

System-D

Alex Mosquera Gundin, Samuel Varela Camba, Angel Berreco Kozlan

April 12, 2025

Contents

1	Introducción a SystemD	3
2	Instalación y configuración de Systemd	3
2.0.1	Instalacion de Systemd con la ayuda de un gestor de paquetes	3
2.0.2	Instalacion de Systemd compilandolo a mano	3
2.1	Configuración de Systemd	4
3	Comparativa con otros sistemas de inicio	5
3.1	Diferencias entre Systemd y SysVinit	5
3.2	Ventajas y desventajas de Systemd	6
4	Configuraciones del sistema	6
4.1	Configuraciones del sistema con Systemd	6
5	Unidades y gestion de servicios	8
5.1	Unidades en systemd	8
5.1.1	Características principales de los Units en systemd	9
5.1.2	Cambios de compatibilidad	9
5.2	Gestión y creación de Units en systemd	10
5.2.1	¿Como crear unidades en systemd?	10
5.2.2	Creacion de un .service en systemd	11
5.2.3	Creacion de un .target en systemd	11
5.2.4	Creacion de un .mount en systemd	12
5.2.5	Creacion de un .timer en systemd	12
5.2.6	Creacion de un .socket en systemd	12
5.2.7	Creacion de un .slice en systemd	13
5.2.8	Creacion de un .network en systemd	13
6	Funcionamiento y Arranque del Sistema	13
6.1	Arranque del sistema	13
6.1.1	Encendido y ejecucion del firmware	14
6.1.2	Cargador de arranque	14
6.1.3	Carga del kernel del sistema	14
6.1.4	El Proceso de inicio (PID 1)	14
6.1.5	Sesion de Usuario y Login	15
7	Registro y Monitoreo	15
7.1	Gestión de logs con journalctl	15
8	Componentes Avanzados de SystemD	18
8.1	systemd-boot	18
8.2	systemd-nspawn	18
8.3	systemd-slice	19
8.4	systemd-network	20
8.4.1	Creacion de bridge	21
8.4.2	Creacion de una VLAN	21
8.4.3	Configuracion de propiedades fisicas	21
8.4.4	Comandos de gestión	22
8.5	systemd-resolved	22
8.6	systemd-analyze	23
8.7	systemd-logind	23
8.8	systemd-creds	24
9	Conversión de un .service a SysVinit	25

1 Introducción a SystemD

SystemD es el gestor de arranque y servicios moderno para Linux, concebido como una solución integral para la administración del sistema. Se trata de un conjunto integrado de daemons, bibliotecas y utilidades que actúan como una plataforma central para la configuración y el control del sistema operativo GNU/Linux. Desarrollado principalmente por Lennart Poettering en 2010, a partir de 2014, systemd se ha convertido en el sistema de inicio predeterminado en la mayoría de las distribuciones modernas, reemplazando los métodos tradicionales heredados de Unix. Aunque en principio fue desarrollado para la gestión del arranque como una alternativa a los sistemas Init, la adopción de más funcionalidades ha llevado a que se convierta en una herramienta recargada de muchas tareas en el sistema, rompiendo la filosofía tradicional de Unix de "hacer solo una cosa y hacerla bien". Esto ha generado tanto elogios como críticas, que han provocado que muchos usuarios de la comunidad, al ver que sus sistemas migran de SysVinit a SystemD, hayan desarrollado clones de las distribuciones. Un ejemplo de esto es Devuan, que se basa en Debian pero utiliza SysVinit como sistema de inicio.

2 Instalación y configuración de Systemd

Tenemos dos formas de instalar Systemd: compilándolo a mano, o con la ayuda de un gestor de paquetes.

2.0.1 Instalación de Systemd con la ayuda de un gestor de paquetes

Antes de nada, tenemos que verificar si tenemos Systemd instalado, para ello podemos usar la herramienta que nos encuentra el id de un proceso corriendo:

```
# pidof systemd, o tambien # systemctl --version
```

Una vez comprobado que systemd no está instalado, vamos a mostrar un ejemplo de instalación para sistemas debian y derivados usando un gestor de paquetes.

```
1 $ sudo apt-get update && sudo apt-get install systemd systemd-sysv
```

Una vez instalado el paquete, ahora debemos habilitar systemd como sistema de arranque por defecto.

```
1 $ sudo systemctl set-default multi-user.target # si queremos entorno grafico, si no
2 # podemos usar otros targets para cubrir otras necesidades
3
4 #si ejecutamos: sudo systemctl isolate multi-user.target #cambiamos de forma inmediata
5 al target especificado.
6 $ sudo reboot
```

Después de reiniciar, verificamos si systemd es el proceso de inicio ejecutado con

```
1 $ ps -p 1
```

Nos deberá salir una salida con el proceso de systemd.

2.0.2 Instalación de Systemd compilándolo a mano

Este ejemplo de instalación a partir del código fuente lo hemos obtenido del libro 'Linux from Scratch Version Systemd'. Partiendo de la base de que ya tenemos el código base descargado y extraído en nuestro sistema, primero vamos a preparar el entorno de instalación con el siguiente comando

```
1 $ mkdir -pv /etc/systemd/system
```

Esto nos crea el directorio de configuraciones específicas del sistema. Ahora vamos a compilar systemd con

```
1 $ ./configure --prefix=/usr \
2               --sysconfdir=/etc \
3               --localstatedir=/var \
4               --libexecdir=/usr/lib \
5               --bindir=/usr/bin \
6               --sbindir=/usr/sbin \
7               --docdir=/usr/share/doc/systemd-<version>
8
9 # una vez finalizado
10 $ make
11 $ make install
```

donde <version> es la versión específica del paquete systemd que estamos instalando.

Ahora crearemos el enlace simbólico para que el sistema reconozca systemd como su init system

```
1 $ ln -sf /lib/systemd/systemd /sbin/init
```

2.1 Configuración de Systemd

Una vez instalado, tenemos que asegurarnos de que el sistema tenga todos los archivos de unidades necesarios para arrancar. Un ejemplo de configuración básica para el servicio de usuario agetty en la terminal de control tty1 es:

```
1 $ cat > /etc/systemd/system/getty@tty1.service <<< EOF
2 [Unit]
3 Descripcion=Getty on tty1
4 After=systemd-user-sessions.service plymouth-quit-wait.service
5 # systemd-user-sessions.service es el servicio que gestiona sesiones de usuario en
6   systemd
7 # plymouth-quit-wait es el servicio relacionado con la pantalla de inicio (splash screen
8   ) plymouth.
9 [Service]
10 ExecStart=-/sbin/agetty --noclear %I 38400 linux
11 # El menos antes del comando indica que si el programa falla, systemd no tratara el
12   error como critico.
13 # /sbin/getty es el programa que maneja los inicios de sesion en terminales virtuales.
14 # --noclear evita que la pantalla se borre antes de mostrar el prompt de login.
15 # %I: variable que se ve reemplazada por tty, ya que es el servicio getty@tty1.
16 # 38400 es la velocidad del puerto serie (no relevante en terminales virtuales)
17 # linux: indica el tipo de terminal virtual que se esta ejecutando.
18 Type=idle
19 Restart=always
20 [Install]
21 WantedBy=getty.target
22 EOF
23
24 # Ahora habilitaremos el servicio para que se active automaticamente
25 $ systemctl enable getty@tty1.service
26
27 # Configuramos el target por defecto
28 # si queremos modo multiusuario
29 $ systemctl set-default multi-user.target
30
31 # si queremos interfaz grafica
32 $ systemctl set-default graphical.target
33
34 # para habilitar la persistencia de los logs en disco:
35 $ mkdir -p /var/log/journal
```

```
$ systemctl restart systemd-journald  
$ reboot
```

3 Comparativa con otros sistemas de inicio

Systemd no es el único sistema de inicio existente, podemos destacar algunos de sus predecesores más antiguos como SysVinit, Upstart, OpenRc, Runit y ls6. Upstart fue desarrollado por Ubuntu para mejorar SysVinit, tiene soporte para eventos y ejecución paralela. OpenRc está basado en scripts, pero es más eficiente y con mejor gestión de dependencias que SysVinit. Runit tiene una estructura de directorios en `/etc/runit` en lugar de scripts largos. S6 es similar a Runit pero con características más avanzadas. SysVinit fue el sistema tradicional en la mayoría de distribuciones Linux antes de la llegada de Systemd y por tanto, es el único en el que nos vamos a centrar para compararlo con este último.

3.1 Diferencias entre Systemd y SysVinit

1. Paralelización del arranque: Systemd carga servicios en paralelo simultáneamente en función de las dependencias establecidas para reducir tiempos de inicio, SysVinit lo hace de manera secuencial donde los servicios se inician uno tras otros según los run levels.
2. Gestión centralizada: Systemd tiene `systemctl` para controlar servicios, unidades y registros. SysVinit tiene scripts.
3. Activación bajo demanda: A diferencia de SysVinit que se controla bajo scripts, Systemd puede iniciar servicios cuando se necesiten, mediante el uso de sockets y timers.
4. Gestión avanzada de logs: Systemd tiene `journalctl` para manejar registros detallados que crea `journald`. SysVinit carece de ningún tipo de sistema de gestión de logs y necesita de uno externo como puede ser `syslog` o `rsyslog`.
5. Gestión de las dependencias: En Systemd las dependencias son declarativas, es decir, se especifican en el archivo de configuración del servicio de forma explícita. En SysVinit están basadas en scripts y en números de secuencia. Las dependencias declarativas de Systemd definen relaciones entre servicios, lo que es una gestión mucho más avanzada que la de SysVinit (que es bastante limitada).
6. Soporte de cgroups: mientras que Systemd tiene un control total de los cgroups (funcionalidad del kernel de linux que permite agrupar procesos y controlar los recursos del sistema que cada grupo puede usar), SysVinit no tiene integración nativa con estos.
7. Compatibilidad: al ser SysVinit el estándar durante décadas, este es más compatible con numerosos scripts y herramientas. Por otro lado, aunque Systemd es compatible con scripts, SysVinit y LSB, su adopción ha generado debates en la comunidad debido a su naturaleza más compleja y su integración profunda en el sistema.
8. Archivos de configuración: SysVinit usa scripts de shell ubicados en `/etc/init.d/` y enlaces simbólicos en `/etc/rc.d*/`, mientras que systemd usa archivos de unidad (`.service`, `.socket`, `.timer`, `.target`, etc ...) ubicados en `/etc/systemd/system/` y en `/usr/lib/systemd/system/`.
9. Monitorización de servicios: SysVinit tiene una monitorización básica ya que carece de mecanismos integrados para reiniciar servicios fallidos automáticamente, mientras que systemd tiene una monitorización de servicios avanzada que permite reiniciar automáticamente servicios que fallan y monitorizar su estado en tiempo real.
10. Centralización excesiva: A diferencia de SysVinit que solo se encarga del arranque, Systemd se encarga a mayores de:
 - ★ Los registros del sistema.
 - ★ La gestión de las interfaces de red y conexiones de red (interfaces ethernet y wifi, conexiones wifi y ethernet, etc ...).
 - ★ Tiene un arrancador llamado `systemd-boot` como alternativa al grub.
 - ★ Se encarga también con `systemd-resolved` de los DNS del sistema

- ★ Con systemd-tmpfiles se encarga de la gestión de los archivos temporales del sistema.
- ★ Con systemd-timesyncd se encarga de la sincronización de la hora del sistema.
- ★ Se encarga de la gestión de energía del sistema con systemd-[poweroff, reboot, suspend, halt, hibernate].
- ★ Se encarga del login con systemd-logind.
- ★ Se encarga del nombre de la máquina con systemd-hostnamed.

3.2 Ventajas y desventajas de Systemd

- Ventajas:
 1. Inicio más rápido del sistema.
 2. Registro unificado con journalctl (si es que se puede ver como una ventaja).
 3. Control granular de servicios y recursos.
 4. Gestión eficiente con cgroups.
- Desventajas:
 1. Mayor complejidad en la configuración.
 2. Puede consumir más recursos en comparación con SysVinit.
 3. Centralización excesiva que rompe con la filosofía tradicional Unix. Por ejemplo, en una máquina Arch Linux con systemd, se han contado un total de 110 services units de systemd.

4 Configuraciones del sistema

4.1 Configuraciones del sistema con Systemd

A continuación, se presentan algunos comandos esenciales para la gestión de servicios con ‘systemd’:

- Iniciar un servicio:

```
systemctl start nombre-servicio.service
```

- Detener un servicio:

```
systemctl stop nombre-servicio.service
```

- Ver el estado de un servicio:

```
systemctl status nombre-servicio.service
```

- Reiniciar un servicio:

```
systemctl restart nombre-servicio.service
```

- Habilitar un servicio para que arranque con el sistema:

```
systemctl enable nombre-servicio.service
```

- Deshabilitar un servicio para que no arranque automáticamente:

```
systemctl disable nombre-servicio.service
```

- Recargar la configuración de un servicio sin reiniciarlo:

```
1 systemctl reload nombre-servicio.service
```

- Enmascarar un servicio para evitar su inicio manual o automático (que lo inicieº otro servicio):

```
1 systemctl mask nombre-servicio.service
```

- Desenmascarar un servicio:

```
1 systemctl unmask nombre-servicio.service
```

- Listar objetivos disponibles:

```
1 systemctl list-units --type=target
```

- Ver los logs del último arranque:

```
1 journalctl -b
```

- Ver logs en tiempo real:

```
1 journalctl -f
```

- Ver logs de un servicio específico:

```
1 journalctl -u nombre-servicio.service
```

- Configurar el sistema para arrancar en modo multiusuario (sin entorno gráfico):

```
1 systemctl set-default multi-user.target
```

- Configurar el sistema para arrancar en modo gráfico:

```
1 systemctl set-default graphical.target
```

- Cambiar de forma inmediata el target en modo gráfico:

```
1 systemctl isolate graphical.target
```

- Configurar el identificador de la máquina:

```
1 #Se encuentra en /etc/machine-id
2 systemd-machine-id-setup
```

- Cambiar el nombre de la máquina (hostname):

```
1 #El fichero es /etc/hostname
2 #El daemon es systemd-hostnamed
3 hostnamectl set-hostname <nombre>
```

- Configurar el idioma y el teclado de la máquina:

```
1 #Archivo se encuentra en /etc/locale.conf
2 #El daemon es systemd-localed
3 localectl set-locale <LOCALE> #Para establecer determinado locale
4 localectl list-locales # Para listar los locales disponibles en el sistema
5 localectl #Para ver la configuracion actual
```

- Configurar la fecha y hora de la máquina:

```
1  #El daemon es systemd-timedated
2  timedatectl set-timezone <timeZone> #Para establecer determinada zona horaria
3  timedatectl set-ntp <{true,false}> #activar o no el ntp
4  timedatectl set-time "YYYY-MM-DD HH:MM:SS" #Para cambiar hora y fecha manualmente
```

- Para cargar módulos del kernel:

```
1  #En /etc/modules-load.d/ tenemos archivos de configuracion (*.conf) para indicar
   #que modulos del kernel debemos cargar automaticamente con el arranque del
   #sistema
2  #Esta gestionado por systemd-modules-load.service
```

- Ver el archivo de una unit de systemd:

```
1  systemctl cat <unit>
2  systemctl cat NetworkManager.service # por ejemplo
```

- Ver las dependencias de una unit de systemd:

```
1  systemctl list-dependencies <unit>
2  systemctl list-dependencies NetworkManager.service # por ejemplo
```

- Ver el objetivo actual del sistema:

```
1  systemctl get-default
```

5 Unidades y gestion de servicios

5.1 Unidades en systemd

Systemd usa unidades para administrar diferentes aspectos del sistema. Los tipos de unidades más importantes que tenemos en systemd son:

1. **.service**: servicios, definen y controlan servicios en el sistema, como servidores web, bases de datos, scripts personalizados, el daemon de las conexiones a internet, etc.
2. **.target**: agrupaciones de unidades. Los targets son grupos de unidades que permiten cambiar el estado del sistema (modo gráfico, modo texto, modo recuperación, etc). Los targets tienen un conjunto de services que se arrancan y podemos decir que, una vez arrancados dichos servicios, el sistema está en un estado de los definidos anteriormente.
3. **.mount**: puntos de montaje, se pueden definir puntos de montaje que se montan automáticamente al arrancar el sistema.
4. **.automount**: punto de montaje automático del sistema de ficheros. Es necesario tener un **.mount** asociado, permite el montaje bajo demanda, es decir, solo cuando se accede al punto de montaje, y no se monta automáticamente al arrancar el sistema.
5. **.timer**: tareas programadas. Este reemplazo de cron permite programar la ejecución de servicios en momentos específicos.
6. **.socket**: comunicación basada en sockets. Las unidades **.socket** configuran servicios que se inician automáticamente cuando reciben una conexión en un socket, similar a inet.d.
7. **.device**: archivo de dispositivo reconocido por el kernel, discos duros o periféricos USB.
8. **.path**: archivo o directorio de un sistema de ficheros, se utiliza para monitorizar cambios en rutas de archivos.

9. `.scope`: representa procesos iniciados externamente a `systemd` pero que éste gestiona y supervisa en términos de recursos y límites, es decir, gestiona el porcentaje de recursos propios del sistema que este proceso puede llegar a ocupar.
10. `.slice`: grupo jerárquico para organizar procesos y unidades, permitiendo asignar recursos mediante `cgroups`.
11. `.swap`: administra archivos o particiones como espacio de intercambio (swap) para la memoria virtual del sistema operativo.
12. `.network`: unidad para gestionar configuraciones de red.
13. `.netdev`: unidad para definir dispositivos de red.

5.1.1 Características principales de los Units en systemd

Systemd y el gestor de servicios proporcionan las siguientes características principales:

- ★ **Socket-based activation**: Al arrancar el sistema, `systemd` se encarga de crear sockets de escucha para todos los servicios que soportan este tipo de activación y pasa los sockets a estos servicios tan pronto como se inician. Esto no solo permite que `systemd` pueda iniciar servicios en paralelo, si no que también hace posible reiniciar un servicio sin perder ningún mensaje que se le ha sido enviado mientras no estaba disponible: el socket correspondiente sigue siendo accesible y todos los mensajes se ponen en una cola. `Systemd` utiliza `socket-units` para la activación basada en sockets.
- ★ **Bus-based activation**: los servicios del sistema que utilizan D-Bus para la comunicación entre procesos pueden iniciarse bajo demanda la primera vez que una aplicación cliente intenta comunicarse con ellos. `Systemd` utiliza `dbus-service-files` para la activación basada en dbus.
- ★ **Device-based activation**: los servicios del sistema que admiten la activación basada en dispositivos pueden iniciarse a petición cuando se conecta un tipo concreto de hardware o esta disponible. `Systemd` usa `device-units` para la activación basada en dispositivos.
- ★ **Path-based activation**: los servicios del sistema que soportan la activación basada en la ruta pueden iniciarse bajo demanda cuando un archivo o directorio concreto cambia de estado. `Systemd` utiliza `path-units` para la activación basada en la ruta.
- ★ **Mount & automount point management**: `systemd` controla y gestiona los puntos de montaje y automontaje. `Systemd` utiliza `mount-units` para los puntos de montaje y `automount-units` para los puntos de automontaje.
- ★ **Aggressive parallelization**: debido al uso de la activación basada en sockets, `systemd` puede iniciar los servicios del sistema en paralelo tan pronto como todos los sockets de escucha están en su lugar, en combinación con los servicios del sistema que soportan la activación bajo demanda, la activación en paralelo reduce significativamente el tiempo necesario para arrancar el sistema.
- ★ **Transactional unit activation logic**: Antes de activar o desactivar una unidad, `systemd` calcula sus dependencias, crea una transacción temporal y verifica que esta transacción sea consistente. Si esta es inconsistente, `systemd` intenta automáticamente corregirla y eliminar de ella los trabajos no esenciales antes de informar de un error.
- ★ **Backwards compatibility with SysVinit-systemd**: soporta los scripts de SysVinit como se describe en el Linux Standard Base Core Especification, lo que facilita la ruta de actualización de las unidades de servicio de `systemd`.

5.1.2 Cambios de compatibilidad

`Systemd` y el gestor de servicios están diseñados para ser mayoritariamente compatibles con SysVinit y Upstart, los siguientes son los cambios de compatibilidad más notables al sistema Red Hat Enterprise Linux 6 que utilizaba SysVinit:

- ★ Systemd solo tiene un soporte limitado para los niveles de ejecución. Proporciona una serie de unidades de destino que se pueden asignar directamente a estos niveles de ejecución y, por razones de compatibilidad también se distribuye con el comando `runlevel`, sin embargo, no todos los objetivos de systemd pueden ser asignados directamente a niveles de ejecución, y como consecuencia, este comando puede devolver N para indicar un nivel de ejecución desconocido. Se recomienda evitar el uso del comando `runlevel` si es posible.
- ★ La utilidad `systemctl` no admite comandos personalizados, además de los comandos estándar como `start`, `stop`, y `status` los autores de los scripts de SysVinit podrían implementar soporte para cualquier número de comandos arbitrarios con el fin de proporcionar funcionalidad adicional. Por ejemplo, el script de `init` para `iptables` podría ser ejecutado con el comando `panic`, que inmediatamente activaría el `panic mode`, y reconfiguraría el sistema para comenzar a dejar caer todos los paquetes entrantes y salientes. Esto no está soportado en `systemd` y `systemctl` solo acepta comandos documentados.
- ★ La utilidad `systemctl` no se comunica con los servicios que han sido iniciados a través de la línea de comandos. Cuando `systemd` inicia un servicio del sistema almacena el id de proceso para poder seguirlo. La utilidad `systemctl` utiliza entonces ese id (PID) para consultar y gestionar ese servicio. En consecuencia, si un usuario inicia un daemon en concreto por línea de comandos, `systemctl` no puede detectar su estado actual ni detenerlo.
- ★ Los servicios del sistema no pueden leer del flujo de entrada estándar (`stdin`). Cuando `systemd` inicia un servicio, conecta su entrada estándar a `/dev/null` para evitar cualquier interacción con el usuario. Los servicios del sistema no heredan ningún contexto (como las variables de entorno `HOME` y `PATH`) del usuario que los invoca y su sesión. Cada servicio se ejecuta en un contexto de ejecución limpio.
- ★ Cuando se carga un script de SysVinit, `systemd` lee la información de dependencia codificada en la cabecera Linux Standard Base (LSB) y la interpreta en tiempo de ejecución.
- ★ Todas las operaciones en unidades de servicio están sujetas a un tiempo de espera por defecto de 5 minutos para evitar que un servicio que no funcione congele el sistema. Este valor está codificado para los servicios que se generan a partir de los `init` scripts y no se puede cambiar. Sin embargo, se pueden utilizar archivos de configuración individuales para especificar un valor de tiempo de espera más largo por servicio.

5.2 Gestión y creación de Units en systemd

Comandos básicos para la gestión de units:

```

1  # recargar unidades
2  $ sudo systemctl daemon-reload
3
4  # recarga unidad a nivel de usuario
5  $ systemctl --user daemon-reload
6
7  # correr o parar servicios en el contexto del usuario
8  $ systemctl --user [start | stop] nombre-servicio.service

```

5.2.1 ¿Como crear unidades en systemd?

Primero vamos a mirar donde se almacenan las unidades, esta ruta dependerá de su propósito y origen; tenemos las siguientes rutas:

- `/etc/systemd/system`: este directorio está reservado para los archivos de unidad creados y gestionados por el administrador del sistema.
- `/lib/systemd/system`: este directorio contiene los archivos de unidad instalados por los paquetes del sistema, no se recomienda modificar archivos en este directorio ya que pueden ser sobrescritos durante las actualizaciones del sistema.
- `/run/systemd/system`: este directorio se utiliza para archivos de unidad creados en tiempo de ejecución. Los archivos de esta ruta son temporales y se eliminan al apagar el sistema.

- `$HOME/.config/systemd/usr/`: este directorio es donde los usuarios pueden configurar sus archivos de unidad de systemd personalizados, es útil para servicios que no requieren permisos de administrador y que deben ejecutarse en el contexto del usuario.

Vamos a realizar el ejemplo de crear cada tipo de unit en systemd en `/etc/systemd/system`:

5.2.2 Creacion de un .service en systemd

Dentro del directorio `/etc/systemd/system` definimos un archivo con extensión `.service`, esta unidad va a ejecutar un script que está en `/usr/bin/miScript.sh` que necesita que esté montado el sistema de ficheros (lo único que hace es mirar si existe un archivo en un directorio de log y si no existe lo crea y añade un timestamp al fichero).

```

1  [Unit]
2  Description=mi script de prueba
3  # como el script no depende de nada mas que el sistema de fichero queremos que se
   # ejecute lo antes posible en el arranque sin agregar tiempo de booteo a la maquina,
   # por lo tanto, solo es necesario que antes del script tenga cargado el tmpfiles
4  # local-fs.target nos asegura que el sistema de ficheros este montado.
5  After=sysinit.target systemd-tmpfiles-setup.service local-fs.target
6  Before=multiuser.target # Before para que se ejecute antes de que se arranque el
   multiuser.target
7
8  [Service]
9  Type=oneshot # indica que el script se ejecutara una sola vez, esto es adecuado porque
   el script no queda en segundo plano
10 ExecStart=/usr/bin/miScript.sh
11 RemainAfterExit=true # hace que el servicio se considere activo incluso una vez haya
   terminado, esto nos evita reinicios inesperados.
12 # el servicio al ser del tipo oneshot, systemd una vez que acaba el servicio, marca el
   servicio como inactivo y por tanto lo considera como detenido o terminado, con el
   true nos queda activo aunque haya sido terminado y asi systemd recuerda que fue
   ejecutado correctamente, sino, lo podria considerar inactivo y/o muerto y esto
   podria causar problemas de dependencias
13
14 [Install]
15 WantedBy=multiuser.target # establece el target (objetivo/s) bajo los que el servicio
   deberia estar iniciado.

```

Una vez acabado el archivo, para comprobar que no hay errores de sintaxis utilizamos el comando

```
1 $ systemd-analyze /ruta/a/archivo.service
```

recordar los permisos necesarios al service, propietario root, permisos de lectura y escritura para el root y permisos de lectura para otros usuarios por los que nos queda el octeto 644. Por último paso queda recargar los servicios con el comando:

```
1 $ sudo systemctl daemon-reload
```

Hay más parámetros que se pueden configurar dentro de nuestro service. Por ejemplo:

```

1  Requires=nombre-servicio.service # significa que necesitamos que el service del requiere
   este activo antes que nuestro service
2  Wants=nombre-servicio.service # indica que el servicio que este puesto en wants debe
   iniciarse junto con nuestro servicio (si esta habilitado)
3  Restart=always # por ejemplo con always podemos decir que se reinicie el servicio si
   falla

```

hay más parámetros como `ExecStart`, `ExecStop`, `User`, `WorkingDirectory`, entre muchos otros.

5.2.3 Creacion de un .target en systemd

Los targets agrupan servicios o estados del sistema. En `/etc/systemd/system` vamos a definir un fichero `miTarget.target`.

```

1  [Unit]
2  Description=Modo especial sin entorno grafico

```

```

3   Requires=network.target miServicio.service #Nuestro target necesita de red y de un
      servicio personalizado (miServicio)
4   After=network.target #Nuestro target se ejecuta despues de que la red este lista
5   Wants=sshd.service #por ejemplo si queremos contar tambien con el servicion ssh (no
      tenemos entorno grafico)
6
7   [Install]
8   WantedBy=multi-user.target # se activara en el entorno multiusuario

```

5.2.4 Creacion de un .mount en systemd

Los archivos .mount permiten definir archivos de montaje de disco, vamos a definir un .mount en /etc/systemd/system/miMount.mount

```

1   [Unit]
2   Description=Montaje automatico de datos
3   Requires=network.target # queremos que la red este disponible antes del montaje
4   After=network-only.target # se monta cuando la red este completamente operativa
5
6   [Mount]
7   What=/dev/sdb1 # es el dispositivo a montar
8   Where=/mnt/datos # ruta donde vamos a montar el dispositivo
9   Type=ext4 # filesystem type
10  Options=defaults #opciones de montaje por defecto
11
12  [Install]
13  WantedBy=multi-user.target

```

5.2.5 Creacion de un .timer en systemd

Los timers permiten programar tareas para que se ejecuten periódicamente de forma similar a cron. Vamos a definir nuestro .timer personalizado en /etc/systemd/system/miTimer.timer que ejecutará nuestro service personalizado cada 10 min (por ejemplo)

```

1   [Unit]
2   Description= Ejecuta mi SERVICE cada 10 min
3   Requires=miService.service # suponemos service en mismo directorio que timer
      #necesitamos que nuestro servicio personalizado este disponible (por eso el requires)
4
5
6   [Timer]
7   OnBootSec=5min # Primera ejecucion que se realice a los 5 min del arranque
8   OnUnitActiveSec=10min # Sucesivas ejecuciones que se realicen cada 10 min
9   Unit=miService.service # define el servicion a ejecutar
10
11  [Install]
12  WantedBy=timers.target # se activa en el grupo de timers

```

5.2.6 Creacion de un .socket en systemd

Los sockets permiten lanzar servicios cuando se recibe una conexión, para ello, vamos a usar un servicio creado por nosotros mismos a parte que será un servidor (ya implementado) de modo que cuando el socket reciba las conexiones, lanzará una nueva instancia del servicio por cada conexión recibida. La ruta del servicio y del socket suponemos que esta en /etc/systemd/system.

```

1   [Unit]
2   Description= Escucha en el puerto 8080
3
4   [Socket]
5   ListenString=8080 #define el puerto donde se van a escuchar las conexiones
6   Accept=yes # crea una nueva instancia del servicio por cada conexion recibida
7
8   [Install]
9   WantedBy=sockets.target # se activa en el grupo de sockets

```

5.2.7 Creacion de un .slice en systemd

Los slice permiten asignar límites de CPU y memoria a servicios, vamos a crear nuestro slice personalizado en /etc/systemd/system/miSlice.slice

```
1 [Unit]
2 Description= Grupo de control para limitar recursos
3
4 [Slice]
5 CPUQuota=50% # maximo de 50% de CPU asignada
6 MemoryLimit=500M # maximo de 500MB de RAM asociada
```

Ahora para asignar un servicio al slice, debemos ejecutar

```
1 $ sudo systemctl set-property nombre-servicio.service Slice=miSlice.slice
```

5.2.8 Creacion de un .network en systemd

Los network son archivos de configuración que nos permiten configurar y administrar las interfaces de red y sus conexiones, crearemos uno en /etc/systemd/network/eth0.network

```
1 [Match] #define en que interfaz aplicar la configuracion
2 Name=eth0
3
4 [Network] #Seccion donde vamos a configurar la parte de red
5 Address=192.168.1.100/24 #Ip fija que queremos que tenga la interfaz
6 Gateway=192.168.1.1 #la puerta de enlace para esa interfaz
7 DNS=8.8.8.8 #el DNS para esa interfaz
```

Ahora para aplicar los cambios ejecutar:

```
1 $ sudo systemctl restart systemd-networkd
2 #Comprobar la configuracion con
3 $ip addr show eth0
4 $ip route
```

Si hubiéramos querido que la interfaz obtuviese su configuración por dhcp en la sección [Network] tendríamos que haber dejado solo DHCP=yes Al igual que hemos mostrado como configurar una IPv4 estática y una configuración dinámica, se puede configurar IPv6, VLANs, Bounding (Agregación de enlaces), etc.

6 Funcionamiento y Arranque del Sistema

6.1 Arranque del sistema

Systemd sigue un flujo estructurado a la hora de arrancar. Para introducir el arranque de la máquina, vamos a ver primero los pasos de manera simplificada:

1. Encendido y ejecución del firmware (Normalmente BIOS o UEFI).
2. Carga del cargador de arranque.
3. El gestor de arranque GRUB (u otro) carga el kernel de Linux.
4. El kernel inicial el proceso systemd (PID 1) que es la raíz de todo el arbol de procesos del sistema.
5. Systemd monta los sistemas de archivos esenciales.
6. Se ejecutan los targets.units segun las dependencias (como el basic.target, multi-user.target, etc..).
7. Los servicios y daemons requeridos se inician paralelamente.
8. Se llega al estado de operación normal del sistema.

Ahora desde una vista más detallada, vamos a explicar el proceso completo de arranque de una máquina linux con systemd:

6.1.1 Encendido y ejecucion del firmware

Cuando presionamos el botón de encendido, la CPU se inicializa y ejecuta el firmware del sistema que normalmente sera BIOS en equipos más antiguos o UEFI en equipos mas modernos.

- BIOS: Realiza el POST (Power-On Self Test), verifica el hardware y busca un dispositivo de arranque (disco duro, USB, red, etc).
- UEFI: Similar a BIOS pero con mas funcionalidades como el arranque seguro (Secure Boot), compatibilidad con particiones GPT y un sin fin de opciones de configuración.

Luego, el firmware busca el cargador de arranque en el dispositivo de almacenamiento

- En firmware BIOS, busca el código del MBR (Master Boot Record) que contiene el cargador de arranque o un código genérico que carga el primer bloque de la partición marcada como activa donde estará el código del cargador de arranque.
- En UEFI, busca un archivo en la partición EFI, generalmente `/EFI/BOOT/bootX64.efi`, como aclaración indicar que en la configuracion de la UEFI se puede poner otro cargador que este en una partición `/EFI` de un dispositivo desde el que se pueda arrancar.

6.1.2 Cargador de arranque

El firmware carga el cargador de arranque y le pasa a este el control. EL cargador de arranque será el encargado de cargar el kernel del sistema operativo, podemos destacar los siguientes cargadores

- GRUB (Grand Unified Bootloader): es el más usado, permite escoger entre distintos sistemas operativos, opciones de recuperación, modificación de parámetros que se le pasan al kernel, etc.
- Systemd-Boot: más simple que el GRUB, se puede usar en sistemas con UEFI.
- Podemos mencionar mas cargadores como: refind, Lilo, syslinux entre otros cargadores más antiguos y/o especializados.

El cargador de arranque cargará la imagen del kernel en memoria, la imagen inicial con los módulos del kernel (si aplica) (initrd o initramfs), le pasa parametros al kernel como puede ser la ubicación de la partición raíz del sistema (ROOT).

6.1.3 Carga del kernel del sistema

El kernel, que es la parte central del S.O, se encarga de inicializar los controladores del hardware (drivers), configurar la memoria y los procesos, configurar el sistema de ficheros virtual (`/proc`, `/sys`, `/dev`), montar un sistema de ficheros mínimo con initramfs o initrd (si aplica) entre otras funciones.

Cuando el kernel ya tiene acceso al sistema de ficheros raíz (ROOT), desmonta el initramfs (si aplica) y transfiere el control al proceso de inicio con PID 1 (en nuestro caso con un sistema con systemd sera systemd).

6.1.4 El Proceso de inicio (PID 1)

El primer proceso que se ejecuta en un sistema con systemd es systemd (ubicado en `/sbin/init` el cual es un enlace simbólico a `/lib/systemd/systemd`). Este proceso se encarga de:

1. Leer la configuración en `/etc/systemd/system/default.target` para determinar el modo de inicio (destacar los modos emergency, multi-user, graphical); este `.target` le dirá a systemd las demás units y el gráfico de dependencias para saber en que orden inicializar.
2. Montar los sistemas de ficheros definidos en el `/etc/fstab`.
3. Ejecutar los targets de arranque en orden:
 - (a) `sysinit.target` → inicializa servicios esenciales como registros de logs, configuración de dispositivos y archivos del sistema.

- (b) `basic.target` → carga las unidades básicas necesarias para el funcionamiento del sistema.
- (c) `(multi-user, graphical, emergency).target` → dependiendo del entorno, carga todos los servicios asociados al `.target`.
- (d) Iniciar servicios del sistema como puede ser `Networkd.service` (configuración de red), `cron.service`, `sshd.service`, `gdm.service` (o similar como `ssdm`, `lightdm`, etc. Para gestores de sesiones graficas).

6.1.5 Sesión de Usuario y Login

Una vez que `systemd` ha terminado de cargar todos los servicios necesarios, el sistema está en pleno funcionamiento y en modo de operación normal, nos carga un login:

- Con `graphical.target` nos cargará un gestor de pantalla como puede ser `gdm`, `lightdm`, `sddm`, `xdm`, etc.
- Con `multi-user.target` el servicio `getty@tty1.service` nos proporcionará un prompt donde nos podremos loggear y tener una shell.

Tenemos varios comandos de `systemd` para ver el arranque de nuestro sistema, hemos querido destacar los siguientes:

- `systemd-analyze blame` → el sistema nos devuelve una lista ordenada de units con su respectivo tiempo de inicio de mayor a menor (tiempo). Esta lista muestra cuanto tiempo tardó cada unit en activarse completamente después de que `systemd` lo haya comenzado.
- `systemd-analyze critical-chain` → se obtiene una representación en forma de árbol jerárquico, donde cada unit depende de otro, muestra el tiempo que tardó cada unit en arrancar.
- `systemd-analyze plot` → este comando genera un gráfico SVG mostrando visualmente el tiempo de arranque de cada unit.
- `systemd-analyze time` → ejecutaremos este comando si solo queremos saber el tiempo que tardó en arrancar el sistema; el tiempo nos lo divide en:
 1. Tiempo de Kernel: tiempo que tardó el kernel en arrancar
 2. User Space: tiempo de usuario, tiempo que tardó en arrancar `systemd` y las units asociadas al usuario.
- `systemctl list-dependencies nombre-servicio.service` → muestra las dependencias de un servicio.

7 Registro y Monitoreo

7.1 Gestión de logs con journalctl

Tal como comentamos en unas secciones anteriores, ahora vamos a explicar más en detalle la gestión de los logs con `systemd`. `Systemd` usa el daemon `journald` para la gestión y visión de logs, substituyendo a los antiguos logs en formato texto como `syslog`, `rsyslog`, etc. Los componentes principales de `journald` son:

1. `systemd-journald.service` → proceso principal que administra el servicio `journal`.
2. `systemd-journald-flush.service` → proceso que se encarga de hacer flush de los logs que estén en memoria (guardarlos en disco).
3. `/run/systemd/journal/` → almacén de logs en memoria, estos logs se pierden al reiniciar el sistema (son volátiles).
4. `/var/log/journal/` → almacén persistente de logs en disco, se requiere que el `journal` esté configurado para que guarde persistentemente los logs.
5. `journalctl` → herramienta de línea de comandos para la administración y consulta de los logs; esta herramienta es necesaria debido a que el formato de los logs de `journal` no es formato texto plano, si no que es formato binario.

`Journald` puede capturar logs desde diferentes aplicaciones y servicios del sistema. Vamos a destacar los siguientes:

- STDOUT/STDERR estándar de los procesos supervisados por systemd. Podemos crear un flujo de datos, a esto se le llama namespace. A un mismo namespace podemos meter 1 o más procesos. De cada namespace se encarga systemd-journald@namespace.service. Por ejemplo systemd-journald@ssh.service.
- Llamadas desde lenguajes de programación a la API; sd_journal_send().
- Puede recibir logs desde syslog si está configurado para ello.
- Recibir eventos del kernel (/dev/kmsg) como los logs de arranque y mediante el comando dmesg.

Journald a los logs les asigna metadatos como puede ser: PID, UID, cgroup, timestamp, etc. Los logs se guardan en memoria (en /run) y con la configuración a continuación se podrán guardar en disco (/var/log/journal). Para la configuración de los logs persistentes y más configuraciones sobre journald, debemos modificar su fichero de configuración ubicado en /etc/systemd/journald.conf. Vamos a ver las siguientes opciones:

- Storage=volatile, persistent, auto, none # (volatile solo en RAM, persisten en disco, auto usa el disco si está disponible y si no en la RAM, none para no guardar logs (se desactiva)).
- SystemMaxUse=500M # límite del tamaño para los logs del disco.
- SystemKeepFree=10% # deja un 10% libre en disco.
- SystemMaxFileSize=50M # tamaño máximo de cada archivo de journal.
- RuntimeMaxUse=100M # límite de espacio para los logs en RAM.
- Compress=yes # habilita la compresión de logs.
- ForwardToSyslog=yes # habilita el envío de logs a syslog.
- RateLimitInterval=10s # intervalo de tiempo para limitar los registros.
- RateLimitBurst=1000 # Número máximo de logs permitidos en RateLimitInterval.

Vamos a explicar la estructura de los logs, journald almacena los logs con una estructura binaria, esto permite una indexación eficiente en las búsquedas, compresión automática (si está configurado) para reducir el uso de espacio, integridad mediante un hash de datos y registros firmados, cada entrada del journal contiene campos estándar (_PID, _UID, _GID, _CONN, _EXE, _CMDLINE, _BOOT_ID, _MACHINE_ID, _SYSTEMD_UNIT entre otros). Esto es un ejemplo de una entrada de log en formato JSON sacada mediante el comando journalctl -o json-pretty:

```

1  {
2    "__REALTIME_TIMESTAMP" : "1743696360823850",
3    "__MONOTONIC_TIMESTAMP" : "2018132629",
4    "_PID" : "754",
5    "_HOSTNAME" : "samusArch",
6    "_TRANSPORT" : "syslog",
7    "_SYSTEMD_INVOCATION_ID" : "ca97f154d9ba440dbcd8b84904f13c4b",
8    "_SYSTEMD_SLICE" : "system.slice",
9    "__SEQNUM_ID" : "733e2846c6c94a92b4ad9f048da0ae18",
10   "_GID" : "0",
11   "_BOOT_ID" : "bfd16b3cfb3147419cc41c934e30cbb7",
12   "_SYSTEMD_UNIT" : "wpa_supplicant.service",
13   "SYSLOG_IDENTIFIER" : "wpa_supplicant",
14   "_MACHINE_ID" : "27f763834a434567b677b61c81afa737",
15   "SYSLOG_PID" : "754",
16   "_CAP_EFFECTIVE" : "1fffffffff",
17   "MESSAGE" : "wlan0: CTRL-EVENT-SIGNAL-CHANGE above=1 signal=-54 noise=9999 txrate
18     =360000",
19   "SYSLOG_TIMESTAMP" : "Apr  3 18:06:00 ",
20   "SYSLOG_FACILITY" : "3",
21   "_CMDLINE" : "/usr/bin/wpa_supplicant -u -s -O /run/wpa_supplicant",
22   "__SEQNUM" : "254537",
23   "_COMM" : "wpa_supplicant",
24   "PRIORITY" : "5",
25   "_SYSTEMD_CGROUP" : "/system.slice/wpa_supplicant.service",

```



```

25     "_RUNTIME_SCOPE" : "system",
26     "_UID" : "0",
27     "_SOURCE_REALTIME_TIMESTAMP" : "1743696360823829",
28     "_EXE" : "/usr/bin/wpa_supplicant",
29     "__CURSOR" : "s=733e2846c6c94a92b4ad9f048da0ae18;i=3e249;b=
        bfd16b3cfb3147419cc41c934e30cbb7;m=784a4295;t=631e1f22edc2a;x=b8541a0be2f6279a"
30 }

```

En cuanto a la seguridad, journald implementa protección de logs mediante las siguientes técnicas:

- MACs (Message Authentication Codes) → son códigos que sirven para verificar la integridad de los logs y asegurarse que éstos no fueron modificados. Se puede habilitar la integridad con `Seal=yes` en `journal.conf`
- Restricción de Acceso → los usuarios sin privilegios no pueden acceder a todos los logs.
- Se requiere estar en el grupo `systemd-journal` para ver los logs del sistema.
- Cifrado de logs (FDE o Full Disk Encryption) → Si `/var/log/journal` (donde se guardan los logs en disco) se encuentra en una partición cifrada con el sistema LUKS, los logs estarán cifrados bajo dicho cifrado.

Para mantener y gestionar los logs, tenemos los siguientes comandos:

- Ver el uso de espacio con → `journalctl -disk-usage`
- Para limpiar logs según tamaño → `journalctl -vacuum-size=500M`, borra logs limita a 500MB.
- Para limpiar logs según fecha → `journalctl -vacuum-time=30d`, para borrar logs más antiguos de 30 días.
- Si queremos eliminar todos los logs → `rm -rf /var/log/journal` y reiniciar el daemon de journal es suficiente.
- Para almacenar los logs que estén en memoria a disco → `journalctl -flush`, esto se va realizando de forma automática gracias al daemon `systemd-journal-flush.service` por lo que tendrá que estar habilitado en el arranque (`enable`).

Consultas avanzadas con `journalctl`:

- Filtrar por nombre-servicio servicio → `journalctl -u nombre-servicio.service`
- Por usuario o PID → `journalctl _UID=UUID_USUARIO`, `journalctl _PID=PID_PROGRAMA`
- Por fecha y hora → `journalctl --since "2025-04-01 12:00:00" --until "2025-04-03 18:52:14"` (entre dos fechas)
- Por contenido del mensaje, haciendo uso de la herramienta `grep` → `journalctl | grep "mensaje"`
- Ver los logs en tiempo real → `journalctl -f`
- Exportar los logs a formato JSON → `journalctl -o json-pretty > archivo.json`
- Consultar los logs según la prioridad → `journalctl -p prioridad` (siendo prioridad un número entre el 0 y el 7)

Comparación resumen entre `journald` y `syslog`:

- `Syslog` usa para los logs formato en texto plano mientras que `journald` es binario.
- `Journald` soporta indexación mientras que `syslog` no.
- `Journald` permite búsqueda avanzada mientras que `syslog` no.
- En `journald` la persistencia es opcional mientras que en `syslog` es por defecto.
- `Journald` asegura la integridad de los logs mediante hashes y MACs mientras que `syslog` no la asegura.
- En la compresión de logs, `journald` lo soporta mientras que `syslog` no (aunque hay herramientas auxiliares como `logrotate`).

8 Componentes Avanzados de SystemD

8.1 systemd-boot

Systemd-boot, anteriormente llamado gummiboot, es un gestor de arranque UEFI para kernel's de linux (solo funciona en UEFI, es decir, no soporta BIOS, y es más simple y rápido que grub pero con menos características avanzadas). Para gestionarlo, se utiliza la herramienta `bootctl`. Para instalar systemd-boot, hacemos

```
1 # esto nos copiará los archivos de boot en la partición UEFI (normalmente montada en /
  boot/efi)
2 $ sudo bootctl install
```

La configuración de systemd-boot se encuentra en `/boot/loader/` y tenemos dos archivos principales:

- `/boot/loader/loader.conf`: para configuración general.
- `/boot/loader/entries/*.conf`: entradas de cada sistema operativo.

Un ejemplo de configuración del `loader.conf` es:

```
1 # carga por defecto la entrada de nombre <entrada>.conf
2 default <entrada>
3
4 # espera 3 segundos antes de arrancar por defecto
5 timeout 3
6
7 # desactiva el editor activo en el menu de arranque
8 editor no
```

Para crear una entrada para un linux `/boot/loader/entries/miLinux.conf`:

```
1 # nombre que aparecera en el menu de arranque
2 title nombre
3 # kernel de linux
4 linux /vmlinuz-linux
5 # modulos del kernel de linux
6 initrd /initramfs-linux.img
7 # options para pasarle parametros al kernel, root=UUID especificamos la particion raiz,
  rw montamos la raiz en modo lectura/escritura, quiet para ocultar mensajes de inicio
  y splash para mostrar el fondo de plymouth en lugar de los mensajes de arranque.
8 options root=UUID=xx... rw quiet splash
```

Para actualizar systemd-boot ejecutaremos

```
1 $ sudo bootctl update
```

8.2 systemd-nspawn

Systemd-nspawn es como un chroot con esteroides, se puede usar para ejecutar un comando o un sistema operativo en un contenedor ligero, es más potente que chroot, ya que virtualiza tanto el sistema de ficheros raíz como el árbol de procesos, los subsistemas IPC y el nombre de host y de dominio. Tiene más aislamiento y seguridad que chroot. Se ejecuta en un entorno similar a un contenedor pero es más simple que docker o LXC. Para gestionar los servicios en un contenedor usa systemd.

Si tenemos un root filesystem en `/var/lib/machines/debian` podemos ejecutarlo como un contenedor con

```
1 # esto nos crea un entorno similar a una maquina virtual pero sin la sobrecarga de un
  hipervisor
2 $ sudo systemd-nspawn -D /var/lib/machines/debian # -D nos inicia una shell en el
  contenedor, con aislamiento de procesos y montajes
```

Antes de usar systemd-nspawn, necesitamos un root filesystem (sistemas de ficheros básico con linux). Para crearlo, vamos a hacer un ejemplo tanto en debian como en arch linux.

```

1  # ejemplo en debian/ubuntu
2  $ sudo debootstrap stable /var/lib/machines/debian http://deb.debian.org/debian/
3
4  # ejemplo para arch linux
5  $ sudo pacstrap -c /var/lib/machines/archlinux base

```

Tenemos unas cuantas opciones para personalizar el comportamiento del contenedor con diferentes flags:

- `-D /ruta` → define la ruta del root filesystem del contenedor.
- `-boot` → inicia systemd dentro del contenedor (requiere systemd en el contenedor).
- `-network-veth` → crea una interfaz de red virtual veth para conectar el contenedor.
- `-bind /ruta:/destino` → monta una carpeta externa dentro del contenedor.
- `-read-only` → monta el root filesystem en modo solo lectura.
- `-private-users=pick` → usa un espacio de usuarios no privilegiado (mayor seguridad).

Ejemplo de iniciar un contenedor con red y systemd activado:

```

1  $ sudo systemd-nspawn --boot --network-veth -D /var/lib/machines/debian

```

systemd-nspawn se integra con machinectl, la cual es una herramienta para gestionar contenedores con systemd. Tenemos los siguientes comandos, donde debian va a ser el nombre de nuestro contenedor de ejemplo:

- `machinectl list-images` → muestra las imágenes de contenedores disponibles en `/var/lib/machines`.
- `machinectl start debian` → iniciar un contenedor.
- `machinectl stop debian` → apagar un contenedor.
- `machinectl reboot debian` → reiniciar el contenedor.
- `machinectl poweroff debian` → similar al comando stop.
- `machinectl status debian` → muestra el estado del contenedor.
- `machinectl login debian` → inicia una shell dentro de un contenedor ya iniciado previamente.
- `machinectl shell debian` → similar a login pero nos inicia sesión ya con un usuario y no con el prompt del login.

Muchas de las utilidades de systemd han sido actualizadas para que funcionen con contenedores, normalmente se usará la flag `-M` o `--machine=` y se le pasa el nombre del contenedor como argumento. Ejemplo de sintaxis:

```

1  $ journalctl -M debian

```

8.3 systemd-slice

systemd-slice es una funcionalidad de systemd que nos ayuda a organizar y limitar el uso de recursos de procesos en el sistema, se basa en los cgroups (Control Groups) para gestionar jerárquicamente el uso de cpu, memoria y I/O de disco entre otros. Una unit.slice agrupa procesos relacionados bajo una jerarquía de control de recursos. Por defecto systemd tiene tres slices principales:

- `system.slice`: contiene servicios de systemd (.service), aquí podemos incluir servicios como el `sshd.service` o `cron.service`
- `user.slice`: contiene procesos iniciados por usuarios (.session).
- `machine.slice`: contiene máquinas virtuales y contenedores (.machine).

Los slices en systemd siguen una jerarquía similar a un sistema de ficheros, un ejemplo de jerarquía de slices es:

```
|--.slice          (Raíz de los slices)
|-- system.slice   (Servicios de 'systemd')
|   |-- sshd.service (Demonio SSH)
|   |-- cron.service (Demonio Cron)
|   |-- apache.slice (Un slice personalizado)
|   |   |-- apache.service (Servicio Apache)
|   |   |-- php-fpm.service (Servicio PHP)
|   |-- db.slice    (Base de datos)
|   |   |-- mysql.service
|   |   |-- redis.service
|-- user.slice      (Procesos de usuarios)
|   |-- user-1000.slice (Usuario con UID 1000)
|   |   |-- session-2.scope (Sesión gráfica)
|   |   |-- firefox.service (Ejemplo de aplicación)
|   |   |-- vscode.service (Ejemplo de aplicación)
|-- machine.slice   (Máquinas virtuales y contenedores)
|-- libvirt-1.scope (Máquina virtual)
|-- docker-xyz.scope (Contenedor Docker)
```

Podemos definir slices personalizados en `/etc/systemd/system`. Tenemos varios comandos:

- `systemd-cgls`: muestra la jerarquía de cgroups y los procesos de cada slice.
- `systemd-cgtop`: similar a la herramienta `top`, pero mostrando el uso de recursos por cgroups.

si un procesos ya esta corriendo, puedes moverlo a otro slice con

```
$ systemctl set-property --runtime nombre-servicio.service Slice=custom.slice
```

esto nos mueve `nombre-servicio.service` a `custom.slice` temporalmente, para hacerlo de forma permanente, debemos editar el archivo `.service`, añadiéndole `Slice=custom.slice`.

8.4 systemd-network

Systemd-networkd es un servicio de systemd que está diseñado para gestionar y configurar la red en sistemas linux, se integra con udev (gestor de dispositivos de linux que detecta, monta y configura hardware en tiempo real). Para ver las interfaces gestionadas por systemd-networkd se usa el comando

```
$ networkctl list
```

tenemos tres rutas para los archivos de configuración:

- `/etc/systemd/network/`: configuraciones personalizadas del sistema. Tiene la máxima prioridad.
- `/usr/lib/systemd/network/`: configuraciones predeterminadas del sistema.
- `/run/systemd/network/`: configuraciones temporales.

Para configurar una interfaz se crea un archivo `.network`, el cual tiene una sección `[Match]` donde se especifica la interfaz con el parámetro `'Name'` o a cualquiera si se deja en blanco, con un `!` delante son los que no coincidan; y luego tenemos una sección `[Network]` donde se le puede especificar la ip (parámetro `Address`), la puerta de enlace (parámetro `Gateway`), el DNS con el parámetro `DNS` o establecer que la configuración se asigne automáticamente por DHCP con el parámetro `DHCP`.

Ahora vamos a explicar la configuración de interfaces más complejas como las de las interfaces virtuales. Los archivos `.netdev` se usan para la creación de interfaces virtuales como las `vlan`, `tuneles`, `bonds` o `bridges`.

8.4.1 Creacion de bridge

Esta es la configuración del archivo `/etc/systemd/network/br0.netdev`:

```
1 [NetDev]
2 Name=br0
3 Kind=bridge
```

En `/etc/systemd/network/br0.network`:

```
1 [Match]
2 Name=br0
3
4 [Network]
5 Address=<ipv4>/<mask>
6 Gateway=<ipv4>
```

En `/etc/systemd/network/eth0.network`:

```
1 [Match]
2 Name=eth0
3
4 [Network]
5 Bridge=br0
```

siendo br0 el bridge y eth0 es la interfaz ethernet que se use al bridge.

8.4.2 Creacion de una VLAN

En la configuración del archivo `/etc/systemd/network/vlan10.netdev`:

```
1 [NetDev]
2 Name=vlan10
3 Kind=vlan
4
5 [VLAN]
6 Id=10
```

en la configuración del archivo `/etc/systemd/network/vlan10.network`:

```
1 [Match]
2 Name=vlan10
3
4 [Network]
5 Address=<ipv4>/<mask>
```

en la configuración del archivo `/etc/systemd/network/eth0.network`:

```
1 [Match]
2 Name=eth0
3
4 [Network]
5 VLAN=vlan10
```

creamos la vlan10 (con ID 10) sobre eth0 asignándole una IPv4.

8.4.3 Configuracion de propiedades fisicas

Para ajustar configuraciones de hardware o renombrar interfaces se usan los archivos `.link`, vamos a realizar un ejemplo renombrando 'eth0' a 'int0'. Para ello vamos a configurar el fichero `/etc/systemd/network/10-int0.link`

```
1 [Match]
2 MACAddress=<MAC interfaz eth0>
3
4 [Link]
5 Name=int0
```

se detecta la interfaz por la MAC y se renombra a `int0`.

Otra configuración física que podemos hacer es la configuración del MTU (Maximum Transmission Unit). Vamos a configurar el MTU para la interfaz `eth0`. En el archivo `/etc/systemd/network/eth0.link`:

```
1 [Match]
2 Name=eth0
3
4 [Link]
5 MTUBytes=9000 # Numero de bytes maximo de paquete (MTU) ahora sera de 9000 Bytes (Jumbo-
   frame)
```

8.4.4 Comandos de gestión

1. `networkctl list` → lista de las interfaces gestionadas por `systemd-networkd`.
2. `networkctl status <interface>` → ver detalles de la interfaz `<interface>`.
3. `systemctl restart systemd-networkd` → reiniciar el servicio `systemd-networkd`.
4. `journalctl -u systemd-networkd -no-paper` → mostrar los logs de red de `systemd-networkd`.
5. `ip addr show` → mostrar la configuración ip.
6. `ip route show` → ver las rutas de red.

8.5 systemd-resolved

`Systemd-resolved` es la parte de `systemd` que se encarga de gestionar la resolución de nombres de dominio (DNS o Domain Name System), ofrece funcionalidades como pueden ser la gestión de dominios locales, resolución de nombres mediante múltiples servidores DNS o la resolución de nombres mDNS (Multicast DNS). `Systemd-resolved` ofrece caché DNS, tiene soporte para DNS sobre TLS y se integra perfectamente con otros componentes de `systemd` como `systemd-networkd`.

La configuración de `systemd-resolved` radica en `/etc/systemd/resolved.conf`, que es el archivo de configuración principal y en la mayoría de sistemas en `/etc/resolve.conf`, que es un enlace simbólico que apunta a `/run/systemd/resolve/stub-resolv.conf`. Entre las opciones principales de `/etc/systemd/resolved.conf` tenemos

- DNS: Establece los servidores DNS a usar.
- FallbackDNS: define servidores DNS de respaldo.
- DNSSEC: permite habilitar/deshabilitar la verificación de la seguridad de las consultas DNS.
- Cache: habilita/deshabilita la caché DNS.

`systemd-resolved` dispone de una herramienta de administración, que es `resolvectl`, que soporta los siguientes comandos:

```
1 $ resolvectl status # nos muestra informacion sobre la configuracion DNS, el estado de
   las conexiones y los servidores usados
2
3 $ resolvectl query <domain> # consulta un nombre de dominio en especifico (por ejemplo
   www.example.com)
4
5 $ resolvectl flush-caches # borra las caches DNS almacenadas por systemd-resolved
6
7 #Tambien se puede usar como
8 $systemd-resolve <domain>
9
10 #Hay varias flags segun queramos el tipo de consulta DNS (inversa, correo, etc.)
```

8.6 systemd-analyze

Herramienta de diagnóstico que nos da información sobre el rendimiento de systemd. Permite analizar el tiempo que tarda en arrancar el sistema, la cantidad de tiempo que tarda cada servicio en arrancar, etc. Ya fueron explicadas en la sección 6.1.5 Sesión de usuario y login.

8.7 systemd-logind

Servicio de systemd que gestiona las sesiones de usuario, los inicios de sesión y la gestión de sesiones en sistemas linux, se encarga de tareas como el inicio de sesión de usuarios, el cierre de sesión, administración de energía como la suspensión y apagado del sistema. Systemd-logind asigna un id de usuario único (UID) y un id de sesión para poder identificar de forma única las sesiones activas. Para interactuar con las opciones de energía disponemos de los siguientes comandos:

```
1 $ systemctl suspend # solicita la suspension del sistema
2 $ systemctl hibernate # solicita la hibernacion del sistema
3 $ systemctl poweroff # solicita el apagado del sistema
4 $ systemctl reboot # solicita el reinicio del sistema
```

El servicio systemd-logind también se encarga de interactuar con los periféricos y dispositivos USB y los vincula a las sesiones de usuario correspondientes. El archivo de configuración principal se ubica en `/etc/systemd/logind.conf`. Podemos hacer configuraciones en archivos de configuración secundarios en la ruta `/etc/systemd/logind.conf.d/`. Podremos configurar en él aspectos relacionados con las sesiones de los usuarios como la gestión de la energía, la inactividad de las sesiones, etc. Un ejemplo de configuración del archivo es el siguiente:

```
1 ##### LAS LINEAS QUE COMIENZAN POR # NO SE USAN, ESTAN COMENTADAS Y POR TANTO
2   DESHABILITADAS #####
3 [Login]
4 NAutoVTs=6 # indica cuantas terminales virtuales se deben generar, 6 indica de crear de
5   la /dev/tty1 a la /dev/tty6
6 ReserveVT=6 # indica que tty se reserva para el gestor de login grafico (gdm, sddm, xdm,
7   lightdm...)
8 KillUserProcesses=no # controla si los procesos de un usuario deben ser matados al
9   cerrar sesion (valores posibles yes o no)
10 #KillOnlyUsers= # solo se matan procesos de estos usuarios
11 #KillExcludeUsers=root #que nunca se maten procesos del usuario (en este caso root)
12 #InhibitDelayMaxSec=5 #indicar el tiempo maximo en segundos que se permite retrasar un
13   apagado/reinicio/suspension... por un inhibidor (mecanismo por el cual un proceso
14   puede impedir temporalmente que se lleve a cabo una operacion del sistema como puede
15   ser un apagado, reinicio, etc. Ej pensar en el VLC que puede hacer que mientras se
16   reproduce una pelicula no se suspenda el sistema)
17 #UserStopDelaySec=10 # tiempo de espera en segundos para finalizar recursos de un
18   usuario despues de que este haya terminado su sesion
19 #SleepOperation=suspend-then-hibernate suspend # define las operaciones que se
20   consideren modo suspension
21 #las opciones del anterior comando son (suspend -> suspender; suspend-then-hibernate ->
22   suspender y al rato hiberna)
23 #Estas lineas de Handle.. se refieren a eventos de botones y comportamiento de la tapa (
24   mi maquina es un portatil)
25 #HandlePowerKey=poweroff # presionar el boton de encendido es apagar
26 #HandlePowerKeyLongPress=ignore #mantener el boton de encendido esta ignorado
27 #HandleRebootKey=reboot #presionar boton de reinicio es reiniciar
28 #HandleRebootKeyLongPress=poweroff #mantener el boton de reinicio es apagar
29 #HandleSuspendKey=suspend #presionar el boton de suspension suspende
30 #HandleSuspendKeyLongPress=hibernate #mantener el boton de suspension hiberna
31 #HandleHibernateKey=hibernate #presionar el boton de hibernar hiberna
32 #HandleHibernateKeyLongPress=ignore #mantener el boton de hibernar se ignora
33 #HandleLidSwitch=suspend #se define que hacer al cerrar la tapa del portatil (con
34   bateria), en este caso suspender
35 #HandleLidSwitchExternalPower=suspend # que hacer al cerrar la tapa del portatil si esta
36   enchufado el equipo a la corriente, en este casos suspender
37 #HandleLidSwitchDocked=ignore #que hacer si se cierra la tapa del portatil y este esta
38   acoplado (tiene conectados HDMI, teclado/raton USB, etc) , en este caso se ignora
39 #HandleSecureAttentionKey=secure-attention-key # especificar la accion que hacer al
40   presionar la combinacion de teclas 'CTRL+ALT+DEL'
```

```

25 #PowerKeyIgnoreInhibited=no #Controlar si al presionar el boton de encendido se ignora
    su significado si hay algun inhibidor o no
26 #SuspendKeyIgnoreInhibited=no # lo mismo pero para boton suspender
27 #HibernateKeyIgnoreInhibited=no
28 #LidSwitchIgnoreInhibited=yes # ej el portatil se suspendera aunque haya algun inhibidor
    (ej el VLC reproduciendo una pelicula) si la tapa del portatil es cerrada.
29 #RebootKeyIgnoreInhibited=no
30 #HoldoffTimeoutSec=30s # tiempo de espera para no tener en cuenta presiones sobre
    botones muy rapidas y seguidas.
31 #IdleAction=ignore #accion a ejecutar entre (ignore, suspend, hibernate, poweroff)
    cuando se lleva un tiempo (IdleActionSec) sin entrada de usuario
32 #IdleActionSec=30min
33 #RuntimeDirectorySize=10% #definir cuanto espacio puede usar /run/user/<uid>/ (como
    porcentaje sobre el total de /run/)
34 #cuando un user inicia sesion, systemd-login le crea un directorio temporal que es /run/
    user/<uuidUser>/ ; es usado por los procesos del usuario para almacenar archivos
    temporales, esta montado en RAM y no en disco por lo que es volatil.
35 #RuntimeDirectoryInodesMax= # numero maimo de inodos disponibles en los directorios de
    usuario bajo /run. Limita el numero de archivos que puede haber en cada /run/user/<
    uid> . Nos evita que por un problema sea de seguridad o fallo de una aplicacion, se
    consuman todos los inodos disponibles aun quedando espacio en el directorio
36 #RemoveIPC=yes # borrar recursos IPC (Inter-Process-Communication) (ej colas de mensajes,
    memoria compartida, semaforos) cuando un usuario se desconecta. Tenerlo activado (
    yes) puede ser problematico en servidores multiusuario o con servicios en segundo
    plano que usan IPCs.
37 #InhibitorsMax=8192 # Numero maximo de inhibidores permitidos
38 #SessionsMax=8192 #Numero maximo de sesiones permitidas al mismo tiempo
39 #StopIdleSessionSec=infinity #tiempo para que una sesion inactiva pase a detenerse
    automaticamente
40 #DesignatedMaintenanceTime=

```

Para ver la total configuración de login (el archivo principal y los secundarios) con el comando `systemd-analyze cat-config systemd/login.conf`. El comando es muy útil ya que nos sirve para poder ver las configuraciones activas que se están aplicando sin tener que ver todos los archivos de configuración en las rutas correspondientes. Systemd-login dispone de la herramienta `loginctl` para la gestión y el manejo de las sesiones.

Vamos a listar los comandos mas usados:

- `loginctl list-sessions` → lista las sesiones activas en el sistema.
- `loginctl start <sessionID>` → inicia la sesión con ID `sessionID`.
- `loginctl stop <sessionID>` → para la sesión con ID `sessionID`.
- `loginctl show <sessionID>` → muestra información sobre la sesión con ID `sessionID`.
- `loginctl terminate <sessionID>` → cierra la sesión con ID `sessionID`.
- `loginctl` → muestra información sobre el estado de la sesión del usuario actual.
- `loginctl poweroff` → apaga el sistema.
- `loginctl reboot` → reinicia el sistema.

8.8 systemd-creds

Este servicio proporcionado por `systemd` está diseñado para almacenar y recuperar de forma segura credenciales usadas por las unidades de `systemd`. Facilita la manipulación de nombres de usuario, contraseñas, claves de APIS, etc. Como funciones principales podemos destacar la integración directa con las units de `systemd`, lo que facilita su uso en servicios y aplicaciones; el almacenamiento seguro evitando la exposición de claves y nos ofrece la posibilidad de, mediante una clave derivada de un chip TPM2 o almacenada en `/var/`, cifrar credenciales.

Ejemplo de cifrado de una credencial, primero vamos a verificar que nuestro sistema disponga de cifrado TPM2 o de una clave almacenada en `/var/`.

```
1 $ systemd-analyze has-tpm2
```


y nos saldra un output parecido a este:

```
yes
+firmware
+driver
+system
+subsystem
+libraries
+libtss2-esys.so.0
+libtss2-rc.so.0
+libtss2-mu.so.0
```

Para cifrar una credencial almacenada en `credential.txt` (el sistema usará el UUID del usuario que cifra la credencial y el identificador único de la maquina o `machine-id`, que se encuentra en `/etc/machine-id`) se usa el comando

```
$ echo -n "api-clave-t00r25" | systemd-creds --user encrypt - credential.cifrada
```

Esto nos genera un archivo `credential.cifrada` con la clave cifrada que solo el usuario que la cifró podrá descifrar. Para descifrarla, usaremos

```
$ systemd-creds --user decrypt credential.cifrada
```

`Systemd-creds` también se puede usar para usuarios no root, cabe mencionar que el alcance del usuario y el `/etc/machine-id` se usan para cifrar la credencial, esto significa que el sistema usa ciertos datos en un contexto para cifrar la clave, de forma que sólo se pueda descifrar en un contexto específico, por lo tanto, un usuario root no podrá descifrarla. Debemos tener en cuenta que si movemos la credencial a otra máquina con diferente `machine-id`, tampoco seremos capaces de descifrarla.

¿Como se integra ésto con las demás unidades de `systemd`? las credenciales se configuran en los archivos de configuración de `units`, mediante las opciones

- `ImportCredential`
- `SetCredential`
- `LoadCredentialEncrypted`
- `LoadCredential`
- `SetCredentialEncrypted`

Entre las ventajas de `systemd-creds` destacamos las siguientes:

- Podemos integrarlo automáticamente con `systemd`.
- Aunque un atacante nos robe el archivo `credential.cifrada`, nos los copie, etc, no podrá descifrarlo debido a que su `machine-id` difiere del del usuario que cifro los datos.
- Ni siquiera el root podrá descifrar la credencial de otro usuario al no tener el mismo contexto y desconocer que usuario la cifró.

Cabe mencionar que es posible el uso de `systemd-creds` sin asociarlo a un UUID de usuario o a un `machine-id` para que sea capaz de funcionar en diferentes máquinas y/o usuarios.

9 Conversión de un `.service` a `SysVinit`

Para la explicación de esta parte vamos a tomar el ejemplo de convertir el `.deb` del `PulseSecure` para linux que está disponible para su descarga en la página de la Universidad de Cantabria y que es totalmente funcional con la VPN de la UDC. Lo vamos a convertir de un `.deb` para sistemas con `systemd` a un `.deb` para sistemas con `SysVinit`. Una vez descargado el paquete desde este enlace <https://sdei.unican.es/software/PulseSecure.deb>, en una terminal, crearemos una carpeta que se llame por ejemplo `'Pulse'`. Ahora descomprimiremos el `.deb` para ver todos sus archivos a esta nueva carpeta con el comando:

```
$ sudo dpkg-deb -R PulseSecure.deb Pulse
```

Entramos en la carpeta Pulse → `$ cd Pulse` y listamos todo el contenido recursivamente con `$ tree`. Ahora vamos a explicar el contenido que contiene este .deb, está compuesto por los directorios:

- DEBIAN → carpeta con información sobre el paquete y las dependencias, destacar el fichero `control` y los scripts de postinstalación, etc.
- lib → dentro de esta carpeta tenemos la carpeta `systemd` donde encontramos el archivo de unidad de `systemd` para el binario: `pulsesecure.service`.
- opt → dentro de esta carpeta, en `pulsesecure/bin` encontramos los binarios del programa. Destacar también las librerías dinámicas (.so) en `pulsesecure/lib`.
- usr → en esta carpeta encontramos una subcarpeta 'share' con iconos, el archivo de configuración del pulse para el menú de aplicaciones (.desktop), etc. También se encuentra documentación sobre el pulse para 'man', etc.

Una vez hemos desglosado el paquete, vamos a centrarnos en transformar el .service del pulse a un script válido para SysVinit. Primero vamos a editar el archivo DEBIAN/control y suprimiremos cualquier dependencia de `systemd` (en `Depends` o `Pre-Depends`) puesto que el paquete ya no va a depender de este (en el caso de este paquete, no había ninguna). Ahora ejecutando `find * -name "*.service"` desde el directorio 'Pulse' para encontrar el .service del pulse, y nos saldrá el archivo `lib/systemd/system/pulsesecure.service`. Vamos a hacerle un 'cat' para ver su contenido:

```
#El contenido original carece de comentarios
[Unit]
Description=pulsesecure service Daemon #descripcion del servicio
After=network.target # para cargarse tiene que haberse iniciado primero el objetivo
#network.target, arrancara despues de network.target
#(representa el estado con la red ya levantada)
[Service]
Type=forking #tipo del servicio es forking que es que se espera que el servicio
#haga fork y el hijo sea el daemon es decir, el programa que se lanza no se queda
#en primer plano, se hace fork a si mismo, el padre finaliza y es el hijo el que
#va actuar en verdad de daemon
Restart=always #especifica que el servicio se reinicie si se detiene por cualquier
#motivo (fallo, salida, etc)
RestartSec=1
User=root #el usuario efectivo que ejecuta el servicio sera root
ExecStart=/opt/pulsesecure/bin/startup.sh start #el binario a ejecutarse al iniciar,
#si falla, systemd marcara el servicio como fallido
ExecStop=/opt/pulsesecure/bin/startup.sh stop #el binario a ejecutarse al parar

[Install]
WantedBy=multi-user.target #el servicio debe iniciarse con el objetivo multi-user
#(representa el estado operativo pero sin interfaz grafica)
```

Una vez hemos visto el contenido del .service y la explicación, vamos a crear un script de SysVinit que haga lo mismo, para ello vamos a crear un archivo en la ruta `/etc/init.d/pulsesecure`; primero crearemos esa carpeta (DETALLE: la ruta es relativa al directorio Pulse; `Pulse/etc/init.d/...`). Creamos dentro de esta ruta el archivo (QUITARLE LOS COMENTARIOS NUESTROS) `pulsesecure`:

```
#!/bin/sh
```

```
#LA CABECERA BEGIN INIT INFO es usada por herramientas como update-rc.d
```

```

### BEGIN INIT INFO
# Provides:          pulsesecure #nombre del servicio
# Required-Start:    $network # que servicio debe estar antes activo
# Required-Stop:     $network #que servicio debe estar despues detenido
# Default-Start:     2 3 4 5 #niveles de ejecucion donde se activa el servicio (2 a 5 (multiusuario))
# Default-Stop:      0 1 6 # niveles de ejecucion donde se detiene el servicio:
#0 es halt, 1 modo monousuario, 6 es reboot
# Short-Description: PulseSecure Service Daemon #descripcion corta
# Description:       Script de inicio para PulseSecure adaptado a SysVinit #descripcion larga
### END INIT INFO

#estas son variables

DAEMON=/opt/pulsesecure/bin/startup.sh # ruta del script del pulse secure
NAME=pulsesecure #nombre del servicio
PIDFILE=/var/run/${NAME}.pid # ubicacion opcional del archivo que contiene el PID
#del proceso (no es necesario, aunque se suele poner)

. /lib/lsb/init-functions #funciones auxiliares como
#log_daemon_msg y log_end_msg para mostrar mensajes por consola

#$1 es lo que recibe el script, (cuando haces el /etc/init.d/script {start, stop, restart})

#si es start
case "$1" in
start)
log_daemon_msg "Iniciando $NAME" #mostramos mensaje de iniciando el pulse
$DAEMON start #ejecuta el /opt/./startup.sh start (lo arranca)
status=$? #captura el codigo de salida
if [ $status -eq 0 ]; then #si el codigo es 0 (exito) marcamos inicio correcto
log_end_msg 0
else #si no (incorrecto)
log_end_msg 1 #marcamos inicio como error
fi
;;
stop) #igual que start pero para detenerlo (ejecuta stop)
log_daemon_msg "Deteniendo $NAME"
$DAEMON stop
status=$?
if [ $status -eq 0 ]; then
log_end_msg 0
else
log_end_msg 1
fi
;;
restart) #pa reiniciar el servicio
$0 stop #detiene el servicio; fijarse que $0 es la ruta de este script asi q llama a stop
sleep 1 #espera 1 segundo
$0 start #arranca el servicio
;;
status) #si se le pasa status
if pidof startup.sh > /dev/null; then #usa pidof para encontrar procesos startup.sh activos
echo "$NAME está activo" #muestra si esta activo
exit 0

```

```

else
echo "$NAME no se está ejecutando" #muestra que no esta activo (no hubo resultado pidof)
exit 1
fi
;;
*)
echo "Uso: $0 {start|stop|restart|status}" #si se le pasa cualquier otra cosa saca como usar
exit 1
;;
esac

exit 0 #fin del script

```

Le damos permisos de ejecución con `$ chmod +x pulsesecure` Ahora el siguiente paso es modificar los scripts de preinstalación y postinstalación que se encuentran en DEBIAN. Primero vamos a modificar el `postinst` dejándolo así: Queremos aclarar que el archivo de `preinst` (preinstalación) es opcional y este paquete no tiene. ARCHIVO ORIGINAL (`postinst` es para después de la instalación):

```

#las ACLs son Access Control List , que son listas de control de acceso en Linux que
#permiten el controlar quien y como se puede acceder a un directorio/archivo de una manera
#mas flexible que los permisos tradicionales de Linux

#Algunos comandos para ver las ACLs son:
#getfacl archivo -> ver ACLs del archivo
#setfacl -> para dar permisos

#setfacl -d -> sirve para definir ACLs por defecto para futuros archivos creados dentro
#-m g::r es para dar permisos de lectura al grupo
#-m o::r es para dar permisos de lectura a otros

#apunta a /var/lib/pulsesecure/pulse; donde pulse guarda archivos de configuración y datos
setfacl -d -m g::r /var/lib/pulsesecure/pulse
setfacl -d -m o::r /var/lib/pulsesecure/pulse

#esta parte de aqui es para migrar los archivos de configuracion de una version antigua
#This config migration step is for 91.R-8 Beta customers to migrate to future version
if [ -d /opt/PulseSecure/Pulse ]; then #si existe la carpeta (donde se instala el Pulse)
#copia los archivos viejos a /var/lib... y luego borra la carpeta
cp /opt/PulseSecure/Pulse/*.dat /var/lib/pulsesecure/pulse/
cp /opt/PulseSecure/Pulse/*.bak /var/lib/pulsesecure/pulse/
cp /opt/PulseSecure/Pulse/DeviceId /var/lib/pulsesecure/pulse/
rm -rf /opt/PulseSecure
fi
systemctl enable /lib/systemd/system/pulsesecure.service #marca el servicio como
#arrancable en la maquina usando systemd
systemctl start pulsesecure #inicia el servicio

```

ARCHIVO MODIFICADO:

```

setfacl -d -m g::r /var/lib/pulsesecure/pulse
setfacl -d -m o::r /var/lib/pulsesecure/pulse

#This config migration step is for 91.R-8 Beta customers to migrate to future version
if [ -d /opt/PulseSecure/Pulse ]; then

```

```

cp /opt/PulseSecure/Pulse/*.dat /var/lib/pulsesecure/pulse/
cp /opt/PulseSecure/Pulse/*.bak /var/lib/pulsesecure/pulse/
cp /opt/PulseSecure/Pulse/DeviceId /var/lib/pulsesecure/pulse/
rm -rf /opt/PulseSecure
fi

#Activar e iniciar servicio con SysVinit
if [ -x /etc/init.d/pulsesecure ]; then
    update-rc.d pulsesecure defaults
    /etc/init.d/pulsesecure start
fi
#basicamente es borrar las lineas de systemd
#y hacer que se ejecute nuestro nuevo script, el update-rc para que habilite
#el servicio, equivale al enable

```

Para el postrm (después de desinstalar) no hay que modificarlo, queda tal cual:

```

#!/bin/bash
if [ -d /var/lib/pulsesecure ]; then
    REPLY=y
    if [ "$DEBIAN_FRONTEND" != "noninteractive" ]; then
        read -p "Save all current Pulse Secure configuration settings?[Yy/Nn]" -n 1 -r
        echo # (optional) move to a new line
        if [[ $REPLY =~ ^[Nn]$ ]]; then
            rm -rf /var/lib/pulsesecure
        fi
    fi
fi
#esta parte se encarga de la desinstalacion del navegador embebido de chromium que es
#ese que sale cuando te intentas conectar a la VPN y te pide loguearte con las cuenta y
#credenciales de la universidad.
CEF_INSTALL_DIR=/opt/pulsesecure/lib/cefRuntime
CEF_LIB_NAME=libcef.so

if [ -f $CEF_INSTALL_DIR/Release/$CEF_LIB_NAME ]; then
    echo "'Chromium Embedded Browser' is not removed as part of the uninstallation"
    echo "for future use and it will occupy 1.1Gb disk space. The folder 'cefRuntime'"
    echo "is located at /opt/pulsesecure/lib/"
fi

```

Para el archivo prerm (antes de desinstalar) el original es:

```

systemctl stop pulsesecure
systemctl disable pulsesecure #deshabilitar el arranque de pulsesecure
PID_PULSEUI=$(pidof pulseUI) #si hay algun proceso del pulse, sacarle el PID
PID_PULSELAUNCHER=$(pidof pulselauncher)
PID_PULSESEC=$(pidof pulsesecure)
#terminar el proceso en ejecucion si es que esta
kill -9 $PID_PULSEUI 2>/dev/null
kill -9 $PID_PULSELAUNCHER 2>/dev/null
kill -9 $PID_PULSESEC 2>/dev/null

```

Modificado queda:

```
#!/bin/bash
#Deshabilitar el servicio y detenerlo antes de desinstalarlo
# Detener el servicio de PulseSecure
if [ -x /etc/init.d/pulsesecure ]; then
echo "Deteniendo el servicio PulseSecure..."
/etc/init.d/pulsesecure stop
fi

# Deshabilitar el servicio en el arranque
if [ -x /usr/sbin/update-rc.d ]; then
echo "Deshabilitando el servicio PulseSecure para que no se inicie automáticamente..."
/usr/sbin/update-rc.d pulsesecure remove
fi

PID_PULSEUI=$(pidof pulseUI)
PID_PULSELAUNCHER=$(pidof pulselauncher)
PID_PULSESVC=$(pidof pulsesecure)
kill -9 $PID_PULSEUI 2>/dev/null
kill -9 $PID_PULSELAUNCHER 2>/dev/null
kill -9 $PID_PULSESVC 2>/dev/nul
```

Ahora por último vamos a empaquetarlo de nuevo en un .deb; para ello nos vamos fuera de la carpeta 'Pulse' y ejecutamos el comando: `$ dpkg-deb -b Pulse PulseSecure-sin-systemd.deb`
 Ahora para instalarlo ejecutar `$ sudo dpkg -i PulseSecure-sin-systemd.deb`
 Si nos diera algún error por dependencias de systemd (faltó eliminar alguna dependencia) podemos instalarlo con:
`$ sudo dpkg -i --ignore-depends=systemd PulseSecure-sin-systemd.deb`

10. Conclusión

A modo de resumen, systemd es un sistema de inicio y gestión de servicios que reemplaza al tradicional SysVinit, cumpliendo la función de iniciar el sistema operativo y administrar sus servicios desde las primeras etapas del arranque. Actualmente, se ha convertido en el sistema de inicio por defecto en la mayoría de las distribuciones GNU/Linux.

Si bien systemd cumple eficientemente con la tarea de sustituir a SysVinit, ha sido objeto de críticas dentro de la comunidad debido a su tendencia a integrar funcionalidades que muchos consideran ajenas a un sistema de inicio, como la gestión de registros (duramente criticado debido a la obligación del uso de la herramienta `journalctl` para acceder a los logs en formato binario frente al formato texto de Sysv), dispositivos, redes o sesiones. Esta ampliación de responsabilidades ha generado un debate entre quienes valoran su eficiencia y quienes prefieren una filosofía más minimalista, acorde al principio Unix de “hacer una sola cosa y hacerla bien”.

No obstante, su adopción generalizada y su ecosistema de herramientas lo convierten en una solución robusta y moderna, cuya comprensión resulta esencial para cualquier profesional que trabaje con sistemas Linux en la actualidad debido a su gran popularidad.

Si nos vamos a las distribuciones con soporte y pensadas para empresa como puede ser RedHat vemos que la adopción de Systemd es total y es mayor que la de otras distribuciones pensadas más para el usuario convencional como puede ser ubuntu que aún mantiene funcionalidades controladas de manera externa y no por systemd. Un ejemplo de esto es que RedHat no tiene Cron y Ubuntu por defecto.

References

- [1] Jean-François Bouchaudy. *Linux - Preparación a la Certificación LPIC-1 (5ª Edición)*. Ediciones ENI, 2022. ISBN: 978-2409032615.
- [2] *systemd.io*. Disponible en: <https://systemd.io>. Último acceso: marzo de 2025.
- [3] Arch Linux. *Systemd - ArchWiki*. Disponible en: <https://wiki.archlinux.org/title/Systemd>. Último acceso: marzo de 2025.
- [4] openSUSE. *Systemd en openSUSE Wiki*. Disponible en: <https://es.opensuse.org/Systemd>. Último acceso: marzo de 2025.
- [5] Douglas R. Reno, DJ Lucas (eds.). *Linux from Scratch, Version 12.3-systemd*. Publicado el 5 de marzo de 2025. Creado por Gerard Beekmans.
- [6] Wikipedia, disponible en <https://en.wikipedia.org/wiki/Systemd>
- [7] https://fedoraproject.org/wiki/SysVinit_to_Systemd_Cheatsheet
- [8] <https://wiki.debian.org/systemd>
- [9] <https://docs.fedoraproject.org/en-US/quick-docs/systemd-understanding-and-administering/>
- [10] Gentoo Linux: <https://wiki.gentoo.org/wiki/Systemd>
- [11] <https://systemd.io/OPTIMIZATIONS/>
- [12] https://systemd.io/TIPS_AND_TRICKS/
- [13] https://docs.redhat.com/es/documentation/red_hat_enterprise_linux/8/html/configuring_basic_system_settings/managing-services-with-systemd_configuring-basic-system-settings#sect-Managing_Services_with_systemd-Introduction-Features
- [14] <https://juncotic.com/proceso-de-inicio-de-gnu-linux-systemd>
- [15] <https://eprint.iacr.org/2023/867.pdf>
- [16] Guia Completa para Administradores de Linux Systemd Journald
- [17] Arch Wiki systemd-nspawn
- [18] Arch Wiki systemd-boot
- [19] Debian systemd-slice
- [20] Arch Wiki systemd-resolved
- [21] Arch Wiki systemd-networkd
- [22] Arch Wiki systemd-creds
- [23] Atareao: trabajando con systemd
- [24] Atareao: gestionar servicios en Systemd
- [25] Atareao: Como crear un servicio con Systemd
- [26] Atareao: trabajando con Systemd, cron systemd timer
- [27] Reddit Controversial Systemd VS SysVinit
- [28] No a Systemd