systemd

- Niccoló Picca <picca.niccolo@gmail.org>

A cosa serve systemd?

A cosa serve systemd?

- Gestisce il boot
- Monta i dischi
- Gestisce i servizi

Perché proprio systemd?

Perché proprio systemd?

- Nato nel 2010, é diventato estremamento popolare
- Init System di default su:
  - Ubuntu
  - Debian
  - Fedora
  - Arch
  - Red Hat

Like it or not, it’s here to stay

L’idea chiave:

Start less, start in parallel

Start less, start in parallel

Ovvero, non bloccare un servizio a meno che non sia necessario

Start less, start in parallel

Esempio:

- Abbiamo una webapp W e un server MySQL M
- W dipende da M
- Comunicano via TCP sulla porta 3306
- In un init system tradizionale, W viene avviato dopo M

Start less, *start in parallel*

The systemd way:

- systemd ascolta sulla porta 3306
- I due servizi vengono avviati in contemporanea
- W viene caricata finché non prova a collegarsi a MySQL
- Appena M é pronto, systemd cede il controllo della porta
- W finisce l’avvio

Anche con il filesystem!

- Un servizio può richiedere che una certa directory sia montata.

  Ad esempio se si ha /home su una partizione distinta

- Se invece non ne ha bisogno, non aspetta!
- Quindi, il boot procede mentre è in corso un fsck su /home

Start less, start in parallel

- systemd limita il piú possibile gli avvi non necessari
  - Ad esempio, se nessuno si collega al server SQL, resta in attesa
- I servizi sono lanciati on-demand
  - Utile per servizi lanciati di rado (bluetooth…)
  - Poco utile su server
  - É possibile persino montare FS on demand! (autofs)

Altri vantaggi:

- Script di avvio dei servizi uniformi su tutte le distro
- Script di avvio mantenuti dagli sviluppatori
- Il formato per descrivere un servizio è dichiarativo
  - Niente init script Turing-completi!
- Sistema di boot in C (e non gestito da shell script)

Contro:

A molti dev non piace

Non rispetta la filosofia Unix:

“Make each program do one thing well”

16MB di PID1

- L’init system é il processo piú importante del sistema
- In caso di crash, tutto il sistema va in kernel panic
- Idealmente, dovrebbe essere il piú semplice possibile

16MB di PID1

- systemd invece é grosso e complesso
- Si interfaccia pure con la rete!
- Un bug in systemd potrebbe avere esiti catastrofici

Perché é cosí grande?

Non solo un init system:

- logind
- systemd-udev
- colord
- journald

Gli Unit file

Unit file

Una unit rappresenta una risorsa

Esempi di unit sono:

- **service**: un servizio (ad esempio un server)
- **socket**: un socket, usato per attivare servizi
- **device**: un device gestito da systemd
- **mount**: un punto di mount, ad esempio /home
- **timer**: un evento da eseguire periodicamente (cronjob)

Unit file

Si trovano principalmente in 3 posti:
`/{etc,run,lib}/systemd/system`

- **/etc**: configurazioni specifiche del sistema attuale
- **/run**: configurazioni generate automaticamente
- **/lib**: unit scritte da sviluppatori

Esempio di Unit: cups.service

```
Description=CUPS Scheduler
Documentation=man:cupsd(8)

[Service]
ExecStart=/usr/sbin/cupsd -l
Type=simple

[Install]
Also=cups.socket cups.path
WantedBy=printer.target
```

[Unit]

Supponiamo di avere uno unit file a

- **Requires=b** a dipende strettamente da b
- **Wants=b** a dipende debolmente da b
- **BindsTo=b** Lo stato di esecuzione a é legato a quello di b
- **Conflicts=b** Se a è avviato, b viene arrestato

Ordine:

- **requires** non forza l'ordine d'avvio
- le units vengono avviate in parallelo
- Per forzare l'ordine, bisogna specificare **before** e **after**

[Install]

- Specifica cosa fare quando una unit viene abilitata
- Abilitare significa (di solito) avviare al boot
- Supponiamo di avere un'altra unit c:
  - **WantedBy=c**: Come se c avesse Wants=a
  - **RequiredBy=c**: Come se c avesse Requires=a
  - **Also=c**: a e c sono "sorelle" e vengono gestite assieme

[Service]

Type:

Specifica il comportamento del processo:

- **simple**: il processo resta in esecuzione
- **forking**: il processo crea un figlio ed esce
- **oneshot**: il processo termina a breve
- **dbus**: il processo si rende disponibile su un dbus
- **notify**: il processo avviserá systemd alla fine dell'avvio

[Service]

Exec

- **ExecStart=**: Comando da invocare all'avvio
- **Execreload=**: Comando da invocare per ricaricare la configurazione
- **ExecStop=**: Comando da invocare all'arresto
- **Exec{Start,Stop}{Pre,Post}=**: Comando da invocare prima/dopo l'avvio/arresto
- **RemainAfterExit=yes**: In un servizio oneshot, é da considarsi in esecuzione anche dopo essere terminato

systemctl:

É un comando per gestire facilmente le unit di systemd

systemctl:

- **systemctl** Mostra la lista delle unit attive.
- **systemctl –all** Mostra tutte le unit disponibili.
- **systemctl start unit** Attiva la unit.
- **systemctl stop unit** Arresta la unit.
- **systemctl reload unit** Ricarica la configurazione di una unit.
- **systemctl enable unit** Abilita una unit
- **systemctl disable unit** Disinstalla una unit.
- **systemctl status unit** Mostra lo stato di una unit
- **systemctl mask unit** Impedisce l’esecuzione di una unit.
- **systemctl unmask unit** Permette l’esecuzione di una unit.

Template di unit

- Le unit con @ nel nome sono dei template
  - Ovvero, sono parametrici rispetto a quello che segue @
- Ad esempio: openvpn@.service → openvpn@**443**.service
- É possibile avviare la stessa unit con parametri diversi

Generatori di unit:

- Un generatore ispeziona il sistema e crea le unit appropriate
  - Ad esempio in base ai device disponibili, o a file di configurazione
- Al boot, vengono eseguiti quelli in `/lib/systemd/system-generators/`
- Le unit corrispondenti vengono create in `/run/systemd/generator/`

Esempio: systemd-fstab-generator

- Il file /etc/stab contiene la configurazione dei punti di moun

```
# <device>              <dir>         <type>    <options>              <c
/dev/sda1               /             ext4      defaults,noatime       0
/dev/sda2               none          swap      defaults               0
/dev/sda3               /home         ext4      defaults,noatime       0
```

- systemd-fstab-generator traduce le entry in mount unit

User Unit

- É possibile avere unit in un contesto utente
- Possono essere create in `~/.config/systemd/user`
- Utilizzabili con **systemctl --user**

Journaling

- Il Journaling registra messaggi utili all'amministratore
  - Tipicamente messaggi dal kernel o da servizi
- systemd ha un sistema di journaling, **journald**

journald

- registra **tutti** gli eventi di sistema
- il log é:
  - indicizzato
  - robusto alla corruzione
  - Forward Secure Sealing
- gli eventi sono registrati in formato binario
- tipicamente non é persistente
- configurabile da `/etc/systemd/journald.conf`

journald

- **journalctl** Mostra tutti i log disponibili.
- **journalctl -b** Mostra solo i log di questo boot.
- **journalctl -b -1** Mostra solo i log del boot precedente.
- **journalctl –list-boots** Mostra i boot per cui sono disponibili log.
- **journalctl –since yesterday** Mostra i log da ieri.
- **journalctl -u my.service** Mostra solo i log per my.service.

Altri strumenti

- **machinectl** Gestisce container (systemd-nspawn e altri).
- **loginctl** Interagisce con logind (ad es. per listare le sessioni attive)
- **timedatectl** Permette di controllare l'ora e il client NTP.
- **hostnamectl** Permette di modificare hostname e nome di dominio.
- **localectl** Permette di impostare la lingua e il layout della tastiera.

Any questions?

- Niccoló Picca <picca.niccolo@gmail.org>

Thank you!

Un sentito ringraziamento ad Alessandro Di Federico, autore delle slides originali

- Niccoló Picca <picca.niccolo@gmail.org>