TUTORIAL

Cómo usar Systemctl para gestionar servicios y unidades de Systemd

System Tools

By Justin Ellingwood

Published on December 2, 2020 11.6k

Introducción

`systemd` es un sistema init y un administrador del sistema que se ha convertido en el nuevo estándar para las distribuciones Linux. Debido a su gran adopción, merece la pena familiarizarse con `systemd`, ya que hará que administrar servidores sea mucho más fácil. Conocer y utilizar las herramientas y daemons que componen `systemd` le ayudará a apreciar mejor la potencia, la flexibilidad y las capacidades que proporciona, o al menos a simplificar su trabajo.

En esta guía, hablaremos del comando `systemctl`, que es la herramienta de administración central para controlar el sistema init. Explicaremos cómo administrar servicios, comprobar estados, cambiar estados del sistema y trabajar con los archivos de configuración.

Tenga en cuenta que aunque `systemd` es el sistema init predeterminado para muchas distribuciones Linux, no se implementa universalmente en todas las distribuciones. A medida que avanza en este tutorial, si su terminal arroja el error `bash: systemctl is not installed`,

cualquier momento mientras se ejecuta el sistema. Teniendo eso en cuenta, comenzaremos con algunas operaciones básicas de administración de servicio.

En `systemd`, el destino de la mayoría de las acciones son "unidades", que son recursos que `systemd` sabe cómo administrar. Las unidades se categorizan por el tipo de recurso al que representan y se definen con archivos conocidos como archivos de unidad. El tipo de cada unidad puede deducirse del sufijo al final del archivo.

Para las tareas de administración de servicio, la unidad de destino será unidades de servicio, que tienen archivos de unidad con un sufijo `.service`. Sin embargo, para la mayoría de los comandos de administración de servicio, puede dejar fuera el sufijo `.service`, ya que `systemd` es lo suficientemente inteligente para saber que probablemente quiere operar sobre un servicio cuando utiliza comandos de administración de servicio.

Iniciar y detener servicios

Para iniciar un servicio `systemd`, ejecutar instrucciones en el archivo de la unidad del servicio, utilice el comando `start`. Si está ejecutando como usuario non-root, tendrá que usar `sudo`, ya que esto afectará al estado del sistema operativo.

```
$ sudo systemctl start application.service
```

Como hemos mencionado antes, `systemd` sabe buscar los archivos `*.service` para los comandos de administración de servicio, de forma que el comando podría escribirse fácilmente así:

```
$ sudo systemctl start application
```

Aunque puede usar el formato anterior para la administración general, para mayor claridad, usaremos el sufijo `.service` para el resto de los comandos, con el objetivo de ser explícitos

Reiniciar y volver a cargar

Para reiniciar un servicio en ejecución, puede usar el comando `restart`:

```
$ sudo systemctl restart application.service
```

Si la aplicación en cuestión puede volver a cargar sus archivos de configuración (sin reiniciar), puede emitir el comando `reload` para iniciar ese proceso:

```
$ sudo systemctl reload application.service
```

Si no está seguro de si el servicio tiene la funcionalidad de volver a cargar su configuración, puede emitir el comando `reload-or-restart`. Esto volverá a cargar la configuración en vigor, si está disponible. De lo contrario, reiniciará el servicio de forma que se recoja la nueva configuración:

```
$ sudo systemctl reload-or-restart application.service
```

Cómo habilitar y deshabilitar servicios

Los comandos anteriores son útiles para iniciar o detener servicios durante la sesión actual. Para indicar a `systemd` que inicie servicios automáticamente en el arranque, debe habilitarlos.

Para iniciar un servicio en el arranque, utilice el comando `enable`:

```
$ sudo systemctl enable application.service
```

Esto creará un enlace simbólico desde la copia del sistema del archivo de servicio

```
$ sudo systemctl disable application.service
```

Esto eliminará el enlace simbólico que indicaba que el servicio debía iniciarse automáticamente.

Tenga en cuenta que habilitar el servicio no lo inicia en la sesión actual. Si desea iniciar el servicio y habilitarlo en el arranque, tendrá que emitir los comandos `start` y `enable`.

Cómo comprobar el estado de los servicios

Para comprobar el estado de un servicio en su sistema, puede usar el comando `status`:

```
$ systemctl status application.service
```

Esto le proporcionará el estado del servicio, la jerarquía de cgroup y las primeras líneas de registro.

Por ejemplo, cuando se comprueba el estado de un servidor Nginx, puede ver un resultado como este:

```
Output
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2015-01-27 19:41:23 EST; 22h ago
 Main PID: 495 (nginx)
   CGroup: /system.slice/nginx.service
           ├─495 nginx: master process /usr/bin/nginx -g pid /run/nginx.pid; error_log stderr;
           └─496 nginx: worker process
Jan 27 19:41:23 desktop systemd[1]: Starting A high performance web server and a reverse proxy
Jan 27 19:41:23 desktop systemd[1]: Started A high performance web server and a reverse proxy
```

```
$ systemctl is-active application.service
```

Esto devolverá el estado actual de la unidad, que es normalmente `activo` o `inactivo`. El código de salida será “0” si está activo, lo que hace que el resultado sea más sencillo de analizar en las secuencias de comando shell.

Para ver si la unidad está habilitada, puede usar el comando `is-enabled`:

```
$ systemctl is-enabled application.service
```

Esto indicará si el servicio está `habilitado` o `deshabilitado` y establecerá el código de salida a “0” o “1”, dependiendo de la respuesta a la pregunta del comando.

Una tercera comprobación es si la unidad está en estado fallido. Esto indica que hubo un problema al iniciar la unidad en cuestión:

```
$ systemctl is-failed application.service
```

Esto devolverá `active` si se está ejecutando adecuadamente o `failed` si se ha producido un error. Si la unidad se detuvo intencionadamente, puede devolver `unknown` o `inactive`. Un estado de salida de “0” indica que se produjo un error y un estado de salida de “1” indica cualquier otro estado.

Descripción general del estado del sistema

Los comandos hasta ahora han sido útiles para administrar servicios individuales, pero no son muy útiles para explorar el estado actual del sistema. Hay varios comandos `systemctl` que proporcionan esta información.

Cómo enumerar las unidades actuales

Esto le mostrará una lista de todas las unidades que `systemd` tiene activas actualmente en el sistema. El resultado tendrá un aspecto similar a este:

```
Output
UNIT                                      LOAD   ACTIVE SUB     DESCRIPTION
atd.service                               loaded active running ATD daemon
avahi-daemon.service                      loaded active running Avahi mDNS/DNS-SD Stack
dbus.service                              loaded active running D-Bus System Message Bus
dcron.service                             loaded active running Periodic Command Scheduler
dkms.service                              loaded active exited  Dynamic Kernel Modules System
getty@tty1.service                        loaded active running Getty on tty1
. . .
```

El resultado tiene las siguientes columnas:

- **UNIT**: El nombre de la unidad de `systemd`
- **LOAD**: Si la configuración de la unidad ha sido analizada por `systemd`. La configuración de las unidades cargadas se mantiene en la memoria.
- **ACTIVE**: Un estado resumido que indica si la unidad está activa. Esta es normalmente una forma bastante básica de saber si la unidad se ha iniciado correctamente o no.
- **SUB**: Este es un estado de nivel inferior que indica información más detallada sobre la unidad. Esto a menudo varía por tipo de unidad, estado y el método real en el que se ejecuta la unidad.
- **DESCRIPTION**: Una descripción textual breve de qué es y hace la unidad.

Ya que el comando `list-units` muestra solo las unidades activas por defecto, todas las entradas por encima se mostrarán `loaded` en la columna LOAD y `active` en la columna ACTIVE. Esta pantalla es, en realidad, el comportamiento predeterminado de `systemctl` cuando se invoca sin comandos adicionales, de modo que verá lo mismo si invoca `systemctl` sin argumentos:

cargar), independientemente de si están activas actualmente, puede usar el marcador `--all`, de esta forma:

```
$ systemctl list-units --all
```

Esto mostrará cualquier unidad que `systemd` haya cargado o intentado cargar, independientemente de su estado actual en el sistema. Ciertas unidades se vuelven inactivas tras ejecutarse, y algunas de las unidades que `systemd` intentó cargar pueden no haberse encontrado en el disco.

Puede usar otros marcadores para filtrar estos resultados. Por ejemplo, puede usar el indicador `--state=` para indicar los estados LOAD, ACTIVE o SUB que deseamos ver. Tendremos que mantener el marcador `--all` para que `systemctl` permita que se muestren las unidades no activas:

```
$ systemctl list-units --all --state=inactive
```

Copy

Otro filtro común es el filtro `--type=`. Podemos indicar a `systemctl` que solo muestre unidades del tipo en el que estemos interesados. Por ejemplo, para ver únicamente las unidades de servicio activas, podemos usar:

```
$ systemctl list-units --type=service
```

Listar todos los archivos de la unidad

El comando `list-units` solo muestra las unidades que `systemd` ha intentado analizar y cargar en la memoria. Ya que `systemd` solo leerá unidades que cree que necesita, esto no incluirá necesariamente todas las unidades disponibles en el sistema. Para ver *todos* los archivos de unidad disponibles en las rutas `systemd`, incluidos aquellos que `systemd` no

presente información sobre los propios archivos. El resultado tiene dos columnas, el archivo de la unidad y el estado.

```
Output
UNIT FILE                                 STATE
proc-sys-fs-binfmt_misc.automount         static
dev-hugepages.mount                       static
dev-mqueue.mount                          static
proc-fs-nfsd.mount                        static
proc-sys-fs-binfmt_misc.mount             static
sys-fs-fuse-connections.mount             static
sys-kernel-config.mount                   static
sys-kernel-debug.mount                    static
tmp.mount                                 static
var-lib-nfs-rpc_pipefs.mount              static
org.cups.cupsd.path                       enabled
. . .
```

El estado normalmente estará `habilitado`, `deshabilitado`, `estático` o `enmascarado`. En este contexto, "estático" significa que el archivo de unidad no contiene una sección `install`, que se utiliza para habilitar una unidad. Como tal, estas unidades no pueden habilitarse. Normalmente, esto significa que la unidad realiza una única acción o se utiliza solo como dependencia de otra unidad y no debería ejecutarse por sí misma.

En breve explicaremos lo que significa `enmascarado`.

Gestión de la unidad

Hasta ahora, hemos estado trabajando con servicios y mostrando información sobre la unidad y los archivos de la unidad que `systemd` conoce. Sin embargo, encontraremos más información específica sobre las unidades usando algunos comandos adicionales.

Mostrar un archivo de unidad

Output

```
[Unit]
Description=ATD daemon
[Service]
Type=forking
ExecStart=/usr/bin/atd
[Install]
WantedBy=multi-user.target
```

El resultado es el archivo de unidad tal como lo conoce el proceso `systemd` que se está ejecutando actualmente. Esto puede ser importante si ha modificado archivos de unidad recientemente o si está omitiendo ciertas opciones en un fragmento del archivo de unidad (hablaremos de esto más tarde).

Mostrar dependencias

Para ver el árbol de dependencias de una unidad, puede usar el comando `list-dependencies`:

```
$ systemctl list-dependencies sshd.service
```

Esto mostrará una jerarquía asignando las dependencias que deben tratarse para iniciar la unidad en cuestión. Las dependencias, en este contexto, incluyen las unidades que son necesarias o deseadas por unidades de nivel superior.

Output

```
sshd.service
├─system.slice
└─basic.target
  ├─microcode.service
  ├─rhel-autorelabel-mark.service
```

Las dependencias recursivas solo se muestran para las unidades `.target`, que indican los estados del sistema. Para listar de forma recursiva todas las dependencias, incluya el indicador `--all`.

Para mostrar las dependencias inversas (unidades que dependen de la unidad especificada) puede añadir el indicador `--reverse` al comando. Otros indicadores que son útiles son los indicadores `--before` y `--after`, que pueden usarse para mostrar las unidades que dependen de la unidad especificada que comienza antes y después de ellas mismas respectivamente.

Comprobar las propiedades de la unidad

Para ver las propiedades de nivel bajo de una unidad, puede usar el comando `show`. Esto mostrará una lista de propiedades que se establecen para la unidad especificada usando un formato `key=value`:

```
$ systemctl show sshd.service
```

```
Output
Id=sshd.service
Names=sshd.service
Requires=basic.target
Wants=system.slice
WantedBy=multi-user.target
Conflicts=shutdown.target
Before=shutdown.target multi-user.target
After=syslog.target network.target auditd.service systemd-journald.socket basic.target system.
Description=OpenSSH server daemon
. . .
```

Si desea mostrar una única propiedad, pude pasar el indicador `-p` con el nombre de la

Output

```
Conflicts=shutdown.target
```

Enmascarar y desenmascarar unidades

Vimos en la sección de administración del servicio cómo detener o deshabilitar un servicio, pero `systemd` también tiene la capacidad de marcar una unidad como *completamente* no iniciable, automática o manualmente, vinculándola a `/dev/null`. Esto se denomina enmascarar la unidad, y es posible con el comando `mask`:

```
$ sudo systemctl mask nginx.service
```

Esto impedirá que el servicio Nginx se inicie, automática o manualmente, siempre que esté enmascarado.

Si comprueba los `list-unit-files`, verá que el servicio ahora se lista como enmascarado:

```
$ systemctl list-unit-files
```

Output

```
. . .
kmod-static-nodes.service             static
ldconfig.service                      static
mandb.service                         static
messagebus.service                    static
nginx.service                         masked
quotaon.service                       static
rc-local.service                      static
rdisc.service                         disabled
rescue.service                        static
. . .
```

```
Output
Failed to start nginx.service: Unit nginx.service is masked.
```

Para desenmascarar una unidad, y hacer que esté disponible de nuevo para su uso, utilice el comando `unmask`:

```
$ sudo systemctl unmask nginx.service
```

Esto devolverá la unidad a su estado anterior, permitiendo que se inicie o habilite.

Editar archivos de la unidad

Aunque el formato específico de los archivos de unidad está fuera del alcance de este tutorial, si necesita realizar ajustes, `systemctl` proporciona mecanismos integrados para editar y modificar archivos de unidad. Esta funcionalidad fue añadida en la versión 218 de `systemd`.

El comando `edit`, por defecto, abrirá un fragmento de código del archivo de la unidad para la unidad en cuestión:

```
$ sudo systemctl edit nginx.service
```

Este será un archivo en blanco que puede usarse para omitir o añadir directivas a la definición de la unidad. Se creará un directorio en el directorio `/etc/systemd/system` que contiene el nombre de la unidad con `.d` anexada. Por ejemplo, para el `nginx.service`, se creará un directorio llamado `nginx.service.d`.

En este directorio, se creará un fragmento de cód`igo llamado o` verride.conf. Cuando se carga la unidad, `systemd`, en la memoria, fusionará el fragmento de código de anulación con el archivo de unidad completo. Las directivas del snippet prevalecerán sobre las

Esto cargará el archivo de unidad actual en el editor, donde se podrá modificar. Cuando sale el editor, el archivo cambiado se escribirá a `/etc/systemd/system`, que tendrá prioridad sobre la definición de la unidad del sistema (normalmente se encuentra en algún lugar de `/lib/systemd/system`).

Para eliminar cualquier adición que haya realizado, elimine el directorio de configuración `.d` de la unidad o el archivo de servicio modificado de `/etc/systemd/system`. Por ejemplo, para eliminar un fragmento de código, podríamos escribir lo siguiente:

```
$ sudo rm -r /etc/systemd/system/nginx.service.d
```

Para eliminar un archivo de unidad completo modificado, escribiríamos:

```
$ sudo rm /etc/systemd/system/nginx.service
```

Tras eliminar el archivo o directorio, debería volver a cargar el proceso `systemd` de forma que deje de intentar hacer referencia a estos archivos y vuelve a usar las copias del sistema. Puede hacerlo escribiendo lo siguiente:

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

Ajustar el estado del sistema (Runlevel) con los destinos

Los destinos son archivos de unidad especiales que describen el estado de un sistema o un punto de sincronización. Igual que otras unidades, los archivos que definen los destinos pueden identificarse por su sufijo, que en este caso es `.target`. Los destinos no hacen mucho por sí mismos, pero se utilizan para agrupar otras unidades.

Esto puede usarse para llevar al sistema a ciertos estados, de forma muy similar que otros sistemas init utilizan runlevels. Se utilizan como referencia para cuando ciertas funciones

unidades que requieren que swap esté disponible pueden especificar esta condición usando las especificaciones `Wants=`, `Requires=` y `After=` para indicar la naturaleza de su relación.

Obtener y establecer el destino predeterminado

El proceso `systemd` tiene un destino predeterminado que utiliza cuando se inicia el sistema. Satisfacer la cascada de dependencias de ese destino individual llevará al sistema al estado deseado. Para encontrar el destino predeterminado de su sistema, escriba:

```
$ systemctl get-default
```

```
Output
multi-user.target
```

Si desea establecer un destino predeterminado diferente, puede usar `set-default`. Por ejemplo, si tiene un escritorio gráfico instalado y desea que el sistema se inicie a esto por defecto, puede cambiar el destino predeterminado:

```
$ sudo systemctl set-default graphical.target
```

Cómo enumerar los destinos disponibles

Puede obtener una lista de los destinos disponibles en su sistema escribiendo:

```
$ systemctl list-unit-files --type=target
```

A diferencia de runlevels, puede haber múltiples destinos activos a la vez. Un destino activo indica que `systemd` ha intentado iniciar todas las unidades relacionadas con el destino y no ha intentado desglosarlas de nuevo. Para ver todos los destinos activos, escriba:

denomina, apropiadamente, `isolate`. Esto es similar a cambiar el runlevel en otros sistemas init.

Por ejemplo, si está operando en un entorno gráfico con `graphical.target` activo, puede apagar el sistema gráfico y poner el sistema en un estado de línea de comando multi usuario asilando el `multi-user.target`. Ya que `graphical.target` depende de `multi-user.target` pero no al revés, todas las unidades gráficas se detendrán.

Quizá desee echar un vistazo a las dependencias del destino que está aislando antes de realizar este procedimiento para asegurarse de que no está deteniendo servicios cruciales:

```
$ systemctl list-dependencies multi-user.target
```

Cuando esté satisfecho con las unidades que se mantendrán activas, puede aislar el destino escribiendo lo siguiente:

```
$ sudo systemctl isolate multi-user.target
```

Cómo usar atajos para eventos importantes

Existen destinos definidos para eventos importantes como apagar o reiniciar. Sin embargo, `systemctl` también tiene algunos atajos que añaden ciertas funciones adicionales.

Por ejemplo, para poner el sistema en el modo rescate (usuario único), puede usar simplemente el comando `rescue` en vez de `isolate rescue.target`,

```
$ sudo systemctl rescue
```

Esto proporcionará la funcionalidad adicional de alertar a todos los usuarios con sesión iniciada sobre el evento

```
$ sudo systemctl poweroff
```

Puede iniciar un reinicio con el comando `reboot`:

```
$ sudo systemctl reboot
```

Todos estos comandos alertan a los usuarios con sesión iniciada de que se va a producir el evento, algo que ejecutar o aislar el destino únicamente no hará. Observe que la mayoría de los equipos vincularán los comandos más cortos y convencionales para estas operaciones de forma que funcionen adecuadamente con `systemd`.

Por ejemplo, para reiniciar el sistema, normalmente puede escribir:

```
$ sudo reboot
```

Conclusión

Ahora debería estar familiarizado con algunas de las capacidades básicas del comando `systemctl` que le permite controlar e interactuar con su instancia de `systemd`. La utilidad `systemctl` será su punto de interacción principal para la administración del estado del servicio y del sistema.

Aunque `systemctl` opera principalmente con el proceso central `systemd`, hay otros componentes para el ecosistema de `systemd` que están controlados por otras utilidades. Otras capacidades, como la administración de registros y las sesiones de usuario se administran mediante daemons y utilidades de administración independientes (`journald`/`journalctl` y `logind`/`loginctl`, respectivamente). Al tomarse el tiempo para familiarizarse con estas herramientas y daemons, la administración le resultará más sencilla.

Report an issue

About the authors

Justin Ellingwood

Senior Technical Writer
@DigitalOcean

Still looking for an answer?

Ask a question

Search for more help

RELATED

DOCTL: DigitalOcean CLI
Tool

Comments

0 Comments

Leave a comment...

Sign In to Comment

Learn More

Company

About
Leadership
Blog
Careers
Partners
Referral Program
Press
Legal
Security & Trust Center

Products

Pricing
Products Overview
Droplets
Kubernetes

Community

Tutorials
Q&A
Tools and Integrations
Tags

Contact

Get Support
Trouble Signing In?
Sales
Report Abuse