**Systemd Essentials: Working with Services, Units, and the Journal**

[System Tools](https://www.digitalocean.com/community/tags/system-tools)[Logging](https://www.digitalocean.com/community/tags/logging)[Linux Commands](https://www.digitalocean.com/community/tags/linux-commands)

* By [Justin Ellingwood](https://www.digitalocean.com/community/users/jellingwood)

Published onApril 20, 2015 336.7kviews

Basic Unit Management

The basic object that systemd manages and acts upon is a “unit”. Units can be of many types, but the most common type is a “service” (indicated by a unit file ending in .service). To manage services on a systemd enabled server, our main tool is the systemctl command.

All of the normal init system commands have equivalent actions with the systemctl command. We will use the nginx.service unit to demonstrate (you’ll have to install Nginx with your package manager to get this service file).

For instance, we can start the service by typing:

* sudo systemctl start nginx.service

We can stop it again by typing:

* sudo systemctl stop nginx.service

To restart the service, we can type:

* sudo systemctl restart nginx.service

To attempt to reload the service without interrupting normal functionality, we can type:

* sudo systemctl reload nginx.service

Enabling or Disabling Units

By default, most systemd unit files are not started automatically at boot. To configure this functionality, you need to “enable” to unit. This hooks it up to a certain boot “target”, causing it to be triggered when that target is started.

To enable a service to start automatically at boot, type:

* sudo systemctl enable nginx.service (–now enable and start)

If you wish to disable the service again, type:

* sudo systemctl disable nginx.service

Getting an Overview of the System State

There is a great deal of information that we can pull from a systemd server to get an overview of the system state.

For instance, to get all of the unit files that systemd has listed as “active”, type (you can actually leave off the list-units as this is the default systemctl behavior):

* systemctl list-units

To list all of the units that systemd has loaded or attempted to load into memory, including those that are not currently active, add the --all switch:

* systemctl list-units --all

To list all of the units installed on the system, including those that systemd has not tried to load into memory, type:

* systemctl list-unit-files

Viewing Basic Log Information

A systemd component called journald collects and manages journal entries from all parts of the system. This is basically log information from applications and the kernel.

To see all log entries, starting at the oldest entry, type:

* journalctl

By default, this will show you entries from the current and previous boots if journald is configured to save previous boot records. Some distributions enable this by default, while others do not (to enable this, either edit the /etc/systemd/journald.conf file and set the Storage= option to “persistent”, or create the persistent directory by typing sudo mkdir -p /var/log/journal).

If you only wish to see the journal entries from the current boot, add the -b flag:

* journalctl -b

To see only kernel messages, such as those that are typically represented by dmesg, you can use the -k flag:

* journalctl -k

Again, you can limit this only to the current boot by appending the -b flag:

* journalctl -k -b

Querying Unit States and Logs

While the above commands gave you access to the general system state, you can also get information about the state of individual units.

To see an overview of the current state of a unit, you can use the status option with the systemctl command. This will show you whether the unit is active, information about the process, and the latest journal entries:

* systemctl status nginx.service

To see all of the journal entries for the unit in question, give the -u option with the unit name to the journalctl command:

* journalctl -u nginx.service

As always, you can limit the entries to the current boot by adding the -b flag:

* journalctl -b -u nginx.service

Inspecting Units and Unit Files

By now, you know how to modify a unit’s state by starting or stopping it, and you know how to view state and journal information to get an idea of what is happening with the process. However, we haven’t seen yet how to inspect other aspects of units and unit files.

A unit file contains the parameters that systemd uses to manage and run a unit. To see the full contents of a unit file, type:

* systemctl cat nginx.service

To see the dependency tree of a unit (which units systemd will attempt to activate when starting the unit), type:

* systemctl list-dependencies nginx.service

This will show the dependent units, with target units recursively expanded. To expand all dependent units recursively, pass the --all flag:

* systemctl list-dependencies --all nginx.service

Finally, to see the low-level details of the unit’s settings on the system, you can use the show option:

* systemctl show nginx.service

This will give you the value of each parameter being managed by systemd.

Modifying Unit Files

If you need to make a modification to a unit file, systemd allows you to make changes from the systemctl command itself so that you don’t have to go to the actual disk location.

To add a unit file snippet, which can be used to append or override settings in the default unit file, simply call the edit option on the unit:

* sudo systemctl edit nginx.service

Copy

If you prefer to modify the entire content of the unit file instead of creating a snippet, pass the --full flag:

* sudo systemctl edit --full nginx.service

After modifying a unit file, you should reload the systemd process itself to pick up your changes:

* sudo systemctl daemon-reload

Using Targets (Runlevels)

Another function of an init system is to transition the server itself between different states. Traditional init systems typically refer to these as “runlevels”, allowing the system to only be in one runlevel at any one time.

In systemd, “targets” are used instead. Targets are basically synchronization points that the server can used to bring the server into a specific state. Service and other unit files can be tied to a target and multiple targets can be active at the same time.

To see all of the targets available on your system, type:

* systemctl list-unit-files --type=target (systemctl –t target)

To view the default target that systemd tries to reach at boot (which in turn starts all of the unit files that make up the dependency tree of that target), type:

* systemctl get-default

You can change the default target that will be used at boot by using the set-default option:

* sudo systemctl set-default multi-user.target

To see what units are tied to a target, you can type:

* systemctl list-dependencies multi-user.target

You can modify the system state to transition between targets with the isolate option. This will stop any units that are not tied to the specified target. Be sure that the target you are isolating does not stop any essential services:

* sudo systemctl isolate multi-user.target

Stopping or Rebooting the Server

For some of the major states that a system can transition to, shortcuts are available. For instance, to power off your server, you can type:

* sudo systemctl poweroff

If you wish to reboot the system instead, that can be accomplished by typing:

* sudo systemctl reboot

You can boot into rescue mode by typing:

* sudo systemctl rescue

Note that most operating systems include traditional aliases to these operations so that you can simply type sudo poweroff or sudo reboot without the systemctl. However, this is not guaranteed to be set up on all systems.

Next Steps

By now, you should know the basics of how to manage a server that uses systemd. However, there is much more to learn as your needs expand. Below are links to guides with more in-depth information about some of the components we discussed in this guide:

* [How To Use Systemctl to Manage Systemd Services and Units](https://www.digitalocean.com/community/tutorials/how-to-use-systemctl-to-manage-systemd-services-and-units)
* [How To Use Journalctl to View and Manipulate Systemd Logs](https://www.digitalocean.com/community/tutorials/how-to-use-journalctl-to-view-and-manipulate-systemd-logs)
* [Understanding Systemd Units and Unit Files](https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files)

By learning how to leverage your init system’s strengths, you can control the state of your machines and more easily manage your services and processes.