Control services and Daemons

Linux #redhat #daemons

- Service or application when started creates processes and when those processes run continuously in the background, they become daemons.
- Most services are daemons.
- Services are controlled by `systemctl`
- `systemctl` is a systemd utility that is responsible for controlling the systemd system and service manager
- systemd is a collection of system management daemons, utilities, and libraries which serves as a replacement of System V init daemon
  - System V init daemon is supported only in older versions of Linux.
- systemd is the parent process of most of the daemons
- The command to control services = `systemctl`.
  - Every time you install an application, a package, or a service in your Linux environment, then you could control that program by running the command `systemctl`

Useful cases for Linux System Administrators:

- Check if systemd installed in your system
  - `systemctl --version`
  - systemd by default comes installed on a Linux operating system version 7, 8, and 9 (RHEL)
- Check if systemd is running
  - `ps -ef | grep system`
- Check all running services
  - `systemctl --all`
- To check the status, start, stop and restart a service
  - `systemctl status|start|stop|restart application.service`
- To reload the configuration of a service
  - `systemctl reload application.service`
- To enable or disable a service at boot time
  - `systemctl enable|disable application.service`
  - This will allow the system to find out which service to start at the time along the operating system.
- To enable or disable a service completely (at boot or manually)
  - `systemctl mask|unmask application.service`

Check if systemd installed in your system

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl --version
```

Output:

```
systemd 252 (252-32.el9_4)
+PAM +AUDIT +SELINUX -APPARMOR +IMA +SMACK +SECCOMP +GCRYPT +GNUTLS +OPENSSL +ACL +BLKID +CURL
```

```
+ELFUTILS -FIDO2 +IDN2 -IDN -IPTC +KMOD +LIBCRYPTSETUP +LIBFDISK +PCRE2 -PWQUALITY +P11KIT -QRENCODE
+TPM2 +BZIP2 +LZ4 +XZ +ZLIB +ZSTD -BPF_FRAMEWORK +XKBCOMMON +UTMP +SYSVINIT default-hierarchy=unified
```

- systemd is installed with version 252 (252-32.el9_4)
  - The "el" in the first line stands for "enterprise Linux 9.4"

Check if systemd is running

Example using `pd -ef` command:

```
[user@localhost ~]$ ps -ef | grep systemd
```

- If you have a systemd running then and only then you could actually execute the `systemctl` command, which controls these units and services.
- Use the `ps -ef` command and pipe it with a `grep systemd` to look for the full list of processes and check the appearances of systemd.

Output:

```
root           1       0  0 10:44 ?        00:00:07 /usr/lib/systemd/systemd rhgb --switched-root --
system --deserialize 31
root         656       1  0 10:45 ?        00:00:01 /usr/lib/systemd/systemd-journald
root         674       1  0 10:45 ?        00:00:01 /usr/lib/systemd/systemd-udevd
root         826       1  0 10:45 ?        00:00:02 /usr/lib/systemd/systemd-logind
mmarin      2164       1  0 10:45 ?        00:00:02 /usr/lib/systemd/systemd --user
mmarin      2269    2164  0 10:45 ?        00:00:01 /usr/libexec/gnome-session-binary --systemd-
service --session=gnome
mmarin      3466    3332  0 11:10 pts/1    00:00:00 grep --color=auto systemd
```

- The very first service, or the process that gets started when your computer starts. This is the systemd process. Its process ID will always be 1.
  - This process actually initiates all the other major or minor processes of your system.
  - If you want certain processes, for example chronED which is a time application to start at boot time. The systemd is the one that you are gonna tell that this is not a service and you want it to start when the system starts.
- If you kill this process then you will kill pretty much your entire operating system.
  - Because all the other services that are running in your system depends and are control by systemd

---

Packages

Software for Linux systems is distributed in the form of packages. A package is **an archive with software files, configuration files, and a list of required dependencies** — additional packages required for the software to run. Note that there are direct and indirect dependencies.

Example using `rpm` command:

```
[user@localhost ~]$ rpm -qa
```

- We have a lot of packages installed in our system.
  - What are those packages?
- The `rpm` command is defined as the RPM Package Manager.
- The `-q` option stands for "query"
- The `-a` option stands for "all" and specifies to query all installed packages.

Output:

```
...
rootfiles-8.1-31.el9.noarch
mailcap-2.1.49-5.el9.noarch
gnome-user-docs-40.0-3.el9.noarch
gutenprint-doc-5.3.4-4.el9.x86_64
telnet-0.17-85.el9.x86_64
```

- It will list all the packaages and applications that are installed in your Linux machine
- How do you know how many packages you have installed din your system?

Example using `rpm` command:

```
[user@localhost ~]$ rpm -qa | wc -l
```

- The `rpm -qa` command, which is a general form of a query command for all packages and applications, is being piped with the `wc` command to count the number of packages.
- The `wc` command prints newline, word, and byte counts for each line.
- The `-l` option of the `wc` command stands for "lines" and specifies to print the newline counts.

Output:

```
1183
```

- Does it mean that every package that you have installed has its service loaded and started and running?
  - No, there are only a few services that are running.
  - The other packages are just there in case it is needed.
- What is the list of services that we have running?
  - See next command...

Check all running services

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl --all
```

- Check all running services using the `systemctl`

Output:

```
  UNIT                                                 LOAD      ACTIVE   SUB       DESCRIPTION
● boot.automount                                       not-found inactive dead      boot.automount
  proc-sys-fs-binfmt_misc.automount                    loaded    active   waiting   Arbitrary
Executable File Formats File System Automount Point
  dev-cdrom.device                                     loaded    active   plugged   VBOX_CD-ROM
  dev-disk-by\x2ddiskseq-2.device                      loaded    active   plugged   VBOX_HARDDISK
  dev-disk-by\x2ddiskseq-3.device                      loaded    active   plugged   VBOX_CD-ROM
  dev-disk-by\x2did-ata\x2dVBOX_CD\x2dROM_VB2\x2d017...loaded    active   plugged   VBOX_CD-ROM
...

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
335 loaded units listed.
```

- By default it will give you one page at a time/
- You will see all these services.
    - On the left hand side is status of wheter its configuration file is loaded or not. (loaded or not loaded)
    - What is the current status at this time, for example (active or inactive)
    - Sub-status (plugged in or dead), if it has another dependency on another system, if this requires mounted.
        - Is pretty much all the threads, all the sub processes that are needed to run with that service.
    - To manage that process or to manage that mounting of that slash boot, that is managed by your `systemctl`
    - If the LOAD field is "not-loaded" we can assume that this is a service that has not a package.
- The firewalld.service is the firewall that allows you to protect your system from incoming or outgoing traffic. It says that its configuration file is loaded, the service is active, and it is running.
- At the end you will see all the services that are actually available to us to manage.

Check the status, start, stop and restart a service

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl status firewalld.service
```

- Checking the status of the firewalld service.
- We ant to check the service so we have to put the ".service" at the end of the service name.

Output:

```
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-06-13 10:45:12 CDT; 1h 16min ago
       Docs: man:firewalld(1)
   Main PID: 871 (firewalld)
      Tasks: 2 (limit: 65779)
     Memory: 43.7M
        CPU: 2.849s
     CGroup: /system.slice/firewalld.service
             └─871 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Jun 13 10:45:10 localhost systemd[1]: Starting firewalld - dynamic firewall daemon...
Jun 13 10:45:12 localhost systemd[1]: Started firewalld - dynamic firewall daemon.
```

- The actual status of the service is that it is started, it's active.
- All the require processes that are needed for this service to be active, are those processes running? Yes, they are running.
- When was the last time they ran? That is the date and time
- It also gives you the documentation for this service.
- The main process ID is 871.
- If you kill the 871 process using the `kill` command it will kill it but that is not the right way to stop the service.
  - To stop the service, what you have to do is use `systemctl`.
- Note that when the service is active, the word "active" will be highlighted in green.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl stop firewalld.service
```

- Attempting to stop the firewalld service.

Output:

```
==== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to stop 'firewalld.service'.
Authenticating as: Maximus Marin (mmarin)
Password:
==== AUTHENTICATION COMPLETE ====
```

- If you are not root user it will ask you for authentication of your own user.
- After stopping the service you can verify it stopped by running the command `systemctl status firewalld.sercice` and verifying it displays as inactive.
- Alternatively you can use the `ps -ef | grep firewalld` to look for all the process currently running listed as firewalld. If the only result is actually the `grep` command looking for that term then the service and all of its processes are actually stopped.
- What if the service you are attempting to stop is not stopping by using the `systemctl stop` command, something is wrong, that's when you have to actually come and kill that service by using the `kill` command and giving the main process ID.
  - Killing is not an efficient way or a graceful way to stop a service. The best way to stop a service is by using the `systemctl stop` command.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl start firewalld.service
```

- Starting the firewalld.service using the `systemctl` command.
- It is a good practice to run the `systemctl status firewalld.service` command just after starting or stopping the service jut to make sure the status did change.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl restart firewalld.service
```

- The `restart` function is needed when you make changes to its configuration files. Lets say you wanted to change the behavior of the firewalld service, or you are making firewalld allowing a port, enabling or disabling or anything that you have made changes to the configuration file.

  - Instead of having to stop and start the service with two different commands, you could just use one command that is simply restart.

Output from `systemctl status firewalld.service`:

```
● firewalld.service - firewalld - dynamic firewall daemon
     Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-06-13 12:28:59 CDT; 12s ago
       Docs: man:firewalld(1)
   Main PID: 4003 (firewalld)
      Tasks: 2 (limit: 65779)
     Memory: 22.8M
        CPU: 348ms
     CGroup: /system.slice/firewalld.service
             └─4003 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Jun 13 12:28:59 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
Jun 13 12:28:59 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
```

- The status function of the `systemctl` command gives you a small and quick log which tells you when you executed the command, what happened, it started, stopped, etc.

Reload the configuration of a service

- What is the difference between restart and reload?
  - Reload is not going to impact your service, whereas stop and start will impact your service.
  - If you are running a service, for example Apache, as soon as you make a configuration changes on Apache, and you stop it, then you are gonna lose the page, and anybody who's on your website, you will lose it and until you restart it.
  - If you are making changes to your configuration file and you use the command `reload` then it will reload the configuration file back again to `systemctl` without stopping and starting the application. That's the beauty of reload.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl reload firewalld.service
```

- Reloading the configuration file of the firewalld service by using the `reload` function of the `systemctl` command.

Enable or disable a service at boot time

- If you want a service to boot while your operating system boots. Or in other word, if you want, for example firewalld, to start when your operating system starts with that, then you have to ENABLE that service.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl enable firewalld.service
```

- Enabling the firewalld service to boot with the system by using the `enable` function of the `systemctl` command.

- To check if the service is enabled or not you can run the `systemctl status firewalld.service` command and right on the loaded section it states enabled or disabled.
  - If the service is enabled, the word "enabled" will be highlighted in green.
  - If the service is disabled, the word "disabled" will be highlighted in yellow.

Output:

```
Created symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service →
/usr/lib/systemd/system/firewalld.service.
Created symlink /etc/systemd/system/multi-user.target.wants/firewalld.service →
/usr/lib/systemd/system/firewalld.service.
```

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl disable firewalld.service
```

- Disabling the firewalld service from booting with system by using the `disable` function of the `systemctl` command.

Output:

```
Removed "/etc/systemd/system/multi-user.target.wants/firewalld.service".
Removed "/etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service".
```

- There are a couple of links that it has to remove.
- Now if you reboot your entire operating system, it will not start your firewalld service, but it doees NOT impact the existing state.

Enable or disable a service completely (mask or unmask)

- It is possible that if a service A has to start and it requires another service B to start with it. Then regardless if it is enabled or disabled, it will start the service.
  - For example, if you have HTTP service that it starts. That service requires to start another service that let's say in our example firewalld. Then regardless if that firewalld is enabled or disabled, it will start.
- If we have that service masked or unmasked, then doesn't matter which service which service try to request for that service to start, it will NOT start.

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl mask firewalld.service
```

- Creating mask for the firewalld service with the `mask` function of the `systemctl` command.

Output:

```
Created symlink /etc/systemd/system/firewalld.service → /dev/null.
```

- It creates a symbolic link (symlink)

Example using `systemctl` command:

```
[user@localhost ~]$ systemctl unmask firewalld.service
```

- Unmasking firewalld service with the `unmask` function of the `systemctl` command.

Output:

```
Removed "/etc/systemd/system/firewalld.service".
```

- It removes the symbolic link previously created.
- Now if a service A comes in and it wants a service B to start, and that server B has been unmasked then it'll not start regardless what that service instruction comes in with.

Output from `systemctl status firewalld.service`:

```
● firewalld.service
     Loaded: masked (Reason: Unit firewalld.service is masked.)
     Active: active (running) since Thu 2024-06-13 12:28:59 CDT; 40min ago
   Main PID: 4003 (firewalld)
        CPU: 553ms
     CGroup: /system.slice/firewalld.service
             └─4003 /usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid

Jun 13 12:28:59 localhost.localdomain systemd[1]: Starting firewalld - dynamic firewall daemon...
Jun 13 12:28:59 localhost.localdomain systemd[1]: Started firewalld - dynamic firewall daemon.
Jun 13 12:41:49 localhost.localdomain systemd[1]: Reloading firewalld - dynamic firewall daemon...
Jun 13 12:41:49 localhost.localdomain systemd[1]: Reloaded firewalld - dynamic firewall daemon.
Jun 13 13:02:15 localhost.localdomain systemd[1]: firewalld.service: Current command vanished from the unit file, execu
```

- This is how the status of the firewalld service would look like if the service is masked. Meaning it can be started by any other service that request it.
- Note that on the Loaded section it appears as "masked".

```
Dataview (inline field '=== AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
Authentication is required to stop 'firewalld.service'.
Authenticating as: Maximus Marin (mmarin)
Password:
==== AUTHENTICATION COMPLETE ===='): Error:
-- PARSING FAILED --------------------------------------------------

> 1 | === AUTHENTICATING FOR org.freedesktop.systemd1.manage-units ====
    | ^
  2 | Authentication is required to stop 'firewalld.service'.
  3 | Authenticating as: Maximus Marin (mmarin)

Expected one of the following:
```

```
'(', 'null', boolean, date, duration, file link, list ('[1, 2, 3]'), negated field,
number, object ('{ a: 1, b: 2 }'), string, variable
```