systemd y amigos

by javier - Sábado, mayo 23, 2015

http://memoriasdeunsysadmin.tk/2015/05/23/systemd-y-amigos/

Cuando salió por fin Ubuntu 15.x, yo venía trabando en kubuntu 14.10. Ya pensando en pasarme por fin a LTS(aunque realmente, Utopic Unicorn anduvo excelente) comencé a averiguar acerca de las experiencias de los demás. Sobre todo, por la polémica y diversidad de opiniones generadas por systemd. Aún no puedo concluir nada sobre el tema, ya que no soy un experto. Sin embargo, tanto los a favor, como los en contra del sistema de manejo de servicios, tenían razones válidas. He leído notas de diversas fuentes y bueno, decidí que lo probaría. Tengo que ser sincero: No había hecho un upgrade así antes. Normalmente, no me duraban mucho las distros. Históricamente, no solía quedarme con una misma distro por mucho tiempo, sino más bien probaba distintas cuando me cansaba de una(o bien cuando comenzaba a tener problemas). La realidad es que ahora he aprendido a resolver varios de ellos, pero aún así, mi experiencia con Ubuntu ha sido muy grata.

Resumiendo, este era mi primer upgrade y sinceramente, otra cosa a favor de Ubuntu, fue sencillísimo y sin ninguna complicación. Por el momento, debo decir que la versión es más que estable y systemd no me ha dado problemas de ningún tipo. Con este background, decidí interiorizarme un poco más en el tema.

Históricamente han habido varios sistemas de inicialización y vamos a ahondar un poco.

El INIT o sistema de inicialización:

Cuál es el rol del sistema de inicialización? Administrar el inicio y detención de cada servicio y de las sesiones. En linux, es necesario un sistema o proceso que maneje el inicio de los demás procesos vitales, como los demonios de red, controladores, demonio de impresión y otros tantos. El sistema de inicialización, o Init, es el que se encarga de iniciar/detener los servicios según corresponda. También se encarga de administrar sesiones. Init es iniciado por el kernel mismo, que da al proceso init un PID = 1. De este modo, todos los demás procesos dependen de init y este es de algún modo, el proceso "padre".

rc:

Uno de los sistemas Init más importantes, es rc, que se encuentra presente en FreeBSD y Slackware, entre otros. En los sistemas que lo usan, el kernel llama /sbin/init y esto genera una consola que dispara /etc/rc. Este es un script que chequea los discos duros y los monta, para luego iniciar otros procesos básicos y finalmente iniciar también el susbistemas de redes. 4.4 BSD es un sistema muy simple y todo su funcionamiento radica en algunos pocos scripts, como /etc/rc, /etc/rc.local y /etc/netstart. No tiene un orden de apagado. Al recibir la señal SIGTERM(apagado, Init se encarga de apagar sus sub-procesos y finalmente, apagará el equipo.ntre ellos, SystemV ha tenido una larga vida. Default de varias de las Distros "mayores", entre ellas Debian y derivados(es por ello que es el que más he utilizado).

System V:

System V proviene de SunOS y Solaris y ha sido adoptado por muchas de las Distros "Mayores". Éste se basa en modos(niveles) en los que el Sistema Operativo puede estar. Para los que no saben mucho de linux, sin ser muy técnico, les comento que estos niveles (llamados runlevels) que, para hacerlo corto y sencillo, son modos en los que puede entrar el sistema operativo. Linux tiene 6 runlevels, siendo 0 el modo halt(apagado), 3-5 modos gráficos(el usuario inicia sesión en un entorno de escritorio), 6 restart(el sistema operativo se reinicia. Obviamente, entrar a cada uno de estos modos, implica el inicio o detención de los servicios correspondientes. Por ejemplo, al entrar en el modo 0, todos los servicios son detenidos.

En sistemas usando Sys V, el kernel ejecuta /sbin/init, el cual cargará parámetros y ejecutará directivas definidias en el archivo /etc/inittab. Inittab define comportamientos como, por ejemplo, el runlevel por defecto, las acciones disparadas por ciertas combinaciones de teclado, mapas de teclado y demás configuraciones de la terminal. También se define aquí el orden de ejecución de otros scripts importantes.

Finalmente, ejecuta /etc/init.d/rcS, el cual lleva al sistema al modo "single-user"(monousuario), donde se llevan a cabo tareas críticas, como el chequeo de hardware, montado de discos, inicio del sistema de redes y demás. Por último, el sistema será llevado al runlevel "por defecto" y comenzará con el arranque de los servicios básicos que deben iniciar con linux.

upstart:

Este es un sistema que incorpora varias mejoras sobre SysV. Aunque no utiliza código del primero, provee una alta compatibilidad hacia atrás.

Probablemente una de las grandes diferencias, comparado con éste, es que upstart utiliza orientación a eventos, para poder responder mejor(init seguía una secuencia definida).

También incorpora el concepto de "trabajos", los cuales se definen dentro de /etc/init/*.conf. Notarán que hay uno por servicio del sistema y además, que cada uno contiene 2 instrucciones muy importantes: start on(define cuándo ser iniciado) y stop on(define cuando ser detenido)

Upstart fue creado por Ubuntu para su distribución e incorpora un "método" nuevo para manejar los servicios; los comandos siguen este formato

start/stop/status servicio

El Kernel da el controla Upstart, al ejecutar la implementación propia de /sbin/init(tal y como lo hacía SysV) la cual crea un evento llamado startup que disparará la inicialización del sistema. De entre los trabajos disparados por startup, uno importantísimo es mountall, que se encarga de montar los sistemas de archivos y luego dispara varios eventos más, relacionados acon los discos y la inicialización de los sistemas de archivos. La cascada continúa con udev y luego el subsistema de red. Es entonces cuando se dispara rc-sysinit, cuyo rol(quizá uno de los más importantes) es el de llevar al sistema a su runlevel por defecto, a través de telinit <runlevel>. Con el fin de brindar compatibilidad hacia atrás, telinit dispara, entre otros, /etc/init/rc.conf job que ejecuta /etc/init.d/rc <runlevel> para comprobar si existe /etc/rc#.d/ y, en caso de que exista, ejecutar los scripts dentro de dicha carpeta.

Para el manejo de servicios, los comandos tienen el siguiente formato: *initctl start/status/stop job*.

También está disponible initctl list.

Los problemas de Upstart:

Si bien Upstart significó algunas mejoras sobre SysV, también tenía sus defectos. No tenía en cuenta, por ejemplo, los estados intermedios entre un evento y otro; debido a su "orientación a eventos", Upstart no chequeaba si un servicio estaba ya corriendo y lo iniciaba de todos modos si un evento lo incluía en su definición;

systemd:

Cómo decía antes, el nuevo Ubuntu 15.x y (Debian 8), reemplazan SysV con SystemD. Resultará algo raro para los lectores fans de Ubuntu, que saben que por un tiempo, gracias a Upstart, Ubuntu era una de las distros cuyo inicio era más rápido, gracias a la paralelización que ponía en práctica su sistema de inicialización.

De todos modos, como vimos antes Upstart aún dejó huecos por cubrir, ya que no era muy eficiente al reaccionar ante los cambios del sistema.

Por el otro, el paralelismo es otro de los puntos débiles a mejorar, para lo cual es necesario eliminar la cadena de dependencia entre los servicios(o al menos, disminuir la dependencia lo más posible). La base de esta dependencia es la necesidad de tener un socket disponible para que los procesos se comuniquen entre sí. Ya mencionamos que Systemd crea todos los sockets primero y luego inicia todos los procesos en paralelo. Con Upstart, un proceso tenía que esperar que el socket que necesita para iniciarse, fuera creado por un servicio iniciado antes que él.

Systemd introduce además, 2 nuevos conceptos: Units y targets: Un target(destino, o blanco, en inglés) es similar a los runlevels y está compuesto por varias unidades. Systemd ejecuta units para alcanzar un target. Las instrucciones para cada unidt están definidas en /lib/systemd/system/. Estos archivos contienen líneas con declaraciones y se asemejan a os archivos .ini de windows. Hay distintos tipos de units entre otros, socket, device, mount y, el más común probablemente, service. Veamos un ejemplo de unit:

```
[Unit]
Description=OpenSSH Daemon
Wants=sshdgenkeys.service
After=sshdgenkeys.service
After=network.target
[Service]
ExecStart=/usr/bin/sshd -D
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always
[Install]
WantedBy=multi-user.target
```

Un target es un tipo especial de archivo unit que define los tipos de units relacionados a dicho target. Un

ejemplo de target sería más o menos cómo esto:

[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
Wants=sockets.target timers.target paths.target
?slices.target
After=sysinit.target sockets.target timers.target
?paths.target slices.target
JobTimeoutSec=15min
JobTimeoutAction=poweroff-force

Como ven, cada archivo contiene una serie de definiciones bastante claras. Por ejemplo, Requires nos muestra las dependencias. Algo importante a tener en cuenta es que, de no existir la regla "After", las unidades podrán ser ejecutadas en paralelo por systemd.

Sobre journal y syslog

Tal vez el punto más polémico de systemd, es que reemplaza esa herramienta que prácticamente cualquier usuario de linux ha usado y le debe tanto: syslogd. Como he dicho antes, yo no soy ningún experto pero, leyendo las razones de esta medida, no estoy del todo convencido de que sea una buena idea. Aún así, "el chico nuevo", journal promete bastante y viene a corregir muchas falencias de syslog, como por ejemplo:

- **La falta de autenticación:** Preocupante vulnerabilidad, que puede permitir a procesos malintencionados, escribir en el log haciéndose pasar por otros.
- **Heterogeneidad:** Lo que significa que entradas generadas por distintos procesos, son distintas.
- **La "no indexación":** Por lo cual buscar algo específico puede llegar a ser muy tedioso.
- **Vulnerable a ciertos DoS**(denegación de servicio, es un tipo de ataque que busca saturar un sistema para que no pueda responder adecuadamente)
- **Posibilidad de falsificación de logs**
- **No disponer de logs en los momentos críticos**, como en el arranque del sistema.

Con journald, los logs son ahora archivos binarios y usar journalctl es bastante sencillo. Algunos de los comandos disponibles son:

journalctl -all(muestra el log completo)

journalctl -f(similar al tail)

journalctl -b -p err(muestra los errores desde el último boot

Pueden revisar la documentación oficial en http://freedesktop.org/wiki/Software/systemd

Les dejo también disponible el artículo original de introducción a systemd, en el blog de su autor, Lennart Poettering(En inglés):

http://0pointer.de/blog/projects/systemd.html

Finalmente, les invito a dejar sus opiniones sobre systemd!

---

memoriasdeunsysadmin.tk