (wallclock): expresiones de eventos de calendario
  - Expresados utilizando un formato complejo (ver `systemd.time(7)`), con patrones sobre estampas de tiempo como: `Vie 2012-11-23 11:12:13`
  - Ejemplos de valores válidos: `hourly` (= `*-*-* *:00:00`), `daily` (= `*-*-* 00:00:00`), `*:2/3` (= `*-*-* *:02/3:00`)
- Temporizadores monotónicos, relativos a distintos puntos de inicio:
  - 5:30h después del arranque del sistema: `OnBootSec=5h 30m`
  - 50s después del inicio de systemd: `OnstartupSec=50s`
  - 1 hora después de que la unidad fue activada por última vez: `OnUnitActiveSec=1h` (puede combinarse con `OnBootSec` o

Ejemplo de unidades de temporizador

- myscript.service:

```
[Unit]
Description=MyScript

[Service]
Type=simple
ExecStart=/usr/local/bin/myscript
```

- myscript.timer:

```
[Unit]
Description=Ejecuta myscript cada hora
[Timer]
# Tiempo a esperar despues del arranque antes
# de la primera ejecucion
OnBootSec=10min
# Tiempo entre cada ejecucion consecutiva
OnUnitActiveSec=1h
Unit=myscript.service
[Install]
WantedBy=multi-user.target
```

Ejemplo de unidades de temporizador (2)

- Iniciar el temporizador:
  `systemctl start myscript.timer`
- Activar que el temporizador inicie al arrancar:
  `systemctl enable myscript.timer`
- Listar todos los temporizadores:
  `systemctl list-timers`

Activación por sockets

- systemd espera conexiones a nombre del servicio hasta que éste esté listo, y entonces le transfiere las conexiones pendientes
- Beneficios:
  - No es necesario expresar el ordenamiento para el arranque
    - Pueden iniciarse en paralelo $\rightsquigarrow$ arranque rápido
    - Esperarán unos a otros cuando sea necesario (cuando tengan que comunicarse entre sí) gracias a la activación por sockets
  - Los servicios que son requeridos esporádicamente no tienen que mantenerse ejecutando, y pueden ser inicializados sobre demanda
- No limitado a servicios de red: También activación de D-Bus y de rutas
- Más información: Converting inetd Service, Socket Activation for developers (+ follow-up)

Ejemplo de activación por socket: dovecot

dovecot.socket:

```
[Unit]
Description=Socket de activ. \
 del servidor de correo \
 IMAP/POP3 Dovecot

[Socket]
# dovecot espera sockets
# distintos IPv4 e IPv6
BindIPv6Only=ipv6-only
ListenStream=0.0.0.0:143
ListenStream=[::]:143
ListenStream=0.0.0.0:993
ListenStream=[::]:993
KeepAlive=true

[Install]
WantedBy=sockets.target
```

dovecot.service:

```
[Unit]
Description=Servidor de correo  \
 IMAP/POP3 Dovecot
After=local-fs.target network.target

[Service]
Type=simple
ExecStart=/usr/sbin/dovecot -F
NonBlocking=yes

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

Ejemplo de activación por socket: sshd

- sshd.socket:

```
[Unit]
Description=Servidor SSH de socket por conexion

[Socket]
ListenStream=22
Accept=yes

[Install]
WantedBy=sockets.target
```

- sshd@.service:

```
[Unit]
Description=Servicio por conexion SSH

[Service]
ExecStart=-/usr/sbin/sshd -i
StandardInput=socket
```

Ejemplo de activacion por socket: sshd (2)

- `sshd@.service` significa que es un servicio instanciado
- Hay una instancia de `sshd@.service` por conexión:

```
# systemctl --full | grep ssh
sshd@172.31.0.52:22-172.31.0.4:47779.service  loaded active running
sshd@172.31.0.52:22-172.31.0.54:52985.service loaded active running
sshd.socket                                   loaded active listening
```

- Los servicios instanciados son también utilizados por getty
  - Ver Serial console e Instanciated services

Bitácoras con journald

- Componente de systemd
- Captura mensajes de syslog, el núcleo, initrd, arranque temprano, mensajes escritos a stdout/stderr por todos los servicios
  - Reenvía todo a syslog
- Formato estructurado (campos llave/valor), puede contener datos arbitrarios
  - Pueden ser vistos en un formato tipo syslog con `journalctl`
- Indexado, bitácora binaria; rotación manejada transparentemente
- Puede reemplazar a syslog (o también trabajar en paralelo)
- No persistente entre reinicios por default – Para hacerlo persistente, crea el directorio `/var/log/journal`, con:

```
install -d -g systemd-journal /var/log/journal
setfacl -R -nm g:adm:rx,d:g:adm:rx /var/log/journal
```

- Puede registrar en un servidor remoto (con `systemd-journal-gateway`, aún no en Debian)

Entrada ejemplo de bitácora

```
_SERVICE=systemd-logind.service
MESSAGE=User harald logged in
MESSAGE_ID=422bc3d271414bc8bc9570f222f24a9
_EXE=/lib/systemd/systemd-logind
_COMM=systemd-logind
_CMDLINE=/lib/systemd/systemd-logind
_PID=4711
_UID=0
_GID=0
_SYSTEMD_CGROUP=/system/systemd-logind.service
_CGROUPS=cpu:/system/systemd-logind.service
PRIORITY=6
_BOOT_ID=422bc3d271414bc8bc95870f222f24a9
_MACHINE_ID=c686f3b205dd48e0b43ceb6eda479721
_HOSTNAME=waldi
LOGIN_USER=500
```

Utilizando `journalctl`

- Ver la bitácora completa: `journalctl`
- Desde el último arranque: `journalctl -b`
- Para un intervalo de tiempo dado: `journalctl --since=yesterday` or `journalctl --until="2013-03-15 13:10:30"`
- Verlo en el formato verboso (nativo): `journalctl -o verbose`
- Filtrar por unidad de systemd: `journalctl -u ssh`
- Filtrar por el campo desde el formato verboso:
  `journalctl _SYSTEMD_UNIT=ssh.service`
  `journalctl _PID=810`
- Seguimiento en línea (≈ tail -f): `journalctl -f`
- Últimas entradas (≈ tail): `journalctl -n`
- Funciona con bash-completion
- Ver también: Journald design document, Using the Journal

Integración con contenedores

- Filosofía general: Integrar la administración de servicios (VMs y contenedores) con aquellos del anfitrión
  - `systemd-machined`: da seguimiento a máquinas, proporciona un API para listar, crear, registrar, matar y terminar máquinas, transferir imágenes (tar, raw, Docker)
  - `machinectl`: Herramienta CLI para manipular máquinas
  - otras herramientas que tienen soporte a contenedores:
    - `systemctl -M contenedor restart foo`
    - `systemctl list-machines`: Estado de los contenedores
    - `journalctl -M contenedor`
    - `journalctl -m`: Bitácora combinada de todos los contenedores
- systemd tiene un mini-gestor de contenedores: `systemd-nspawn`
- Otras soluciones de virtualización pueden hablar con `machined`
- Más información: Container integration

Redes en systemd systemd-networkd

- Reemplazo para `/etc/network/interfaces`, en servidores y VMs
  - No es en realidad para Network Manager en desktops o laptops
- Permite establecer la configuración IP, configurando puentes, vlans, bonding, túneles, etc.
- Los archivos de configuración con una sección `[Match]` se aplican para una dirección MAC, driver, ruta udev, tipo, hostname, etc.
  - foo.link: configuración a nivel enlace – Direcciones MAC, nombre de interfaz, MTU, límites, dúplex, Wake on LAN
  - foo.netdev: creación de dispositivos virtuales de red (puentes, uniones, vlans, túneles IPIP o GRE, VXLAN, tun, tap, veth)
  - foo.network: configuración de dispositivos de red: IP (estática o DHCP, gateway, rutas adicionales, DNS), puentes
- Más información: systemd-networkd(8), systemd.link(5), systemd.network(5), systemd.netdev(5)

Ejemplo 1: DHCP, ruta adicional

- Para mejor rendimiento, systemd incluye un cliente DHCP

```
# /etc/systemd/network/ethernet.network
[Match]
Name=eth0

[Network]
DHCP=yes

[Route]
Gateway=192.168.1.253
Destination=10.0.0.0/8
```

Ejemplo 2: direccionamiento estático y VLAN

```
# /etc/systemd/network/vlan1.netdev
# [Match] es opcional en los archivos netdev
[NetDev]
Name=vlan1
Kind=vlan

[VLAN]
Id=1

# /etc/systemd/network/ethernet.network
[Match]
Name=eth0

[Network]
DHCP=yes
VLAN=vlan1 # crea vlan1 en este dispositivo

# /etc/systemd/network/vlan1.network
[Match]
Name=vlan1

[Network]
Address=192.168.1.1/24
Gateway=192.168.1.254
```

Ejemplo 3: puente y tap

```
# /etc/systemd/network/bridge0.netdev
[NetDev]
Name=bridge0
Kind=bridge

# /etc/systemd/network/bridge0.network
[Match]
Name=bridge0

[Network]
Address=192.168.1.1/24
DHCPServer=yes # systemd tiene su propio servidor DHCP, muy basico

# /etc/systemd/network/tap.netdev
[NetDev]
Name=tap0
Kind=tap

# /etc/systemd/network/tap.network
[Match]
Name=bridge0

[NetDev]
Bridge=bridge0
```

Migración desde sysvinit

- systemd provee "ganchos" hacia los scripts de inicio LSB: `service foo start|stop|...` y `/etc/init.d/foo` redirigen a `systemctl`
- `systemd-sysv-generator` crea unidades que envuelven las llamadas a los scripts LSB:
  - Usando dependencias LSB
  - Los servicios son descritos como *LSB: foo*
    - Listar todos los servicios generados:
      `systemctl list-timers | grep LSB:`

Archivo genérico autogenerado para apache2

```
$ systemctl cat apache2.service
# /run/systemd/generator.late/apache2.service
# Automatically generated by systemd-sysv-generator

[Unit]
Description=LSB: Apache2 web server
Before=runlevel2.target runlevel3.target runlevel4.target
  runlevel5.target shutdown.target
After=local-fs.target remote-fs.target network-online.target
  systemd-journald-dev-log.socket nss-lookup.target
Wants=network-online.target
Conflicts=shutdown.target

[Service]
Type=forking
KillMode=process
[...]
ExecStart=/etc/init.d/apache2 start
ExecStop=/etc/init.d/apache2 stop
ExecReload=/etc/init.d/apache2 reload
```

Más cosas

- Nuevos archivos de configuración inter-distribuciones: `/etc/hostname`, `/etc/locale.conf`, `/etc/sysctl.d/*.conf`, `/etc/tmpfiles.d/*.conf`
- Herramientas para manejar el nombre de host, configuraciones locales, fecha y hora: `hostnamectl`, `localectl`, `timedatectl`
- Soporte para watchdogs
- Manejo de sesiones de usuario
  - Cada usuario en su cgroup
  - Soporte a multi-asiento
  - `loginctl` para manejar sesiones, usuarios, asientos

Conclusiones

- systemd redefine la forma en que administramos sistemas Linux
  - *Si rediseñamos la gestión de servicios desde cero, ¿se vería como systemd?*
- Para los desarrolladores de servicio: Es más fácil dar soporte a systemd que a sysVinit
  - No hace falta hacer fork, soltar privilegios, manejar un archivo donde normalmente irá nuestro PID
  - Sólo envía la salida a stdout (redirigido a syslog, con manejo de prioridades)
- Algunas partes presentan todavía dificultades, o son objetivos móviles, pero son prometedores: Bitácora, respaldos, red
- systemd puede no ser la respuesta definitiva, pero es un punto de datos interesante para evaluar