How to use systemctl to manage Linux services

By Damon Garn

Published: 2022-04-21 · Archived: 2026-04-05 23:40:37 UTC

Suppose you're making configuration changes to a Linux server. Perhaps you just fired up Vim and made edits to the `/etc/ssh/sshd_config` file, and it's time to test your new settings. Now what?

Services such as SSH pull their settings from configuration files during the startup process. To let the service know about changes to the file, you need to restart the service so that it rereads the file. You can use the `systemctl` command to manage services and control when they start.

Restart a service

After editing the `/etc/ssh/sshd_config` file, use the `systemctl restart` command to make the service pick up the new settings:

```
$ sudo systemctl restart sshd
```

You can verify the service is running by using the `status` subcommand:

```
$ sudo systemctl status sshd
```

Stop and start a service

Perhaps while troubleshooting you need to stop a service to determine whether it is the culprit or interfering with some other process. Use the `stop` subcommand for this:

```
$ sudo systemctl stop sshd
```

Once you determine if this service is associated with the issue, you can restart it:

```
$ sudo systemctl start sshd
```

While the `restart` subcommand is useful for refreshing a service's configuration, the `stop` and `start` features give you more granular control.

Control whether the service starts with the system

One consideration with using `stop` and `start` is that the two commands apply only to the current runtime. The next time you boot the system, the service will either start or not start, depending on its default settings. You can use the `enable` and `disable` subcommands to manage those defaults.

When you `disable` the service, it doesn't start the next time the system boots. You might use this setting as part of your security hardening process or for troubleshooting:

```
$ sudo systemctl disable sshd
```

Reboot the system with `reboot` `sudo systemctl reboot`, and the service won't automatically start.

You may determine that you need the service to start automatically. In that case, use the `enable` subcommand:

```
$ sudo systemctl enable sshd
```

The `enable` subcommand doesn't start a service, it only marks it to start automatically at boot. To enable and start a service at the same time, use the `--now` option:

```
$ sudo systemctl enable --now sshd
```

[Free download: Advanced Linux commands cheat sheet.]

Mask a service

You can manually start a disabled service with the `systemctl start` command after the system boots. To prevent this, use the `mask` subcommand. Masking the service links its configuration to `/dev/null`. A user or process will not be able to start this service at all (whereas with a disabled service, a user or process can still start it). Use the `unmask` subcommand to reverse the setting:

```
$ sudo systemctl mask sshd
```

Display all subcommands

Bash's built-in tab-completion feature is one of my favorite tricks for `systemctl` (and other commands). When working with commands that support subcommands, this feature saves you a lot of time. Simply type `systemctl` and add a space, then tap the **Tab** key twice. Bash displays all available subcommands.

The challenge

Do you think you're ready to use `systemctl` to manage your services? Fire up a lab virtual machine and choose a service to work with. Don't do this on a production system! Make sure you can accomplish the following tasks:

1. Check the status of your service. Is it started? Enabled?

2. Stop the service and recheck its status.
3. Disable the service, reboot the system, and confirm the service did not start.
4. Enable the service, reboot the system, and confirm the service did start.
5. Stop the service and use the `mask` subcommand to prevent it from launching. Use the `systemctl start` command to attempt to start it. Were you successful?
6. Use Bash's tab-completion feature to display all available subcommands for `systemctl`.

Wrap up

Many management tasks involve the `systemctl` command, but the ones covered above represent the majority of them. Service management is critical, especially when editing configuration files and hardening a system. Plan to be confident, competent, and quick at using `systemctl` and its common subcommands.
