Community (https://www.linode.com/community/)

Questions (https://www.linode.com/community/questions) Guides & Tutorials (https://www.linode.com/docs/) StackScripts (https://www.linode.com/stackscripts) GitHub (https://github.com/linode) Events (https://www.linode.com/events)

Guides & Tutorials (https://www.linode.com/docs/)
» Quick Answers (https://www.linode.com/docs/quick-answers/)
» Linux Essentials (https://www.linode.com/docs/quick-answers/linux-essentials/)
» Introduction to systemctl

Introduction to systemctl

Updated Wednesday, September 12, 2018 by Linode

Search guides...

Contributed by Linode

Try this Guide

In This Guide:

Contribute on GitHub

Report an Issue (https://github.com/linode/docs/issues/new?title=Introduction%20to%20systemctl%20Proposed%20Changes&body=Link%3A https%3A%2F%2Flinode.com%2fdocs%2fquick-answers%2flinux-essentials%2fintroduction-to-systemctl%2f%0A%23%23%20Issue%0A%0A%23%23%20Suggested%20Fix%0A&labels=inaccurate guide) | View File (https://github.com/linode/docs/blob/master/docs/quick-answers/linux-essentials/introduction-to-systemctl/index.md) | Edit File (https://github.com/linode/docs/edit/develop/docs/quick-answers/linux-essentials/introduction-to-systemctl/index.md)

RSS feed (https://www.linode.com/docs/index.xml)

What is systemctl?

`systemctl` is a controlling interface and inspection tool for the widely-adopted init system and service manager systemd. This guide will cover how to use `systemctl` to manage systemd services, work with systemd Targets and extract meaningful information about your system's overall state.

> **Note**
> This guide is written for a non-root user. Commands that require elevated privileges are prefixed with `sudo`. If you're not familiar with the `sudo` command, see the Users and Groups (/docs/tools-reference/linux-users-and-groups/) guide.

Managing Services

`systemd` initializes *user space* components that run after the Linux kernel has booted, as well as continuously maintaining those components throughout a system's lifecycle. These tasks are known as *units*, and each unit has a corresponding *unit file*. Units might concern mounting storage devices (.mount), configuring hardware (.device), sockets (.socket), or, as will be covered in this guide, managing services (.service).

Starting and Stopping a Service

To start a `systemd` service in the current session, issue the `start` command:

```
sudo systemctl start apache2.service
```

Conversely, to stop a `systemd` service, issue the `stop` command:

```
sudo systemctl stop apache2.service
```

In the above example we started and then stopped the Apache service. It is important to note that `systemctl` does not require the `.service` extension when working with service units. The following is just as acceptable:

```
sudo systemctl start apache2
```

If the service needs to be restarted, such as to reload a configuration file, you can issue the `restart` command:

```
sudo systemctl restart apache2
```

Similarly, if a service does not need to restart to reload it's configuration, you can issue the `reload` command:

```
sudo systemctl reload apache2
```

Finally, you can use the `reload-or-restart` command if you are unsure about whether your application needs to be restarted or just reloaded.

```
sudo systemctl reload-or-restart apache2
```

Enabling a Service at Boot

The above commands are good for managing a service in a single session, but many services are also required to start at boot. To enable a service at boot:

```
sudo systemctl enable nginx
```

To disable the service from starting at boot, issue the `disable` command:

```
sudo systemctl disable nginx
```

> **Note**
> The `enable` command does not start the service in the current session, nor does `disable` stop the service in the current session. To enable/disable and start/stop a service simultaneously, combine the command with the `--now` switch:
>
> ```
> sudo systemctl enable nginx --now
> ```

If the service unit file is not located within one of the known `systemd` file paths, you can provide a file path to the service unit file you wish to enable:

```
sudo systemctl enable /path/to/myservice.service
```

However, this file needs to be accessible by `systemd` at startup. For example, this means files underneath `/home` or `/var` are not allowed, unless those directories are located on the root file system.

Checking a Service's Status

`systemctl` allows us to check on the status of individual services:

```
systemctl status mysql
```

This will result in a message similar to the output below:

```
● mysql.service - MySQL Community Server
  Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2018-08-30 09:15:35 EDT; 1 day 5h ago
Main PID: 711 (mysqld)
   Tasks: 31 (limit: 2319)
  CGroup: /system.slice/mysql.service
          └─711 /usr/sbin/mysqld --daemonize --pid-file=/run/mysqld/mysqld.pid
```

You can also use `is-active`, `is-enabled`, and `is-failed` to monitor a service's status:

```
systemctl is-enabled mysql
```

To view which `systemd` service units are currently active on your system, issue the following `list-units` command and filter by the service type:

```
systemctl list-units --type=service
```

> **Note**
> `list-units` is the default action for the `systemctl` command, so you can simply enter `systemctl` to retrieve a list of units.

The generated list includes all currently active service units, service units that have jobs pending, and service units that were active and have failed:

```
UNIT                            LOAD   ACTIVE SUB       DESCRIPTION
accounts-daemon.service         loaded active running   Accounts Service
apparmor.service                loaded active exited    AppArmor initialization
apport.service                  loaded active exited    LSB: automatic crash report generation
atd.service                     loaded active running   Deferred execution scheduler
blk-availability.service        loaded active exited    Availability of block devices
console-setup.service           loaded active exited    Set console font and keymap
cron.service                    loaded active running   Regular background program processing da
dbus.service                    loaded active running   D-Bus System Message Bus
ebtables.service                loaded active exited    ebtables ruleset management
...
```

The output provides five pieces of data:

- **UNIT**: The name of the unit.
- **LOAD**: Was the unit properly loaded?
- **ACTIVE**: The general activation state, i.e. a generalization of SUB.
- **SUB**: The low-level unit activation state, with values dependent on unit type.
- **DESCRIPTION**: The unit's description.

To list all units, including inactive units, append the `--all` flag:

```
systemctl list-units --type=service --all
```

You can filter the list of units by state. Supply a comma-separated list of unit states to output as the value for the `--state` flag:

```
systemctl list-units --type=service --all --state=exited,inactive
```

To retrieve a list of failed units, enter the `list-units` command with the `--failed` flag:

```
systemctl list-units --failed
```

Working with Unit Files

Each unit has a corresponding *unit file*. These unit files are usually located in the following directories:

- The `/lib/systemd/system` directory holds unit files that are provided by the system or are supplied by installed packages.
- The `/etc/systemd/system` directory stores unit files that are user-provided.

Listing Installed Unit Files

Not all unit files are active on a system at any given time. To view all `systemd` service unit files installed on a system, use the `list-unit-files` command with the optional `--type` flag:

```
systemctl list-unit-files --type=service
```

The generated list has two columns, **UNIT FILE** and **STATE**:

```
UNIT FILE                              STATE
accounts-daemon.service                enabled
acpid.service                          disabled
apparmor.service                       enabled
apport-forward@.service                static
apt-daily-upgrade.service              static
apt-daily.service                      static
...
```

A unit's STATE can be either enabled, disabled, static, masked, or generated. Unit files with a static state do not contain an *Install* section and are either meant to be run once or they are a dependency of another unit file and should not be run alone. For more on masking, see Masking a Unit File.

Viewing a Unit File

To view the contents of a unit file, run the `cat` command:

```
systemctl cat cron
```

```
# /lib/systemd/system/cron.service
[Unit]
Description=Regular background program processing daemon
Documentation=man:cron(8)

[Service]
EnvironmentFile=-/etc/default/cron
ExecStart=/usr/sbin/cron -f $EXTRA_OPTS
IgnoreSIGPIPE=false
KillMode=process

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

If there are recent changes to the unit file that have not yet been loaded into systemd, the output of the `systemctl cat` command may be an older version of the service.

For a low-level view of a unit file, issue the `show` command:

```
systemctl show cron
```

This will generate a list of property key=value pairs for all non-empty properties defined in the unit file:

```
Type=simple
Restart=no
NotifyAccess=none
RestartUSec=100ms
TimeoutStartUSec=1min 30s
TimeoutStopUSec=1min 30s
RuntimeMaxUSec=infinity
...
```

To show empty property values, supply the `--all` flag.

To filter the key=value pairs by property, use the `-p` flag:

```
systemctl show cron -p Names
```

Note that the property name must be capitalized.

Viewing a Unit File's Dependencies

To display a list of a unit file's dependencies, use the `list-dependencies` command:

```
systemctl list-dependencies cron
```

The generated output will show a tree of unit dependencies that must run before the service in question runs.

```
cron.service
● ├─system.slice
● └─sysinit.target
●   ├─apparmor.service
●   ├─blk-availability.service
●   ├─dev-hugepages.mount
●   ├─dev-mqueue.mount
●   ├─friendly-recovery.service
...
```

Recursive dependencies are only listed for `.target` files. To list all recursive dependencies, pass in the `--all` flag.

To check which unit files depend on a service unit file, you can run the `list-dependencies` command with the `--reverse` flag:

```
systemctl list-dependencies cron --reverse
```

Editing a Unit File

> **Note**
> While the particulars of unit file contents are beyond the scope of this article, there are a number of good resources online that describe them, such as the RedHat Customer Portal page on Creating and Modifying systemd Unit Files (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files).

There are two ways to edit a unit file using `systemctl`.

1. The `edit` command opens up a blank drop-in snippet file in the system's default text editor:

   ```
   sudo systemctl edit ssh
   ```

   When the file is saved, `systemctl` will create a file called `override.conf` under a directory at `/etc/systemd/system/yourservice.service.d`, where `yourservice` is the name of the service you chose to edit. This command is useful for changing a few properties of the unit file.

2. The second way is to use the `edit` command with the `--full` flag:

   ```
   sudo systemctl edit ssh --full
   ```

   This command opens a full copy of whatever unit file you chose to edit in a text editor. When the file is saved, `systemctl` will create a file at `/etc/systemd/system/yourservice.service`. This is useful if you need to make many changes to an existing unit file.

In general, any unit file in `/etc/systemd/system` will override the corresponding file in `/lib/systemd/system`.

Creating a Unit File

While `systemctl` will throw an error if you try to open a unit file that does not exist, you can force `systemctl` to create a new unit file using the `--force` flag:

```
sudo systemctl edit yourservice.service --force
```

When the file is saved, `systemctl` will create an `override.conf` file in the `/etc/systemd/system/yourservice.service.d` directory, where 'yourservice' is the name of the service you chose to create. To create a full unit file instead of just a snippet, use `--force` in tandem with `--full`:

```
sudo systemctl edit yourservice.service --force --full
```

Masking a Unit File

To prevent a service from ever starting, either manually or automatically, use the `mask` command to symlink a service to `/dev/null`:

```
sudo systemctl mask mysql
```

Similar to disabling a service, the `mask` command will not prevent a service from continuing to run. To mask a service and stop the service at the same time, use the `--now` switch:

```
sudo systemctl mask mysql --now
```

To unmask a service, use the `unmask` command:

```
sudo systemctl unmask mysql
```

Removing a Unit File

To remove a unit file snippet that was created with the `edit` command, remove the directory `yourservice.service.d` (where 'yourservice' is the service you would like to delete), and the `override.conf` file inside of the directory:

```
sudo rm -r /etc/systemd/system/yourservice.service.d
```

To remove a full unit file, run the following command:

```
sudo rm /etc/systemd/system/yourservice.service
```

After you issue these commands, reload the `systemd` daemon so that it no longer tries to reference the deleted service:

```
sudo systemctl daemon-reload
```

Working with systemd Targets

Like other init system's run levels, `systemd`'s targets help it determine which unit files are necessary to produce a certain system state. `systemd` targets are represented by *target units*. Target units end with the `.target` file extension and their only purpose is to group together other systemd units through a chain of dependencies.

For instance, there is a `graphical.target` that denotes when the system's graphical session is ready. Units that are required to start in order to achieve the necessary state have `WantedBy=` or `RequiredBy=` `graphical.target` in their configuration. Units that depend on `graphical.target` can include `Wants=`, `Requires=`, or `After=` in their configuration to make themselves available at the correct time.

A target can have a corresponding directory whose name has the syntax `target_name.target.wants` (e.g. `graphical.target.wants`), located in `/etc/systemd/system`. When a symlink to a service file is added to this directory, that service becomes a dependency of the target.

When you enable a service (using `systemctl enable`), symlinks to the service are created inside the `.target.wants` directory for each target listed in that service's `WantedBy=` configuration. This is actually how the `WantedBy=` option is implemented.

Getting and Setting the Default Target

To get the default target for your system –the end goal of the chain of dependencies– issue the `get-default` command:

```
systemctl get-default
```

If you would like to change the default target for your system, issue the `set-default` command:

```
sudo systemctl set-default multi-user.target
```

Listing Targets

To retrieve a list of available targets, use the `list-unit-files` command and filter by target:

```
systemctl list-unit-files --type=target
```

To list all currently active targets, use the `list-units` command and filter by target:

```
systemctl list-units --type=target
```

Changing the Active Target

To change the current active target, issue the `isolate` command. This command starts the isolated target with all dependent units and shuts down all others. For instance, if you wanted to move to a multi-user command line interface and stop the graphical shell, use the following command:

```
sudo systemctl isolate multi-user.target
```

However, it is a good idea to first check on the dependencies of the target you wish to isolate so you do not stop anything important. To do this, issue the `list-dependencies` command:

```
systemctl list-dependencies multi-user.target
```

Rescue Mode

When a situation arises where you are unable to proceed with a normal boot, you can place your system in rescue mode. Rescue mode provides a single-user interface used to repair your system. To place your system in rescue mode, enter the following command:

```
sudo systemctl rescue
```

This command is similar to `systemctl isolate rescue`, but will also issue a notice to all other users that the system is entering rescue mode. To prevent this message from being sent, apply the `--no-wall` flag:

```
sudo systemctl rescue --no-wall
```

Emergency Mode

Emergency mode offers the user the most minimal environment possible to salvage a system in need of repair, and is useful if the system cannot enter rescue mode. For a full explanation of emergency mode, refer to the RedHat Customer Portal page (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-targets#sect-Managing_Services_with_systemd-Targets-Emergency). To enter emergency mode, enter the following command:

```
sudo systemctl emergency
```

This command is similar to `systemctl isolate emergency`, but will also issue a notice to all other users that the system is entering emergency mode. To prevent this message, apply the `--no-wall` flag:

```
sudo systemctl emergency --no-wall
```

More Shortcuts

`systemctl` allows users the ability to halt, shutdown and reboot a system.

To halt a system, issue the following command:

```
sudo systemctl halt
```

To shutdown a system, use:

```
sudo systemctl shutdown
```

Finally, to reboot a system, enter the following command:

```
sudo systemctl reboot
```

Similar to the `emergency` and `rescue` commands, these commands will issue a notice to all users that the system state is changing.

More Information

You may wish to consult the following resources for additional information on this topic. While these are provided in the hope that they will be useful, please note that we cannot vouch for the accuracy or timeliness of externally hosted materials.

- Systemctl man page (https://www.freedesktop.org/software/systemd/man/systemctl.html)
- Creating and modifying systemd unit files (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-unit_files)
- Working with systemd targets (https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/system_administrators_guide/sect-managing_services_with_systemd-targets)

Join our Community

Find answers, ask questions, and help others. (https://www.linode.com/community/questions/)

comments powered by Disqus (http://disqus.com)