Managing System and Infrastructure Services

Linux Administration

Asst. Prof. Ashwini Mathur

Service

A Linux service is an application (or set of applications) that runs in the background waiting to be used, or carrying out essential tasks

Systemd utility

What is systemd?

systemd is a Linux system tool initially developed by the Red Hat Linux team. It includes many features, including a bootstrapping system used to start and manage system processes. It is currently the default initialization system on most Linux distributions. Many commonly used software tools, such as SSH and Apache, ship with a systemd service.

It is simple to create a custom systemd service that will run any script or process you choose. Although there are several ways to run a script or start a process when your Linode boots, a custom systemd service makes it easy to start, stop, or restart your script, as well as configure it to start automatically on boot.

Note :

There are currently 3 main init systems used by Linux. A few years ago, there was just one, , but it’s been deprecated in most distros by now.

Currently, most distros are switching to **systemd**, for example, Debian Jessie. The most notable distribution using systemd are Fedora, CentOS, RedHat, OpenSuse, Ubuntu, Mint.

service	systemctl	Description
`service` *`name`* `start`	`systemctl start` *`name`*`.service`	Starts a service.
`service` *`name`* `stop`	`systemctl stop` *`name`*`.service`	Stops a service.
`service` *`name`* `restart`	`systemctl restart` *`name`*`.service`	Restarts a service.
`service` *`name`* `condrestart`	`systemctl try-restart` *`name`*`.service`	Restarts a service only if it is running.
`service` *`name`* `reload`	`systemctl reload` *`name`*`.service`	Reloads configuration.

`service` *`name`* `status`	`systemctl status` *`name`*`.service`
	
	`systemctl is-active` *`name`*`.service`
		Checks if a service is running.

`service --status-all`	`systemctl list-units --type service --all`	Displays the status of all services.

Field	Description
`Loaded`	Information whether the service unit has been loaded, the absolute path to the unit file, and a note whether the unit is enabled.
`Active`	Information whether the service unit is running followed by a time stamp.
`Main PID`	The PID of the corresponding system service followed by its name.
`Status`	Additional information about the corresponding system service.
`Process`	Additional information about related processes.
`CGroup`	Additional information about related Control Groups (cgroups).

Table 10.8. Comparison of Power Management Commands with systemctl

Old Command	New Command	Description
`halt`	`systemctl halt`	Halts the system.
`poweroff`	`systemctl poweroff`	Powers off the system.
`reboot`	`systemctl reboot`	Restarts the system.
`pm-suspend`	`systemctl suspend`	Suspends the system.
`pm-hibernate`	`systemctl hibernate`	Hibernates the system.
`pm-suspend-hybrid`	`systemctl hybrid-sleep`	Hibernates and suspends the system.

Create a Custom systemd Service

1. Create a script or executable that the service will manage. This guide uses a simple Bash script as an example:

   test_service.sh

   ```
   DATE=`date '+%Y-%m-%d %H:%M:%S'`
   echo "Example service started at ${DATE}" | systemd-cat -p info

   while :
   do
   echo "Looping...";
   sleep 30;
   done
   ```

   This script will log the time at which it is initialized, then loop infinitely to keep the service running.

2. Copy the script to `/usr/bin` and make it executable:

   ```
   sudo cp test_service.sh /usr/bin/test_service.sh
   sudo chmod +x /usr/bin/test_service.sh
   ```

3. Create a **Unit file** to define a systemd service:

/lib/systemd/system/myservice.service

```
[Unit]
Description=Example systemd service.

[Service]
Type=simple
ExecStart=/bin/bash /usr/bin/test_service.sh

[Install]
WantedBy=multi-user.target
```

This defines a simple service. The critical part is the `ExecStart` directive, which specifies the command that will be run to start the service.

4. Copy the unit file to `/etc/systemd/system` and give it permissions:

```
sudo cp myservice.service /etc/systemd/system/myservice.service
sudo chmod 644 /etc/systemd/system/myservice.service
```

For more information about the unit file and its available configuration options, see the systemd documentation.

Start and Enable the Service

1. Once you have a unit file, you are ready to test the service:

   ```
   sudo systemctl start myservice
   ```

2. Check the status of the service:

   ```
   sudo systemctl status myservice
   ```

   If the service is running correctly, the output should resemble the following:

   ```
   ● myservice.service - Example systemd service.
      Loaded: loaded (/lib/systemd/system/myservice.service; enabled; vendor preset: enabled)
      Active: active (running) since Tue 2018-05-01 18:17:14 UTC; 4s ago
    Main PID: 16266 (bash)
       Tasks: 2
      Memory: 748.0K
         CPU: 4ms
      CGroup: /system.slice/myservice.service
              ├─16266 /bin/bash /usr/bin/test_service.sh
              └─16270 sleep 30

   May 01 18:17:14 localhost systemd[1]: Started Example systemd service..
   May 01 18:17:14 localhost cat[16269]: Example service started at 2018-05-01 18:17:14
   May 01 18:17:14 localhost bash[16266]: Looping...
   ```

Table 10.6. Comparison of SysV Runlevels with systemd Targets

Runlevel	Target Units	Description
`0`	`runlevel0.target`, `poweroff.target`	Shut down and power off the system.
`1`	`runlevel1.target`, `rescue.target`	Set up a rescue shell.
`2`	`runlevel2.target`, `multi-user.target`	Set up a non-graphical multi-user system.
`3`	`runlevel3.target`, `multi-user.target`	Set up a non-graphical multi-user system.
`4`	`runlevel4.target`, `multi-user.target`	Set up a non-graphical multi-user system.
`5`	`runlevel5.target`, `graphical.target`	Set up a graphical multi-user system.
`6`	`runlevel6.target`, `reboot.target`	Shut down and reboot the system.