

LINUX DevOps

```
or object to mirror  
mirror_mod.mirror_object
```

```
operation == "MIRROR_X":  
    mirror_mod.use_x = True  
    mirror_mod.use_y = False  
    mirror_mod.use_z = False  
    operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
    operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
selection at the end add  
mirror_ob.select=1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier  
mirror_ob.select = 0  
= bpy.context.selected_obj  
data.objects[one.name].select  
print("please select exactly
```

```
-- OPERATOR CLASSES --
```

```
types.Operator):  
    X mirror to the selected  
    object.mirror_mirror_x"  
    mirror X"
```

Automatizando con Shell Scripts

- Vamos a trabajar la **automatización de tareas de administración del sistema** mediante **scripts en Bash**.
- Para ello vamos a desarrollar un script en **shell Bash** que automatice la creación de una **copia de seguridad (dump)** de una base de datos, se abordará lo siguientes temas:
 - Realización de una copia de seguridad de una base de datos.
 - Comprensión de los scripts.
 - Comprensión de los comandos internos y la sintaxis de Bash.
 - Entendiendo los primeros pasos del script de copia de seguridad.
 - Manejo de errores y depuración (**debugging**).

Realización de una copia de seguridad de una base de datos

Las bases de datos más comunes, como **MySQL** y **PostgreSQL**, existen al menos **dos formas diferentes de realizar una copia de seguridad** de una base de datos:

1. Realizar un **volcado de base de datos** extrayendo todos los datos actuales junto con el esquema de la base de datos.
2. Copiar los **logs de replicación**.

- **En un entorno real** realizar una copia de seguridad completa puede ser **lento**, especialmente para bases de datos grandes.
- Mientras se ejecuta, la base de datos coloca un **bloqueo** en sus archivos de datos, lo que significa que no guarda nuevos datos en el disco; en su lugar, almacena todo en los **logs de replicación** hasta que el bloqueo es liberado.
- Para bases de datos grandes, esta operación puede tomar **horas**. Por esta razón:
 1. Realizaremos un **respaldo completo** una vez a la semana.
 2. Copiaremos todos los **logs de replicación** cada hora.
 3. Adicionalmente, crearemos una **instantánea diaria del disco** de nuestra instancia de base de datos en AWS.

- En entornos **en la nube**, también se puede crear una **instantánea (snapshot)** del disco donde se almacena la base de datos y la copia de seguridad.
- Un **volcado de base de datos** también puede ser utilizado como una copia de seguridad completa (**full backup**). Sin embargo, los logs de replicación no son volcados independientes, por lo que es necesario combinarlos con una copia de seguridad completa. Esto se llama **copia de seguridad incremental**.

Script de backup de las BBDD MySQL

1. Creamos el archivo que va a lanzar el script

```
touch ejecutar_backups.sh && chmod +x ejecutar_backups.sh  
ls -l ejecutar_backups.sh
```

2. Necesitaremos agregar un comando básico para realizar la copia de seguridad de la base de datos y continuar desde allí. Realizaremos la copia de seguridad de una base de datos MySQL utilizando la herramienta ***mysqldump***

Para hacer un backup de todas las bases de datos del servidor MySQL, usaremos el parámetro **–all-databases**

```
nano backup_completo.sql  
mysqldump --all-databases > backup_completo.sql|
```

```
nano backup_completo.sql
```

Vamos a mejorar este script...

Para hacer un backup de todas las bases de datos del servidor MySQL, usaremos el parámetro **-all-databases**

```
nano backup_completo.sql  
mysqldump --all-databases > backup_completo.sql
```

```
nano backup_completo.sql
```

Vamos a mejorar este script...

```
./ejecutar_backups.sh  
./ejecutar_backups.sh  
./ejecutar_backups.sh  
./ejecutar_backups.sh
```


La solución más común generar un nombre de archivo de respaldo diferente cada vez que se ejecuta la copia de seguridad. Primero, intentemos obtener una marca de tiempo con la fecha y hora local en el formato YYYYMMDD_HHMM

- **YYYY**: El año actual en un formato de cuatro dígitos.
- **MM**: El mes actual en un formato de dos dígitos.
- **DD**: El día del mes en un formato de dos dígitos.
- **HH**: La hora actual.
- **MM**: El minuto actual.

```
# date  
  
# man date
```

```
date +"%Y%m%d_%H%M"
```

- Para pasar la cadena de salida a una variable en nuestro script, necesitaremos ejecutar el comando date en un subshell (un proceso de Bash ejecutado por nuestro script de Bash):

```
fecha=$(date +"%Y%m%d_%H%M")
```

```
mysqldump --all-databases >backup_completo_`fecha`.sql
```

- 3. Lo podemos parametrizar más, con ruta del archivo de backup, base de datos concreta, usuario y contraseña

4. Manejo de errores en el script. El uso de set en el script nos va a ayudar a ver su funcionamiento. Para poder habilitar opciones de Bash para manejo de errores

- set -e → Detener el script si un comando devuelve un error
- set -u → Tratar variables no definidas como errores
- set -o pipefail → Detectar errores en cadenas de comandos con pipes

- Vamos a utilizar comandos en Linux que nos pueden valer cuando trabajamos con archivos... el find y el rm

Gestionar servicios en Linux

- Vamos a trabajar con los servicios (programas que se ejecutan en segundo plano como daemons) con más profundidad. Y nos centraremos en las unidades systemd (Unit).
- Tanto Mac OS X, Linux, Windows o FreeBSD, todos ellos ejecutan multitud de programas en segundo plano que, en conjunto, proporcionan un sistema útil. Lo mismo ocurre con los servidores de estos sistemas operativos.
- Un programa en segundo plano o proceso en segundo plano (en Unix y Linux se denomina, daemon) es un programa que no está vinculado a ninguna entrada (teclado, ratón, etc.) o salida (monitor, terminal, etc.).

- De este modo, pueden empezar a funcionar incluso cuando no hay nadie conectado al sistema y seguir funcionando cuando el usuario se desconecta.
- También pueden ejecutarse bajo los privilegios de un usuario que nunca puede iniciar sesión en el sistema, lo que hace que su ejecución sea mucho más segura.
- El número de servicios que se ejecutan en un sistema Linux dependerá, en gran parte, de la distribución y, en mayor parte aún, del propósito del sistema.

Historia de la gestión de servicios en Linux

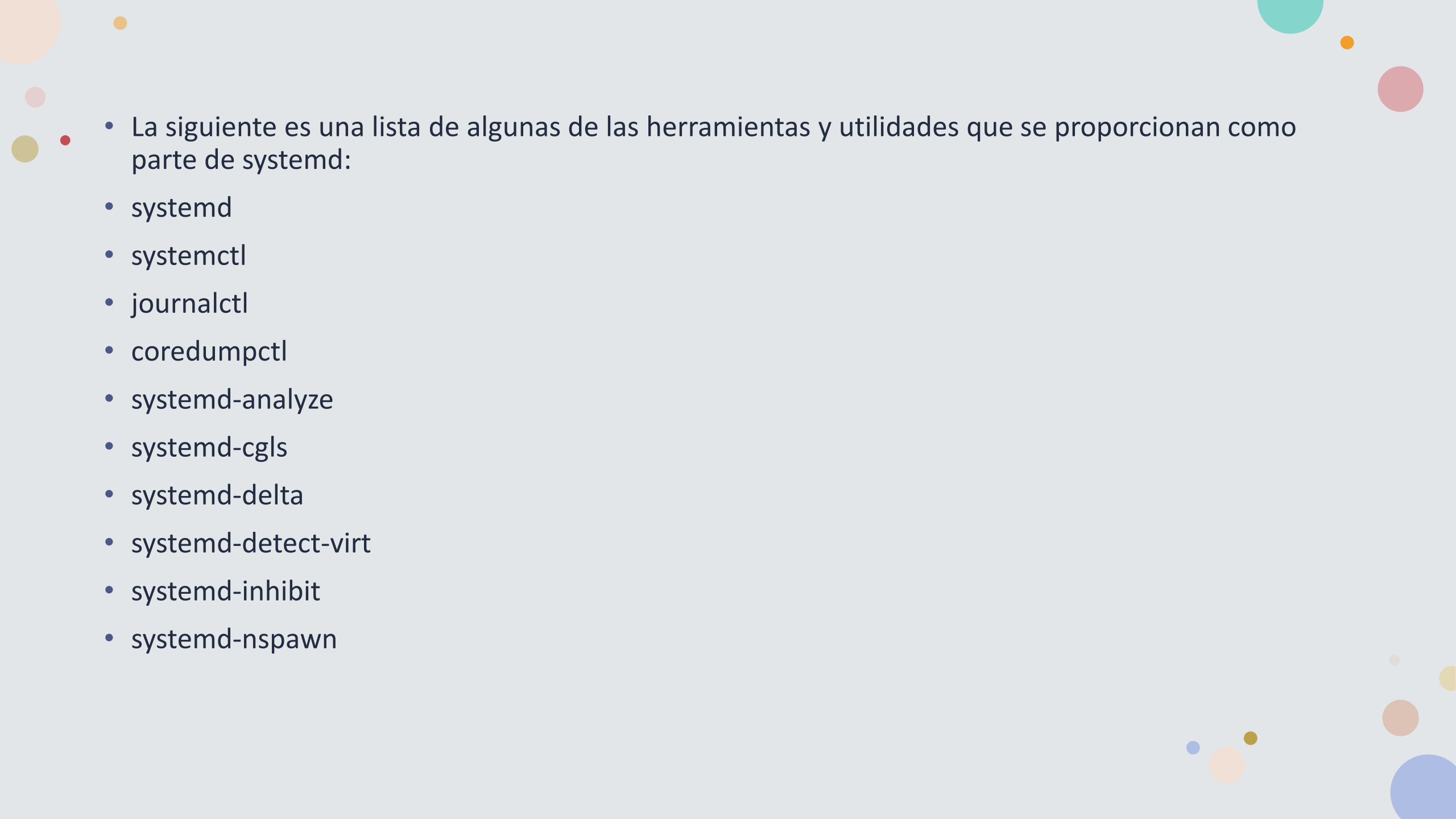
- La tarea de gestionar los servicios del sistema -los programas necesarios para que tu ordenador funcione- no es trivial.
- El software que los ejecuta debe ser estable y robusto.
- Esto, y el hecho de que se espera que el sistema se inicie raramente, especialmente en servidores, hizo que la solución adoptada por Linux sobreviviera durante décadas.
- La creciente necesidad de arrancar servicios en paralelo (a través de la aparición y el uso generalizado de muchas CPUs con hilos) y de forma más inteligente dio paso a varias implementaciones del nuevo sistema init

- systemd es un gestor de servicios para Linux capaz de administrar los servicios que se inician con el sistema operativo.
- Sustituye a los scripts init tradicionales.
- Es responsable de iniciar y detener los servicios del sistema, gestionar el estado del sistema y registrar los eventos del sistema.
- Se ha convertido en el sistema init por defecto de muchas distribuciones populares de Linux, como CentOS, Fedora Linux, Red Hat Enterprise Linux (RHEL) y Ubuntu Linux.

- Este gestor de servicios se encarga de controlar la inicialización del propio sistema (servicios necesarios para el SO Linux), iniciar y detener los servicios del sistema y gestionar los recursos del sistema.
- Proporciona otra forma de gestionar los servicios y otros componentes del sistema, y permite a los administradores del sistema configurar y personalizar el comportamiento de sus sistemas de una forma más estandarizada que con System V (SysV) init.

- Una de las características clave de systemd es su capacidad para iniciar servicios en paralelo, lo que puede reducir significativamente el tiempo de arranque de un sistema. También incluye una serie de herramientas para gestionar y supervisar los servicios del sistema.
- Otra cosa por la que se elogia a systemd es la uniformidad de las configuraciones de servicios que por fin ha traído al mundo Linux.
- En todas las distribuciones de Linux, los archivos de configuración de systemd se entregan en la misma ruta; tienen el mismo aspecto. Sin embargo, aún existen algunas diferencias dependiendo de la ruta en la que se instalen los binarios.
- systemd también es mejor a la hora de saber si un proceso se está ejecutando, lo que hace más difícil que nos encontremos en una situación en la que no podamos iniciar un proceso debido a archivos obsoletos.

- Una de las mayores ventajas de systemd es su conocimiento de las dependencias.
- La configuración del servicio (programa en ejecución bajo el control de systemd) contiene información sobre todos los demás servicios de los que depende y también puede apuntar a servicios que dependen de él.
- Lo que, es más, el servicio puede informar a systemd sobre qué objetivos necesita para ejecutarse: si su servicio necesita que la red esté en funcionamiento, puede poner esta información en la configuración, y systemd se asegurará de que su daemon se inicie sólo después de que la red esté configurada correctamente.



- La siguiente es una lista de algunas de las herramientas y utilidades que se proporcionan como parte de systemd:

- `systemd`
- `systemctl`
- `journalctl`
- `coredumpctl`
- `systemd-analyze`
- `systemd-cgls`
- `systemd-delta`
- `systemd-detect-virt`
- `systemd-inhibit`
- `systemd-nspawn`

- **systemd**: Es el principal gestor de sistemas y servicios. Es el programa principal que controla la inicialización y gestión de servicios y otros componentes del sistema.
- **systemctl**: Es la utilidad de línea de comandos para gestionar los servicios del sistema y otros componentes del sistema. Se puede utilizar para iniciar, detener, reiniciar, activar y desactivar servicios, así como para ver el estado de los servicios y otros componentes del sistema.
- **journalctl**: Se utiliza para ver y manipular el registro del sistema, gestionado por systemd. Se puede utilizar para ver los mensajes de registro, filtrar los mensajes de registro en función de diversos criterios y exportar los datos de registro a un archivo.
- **coredumpctl**: Como su nombre indica, se trata de una utilidad que ayuda a recuperar los volcados de memoria del diario de systemd.
- **systemd-analyze**: Se puede utilizar para analizar el rendimiento de arranque de un sistema. Mide el tiempo que tarda un sistema en arrancar, así como el tiempo que tarda en identificar posibles cuellos de botella y problemas de rendimiento.

- **systemd-cgls:** Es una utilidad de línea de comandos para ver la jerarquía de grupos de control de un sistema. Los grupos de control, o cgroups, son utilizados por systemd para gestionar los recursos del sistema y aislar los procesos entre sí.
- **systemd-delta:** Esta es una utilidad de línea de comandos para analizar las diferencias entre los archivos de configuración predeterminados proporcionados por systemd y cualquier modificación local realizada en estos archivos.
- **systemd-detect-virt:** Esta es una utilidad de línea de comandos para detectar el entorno de virtualización en el que se está ejecutando un sistema. Se puede utilizar para determinar si un sistema se está ejecutando en una máquina virtual (VM), un contenedor, o en metal desnudo.
- **systemd-inhibit:** Es una utilidad de línea de comandos para evitar que se realicen determinadas acciones del sistema, como suspender o apagar el sistema.
- **systemd-nspawn:** Es una utilidad de línea de comandos para ejecutar procesos en contenedores ligeros. Puede utilizarse para crear y gestionar contenedores, así como para ejecutar procesos dentro de contenedores.

Targets

- En systemd, un target es un estado específico en el que puede estar el sistema, y está representado por un nombre simbólico. Los targets se utilizan para definir el comportamiento de alto nivel del sistema, y normalmente se utilizan para agrupar un conjunto de servicios relacionados y otros componentes del sistema.
- Por ejemplo, `multiusuario.target` es un target que representa un sistema preparado para proporcionar acceso multiusuario, con red y otros servicios habilitados; `gráfico.target` es un target que representa un sistema preparado para mostrar una pantalla gráfica de inicio de sesión, con un entorno gráfico de escritorio y servicios relacionados habilitados.

- Los «Targets» se definen típicamente en archivos de unidad, que son archivos de configuración que describen las propiedades y el comportamiento de los componentes del sistema. Cuando se activa un objetivo, systemd iniciará todos los servicios y otros componentes del sistema que estén asociados con ese objetivo.
- systemd incluye una serie de objetivos predefinidos que cubren una amplia gama de estados comunes del sistema, y los administradores también pueden definir objetivos personalizados para adaptarse a las necesidades específicas de sus sistemas. Los objetivos pueden activarse mediante el comando `systemctl` o modificando el objetivo predeterminado que se establece al arrancar el sistema.

- Estos son algunos ejemplos de objetivos predefinidos de systemd:
- `poweroff.target`: Representa un sistema que se está apagando o desconectando
- `rescue.target`: Representa un sistema que se está ejecutando en modo de rescate, con servicios mínimos habilitados
- `objetivo.multiusuario`: Representa un sistema que está listo para proporcionar acceso multiusuario, con red y otros servicios habilitados.
- `objetivo.gráfico`: Representa un sistema que está listo para mostrar una pantalla gráfica de inicio de sesión, con un entorno de escritorio gráfico y servicios relacionados habilitados.
- `reboot.target`: Representa un sistema que se está reiniciando
- `destino.emergencia`: Representa un sistema que se está ejecutando en modo de emergencia, con sólo los servicios más esenciales habilitados

- Este es un ejemplo de un archivo de unidad systemd que define un objetivo personalizado:

```
[Unit]
Description=Unit File With Custom Target
[Install]
WantedBy=multi-user.target
```

- Este archivo de unidad define un objetivo llamado Custom Target que debe activarse como parte de multi-user.target.
- La directiva WantedBy especifica que el objetivo debe activarse cuando se active multi-user.target.

- Aquí hay otro ejemplo de un archivo de unidad systemd llamado custom.target que define un objetivo personalizado:

```
[Unit]
Description=My simple service
[Install]
WantedBy=multi-user.target
```

```
[Service]
ExecStart=/usr/local/bin/my-simple-service
Type=simple
[Install]
WantedBy=custom.target
```

```
[Service]
ExecStart=/usr/local/bin/my-simple-service
Type=simple
[Install]
WantedBy=custom.target
```

- Este archivo de unidad define un destino llamado Archivo de unidad con destino personalizado y un servicio llamado Mi servicio simple. La directiva ExecStart especifica el comando que debe utilizarse para iniciar el servicio, y la directiva Type especifica el tipo de servicio. La directiva WantedBy en la sección [Install] de la unidad de servicio especifica que el servicio debe activarse cuando se active custom.target.
- Ahora, ya que hemos tocado un poco el tema de los archivos de unidad, vamos a profundizar en ellos y ver qué se puede hacer con ellos.

Ficheros Unit

- Los ficheros Unit se almacenan normalmente en el directorio del sistema `/lib/systemd/` de los sistemas de ficheros del SO Linux. Ningún archivo de este directorio debe ser alterado de ninguna manera, ya que serán reemplazados por archivos contenidos en paquetes cuando se actualice un servicio utilizando un gestor de paquetes.
- En su lugar, para modificar un archivo de unidad de un servicio específico, cree su archivo de unidad personalizado dentro del directorio `/etc/systemd/system`. Los archivos en este directorio etc tendrán prioridad sobre la ubicación por defecto.

- systemd es capaz de proporcionar activación de unidad utilizando lo siguiente:
 - Dependencia: Arrancará una unidad simplemente usando un árbol de dependencias pre-construido manejado por systemd. En su forma más simple, necesitará añadir una dependencia con `multi-user.target` o con `network.target` si su servicio está usando redes.
 - Archivos de unidad drop-in: Puede ampliar fácilmente el archivo de unidad predeterminado proporcionando un fragmento que anulará una parte del archivo de unidad predeterminado (por ejemplo, cambiar las opciones de ejecución del demonio). Estos son los archivos que colocarías dentro del directorio `/etc/systemd/system`.
 - Plantillas: También puedes definir archivos de unidad de plantilla. Estas unidades especiales pueden usarse para crear múltiples instancias de la misma unidad general.

- Funciones de seguridad: Puedes especificar acceso de sólo lectura o denegar el acceso a una parte del sistema de archivos, asignar acceso privado/tmp o de red y limitar las capacidades del kernel.
- Ruta: Puede iniciar una unidad basándose en la actividad o la disponibilidad de un archivo o directorio en el sistema de archivos.
- Socket: Se trata de un tipo especial de archivo en el sistema operativo Linux que permite la comunicación entre dos procesos. Mediante esta función, puede retrasar el inicio de un servicio hasta que se acceda al socket asociado. También puede crear un archivo de unidad que simplemente cree un socket al principio de un proceso de arranque y un archivo de unidad separado que utilice este socket.
- Bus: Puedes activar una unidad utilizando la interfaz de bus proporcionada por D-Bus. D-Bus es simplemente un bus de mensajes utilizado para la comunicación entre procesos (IPC), más comúnmente utilizado en GUIs como GNOME o KDE.
- Dispositivo: También puede iniciar una unidad en la disponibilidad de hardware (el archivo dev, ubicado en el directorio /dev). Esto aprovechará un mecanismo conocido como udev, que es un subsistema de Linux que suministra eventos de dispositivo.
- Una vez que haya iniciado un servicio, probablemente querrá comprobar si está funcionando correctamente echando un vistazo a los archivos de registro. De esta tarea se encarga journald.

Logging

- Cada servicio gestionado por systemd envía sus registros a journal, una parte especial de systemd. Hay una herramienta especial de línea de comandos para gestionar esos registros: journalctl. En su forma más simple, la ejecución de un comando journalctl simplemente mostrará todos los registros del sistema, con los más recientes en la parte superior. Aunque el formato de los registros es similar al de syslog -la herramienta tradicional para recopilar registros en Linux- journalctl captura más datos. Recoge datos del proceso de arranque, registros del kernel y más.
- Los registros de arranque son transitorios por defecto. Esto significa que no se guardan entre reinicios del sistema. Sin embargo, existe la posibilidad de grabarlos permanentemente. Hay dos formas de hacerlo, como se indica a continuación:

- Cree un directorio especial. Cuando journald lo detecte durante el arranque del sistema, guardará los registros allí: `sudo mkdir -p /var/log/journal`.
- Edite el archivo de configuración de journald y habilite los registros de arranque persistentes. Abra el archivo `/etc/systemd/journald.conf` con su editor favorito y, en la sección `[Journal]`, edite la opción `Storage` para que tenga este aspecto:

```
[Journal]
```

```
Storage=persistent
```

- journald es capaz de filtrar los registros por servicio mediante el uso de la opción -u nombre.servicio-es decir, journalctl -u httpd.servicio sólo imprimirá los registros del demonio httpd.
- Existe la posibilidad de imprimir registros de un periodo de tiempo determinado, imprimir desde más de un servicio, buscar registros por ID de proceso (PID), y más.