Linux Academy

Hands-on Lab

Change Runlevels and Boot Targets on a Systemd System

Contents

Lab Connection Information

- Labs may take up to five minutes to build
- The IP address of your server is located on the Hands-on Lab page
- Username: linuxacademy
- Password: 123456

Related Courses

LPIC-1: System Administrator - Exam 101

Related Videos

Boot Process - Power on to System Prompt

Boot Process - System Targets and Systemd

Change Runlevels/ Boot Targets and Shutdown or Reboot System

Need Help?

Linux Academy Community

... and you can always send in a support ticket on our website to talk to an instructor!

In this lab, we review system runlevels, learn how Systemd uses targets and units to manage runlevel environments, and practice changing runlevels through the use of the `systemctl` command.

Systemd Runlevel Targets

Log in to the server using the credentials provided on the Hands-on Lab page. Become *root* by using `sudo su -`.

In contrast to the startup scripts and `inittab` of Sysvinit, Systemd uses a system of targets and units to determine runlevels and define any services launched or killed while within a certain operating state.

Instead of scripts, units -- defined as `servicename.service` within the `/usr/lib/systemd/system` directory -- list the dependencies, potential conflicts, and information regarding what needs to be run before or after a service launch. These units are then referenced in targets, such as the `default.target`, which is used in place of the `/etc/inittab`.

The targets directory is located at `/usr/lib/systemd/system`. Switch to this directory now and list the available targets:

```
[root@ip] cd /usr/lib/systemd/system
[root@ip] ls -al
...
lrwxrwxrwx.  1 root root    15 Mar 10 21:20 runlevel0.target →
poweroff.target
lrwxrwxrwx.  1 root root    13 Mar 10 21:20 runlevel1.target → rescue.
target
drwxr-xr-x.  2 root root    49 Mar 10 21:20 runlevel1.target.wants
lrwxrwxrwx.  1 root root    17 Mar 10 21:20 runlevel2.target → multi-
user.target
drwxr-xr-x.  2 root root    49 Mar 10 21:20 runlevel2.target.wants
lrwxrwxrwx.  1 root root    17 Mar 10 21:20 runlevel3.target → multi-
user.target
drwxr-xr-x.  2 root root    49 Mar 10 21:20 runlevel3.target.wants
lrwxrwxrwx.  1 root root    17 Mar 10 21:20 runlevel4.target → multi-
user.target
drwxr-xr-x.  2 root root    49 Mar 10 21:20 runlevel4.target.wants
lrwxrwxrwx.  1 root root    16 Mar 10 21:20 runlevel5.target →
graphical.target
drwxr-xr-x.  2 root root    49 Mar 10 21:20 runlevel5.target.wants
lrwxrwxrwx.  1 root root    13 Mar 10 21:20 runlevel6.target → reboot.
target
...
```

Here we can see the runlevel targets available, as well as a number of previously-defined target files. Determine which runlevel the default target is mapped to:

```
[root@ip] ls -al default.target
lrwxrwxrwx. 1 root root 16 Mar 10 21:20 default.target → graphical.
```

```
target
```

Currently, it is mapped to `graphical.target`, which is our runlevel 5 target.

View the `graphical.target` file:

```
[root@ip] cat graphical.target
[Unit]
Description=Graphical Interface
Documentation=man:systemd.special(7)
Requires=multi-user.target
Wants=display-manager.service
Conflicts=rescue.service rescue.target
After=multi-user.target rescue.service rescue.target display-manager.
service
AllowIsolate=yes
```

As we can see, the file is a text file of information instead of a script. Containing the description, any needed documentation, requirements, wants, conflicts, and services to run after, the file is a basic list of key-value pairs. The keys are fairly easy to understand: The "Description" is a basic description of the target, "Documentation" provides information on how to receive more information, "Requires" defines which targets or units *must* run before using this target, while "Wants" lists which targets or services to run after the required services, "Conflicts" notes what services cannot be running for the target to work, "After" shows which targets and services can run after the runlevel 5 services have started, and "AllowIsolate" denotes whether or not a system can be used with the `systemctl isolate` command.

This creates a hierarchical approach to handling running services on runlevels. For example, we know that our graphical target requires the `multi-user.target` to work:

```
[root@ip] cat multi-user.target
[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

This, in turn, requires `basic.target`:

```
[root@ip] cat basic.target
[Unit]
Description=Basic System
Documentation=man:systemd.special(7)
Requires=sysinit.target
After=sysinit.target
Wants=sockets.target timers.target paths.target slices.target
After=sockets.target paths.target slices.target
```

Which needs the sysinit.target:

```
[root@ip] cat sysinit.target
[Unit]
Description=System Initialization
Documentation=man:systemd.special(7)
Conflicts=emergency.service emergency.target
Wants=local-fs.target swap.target
After=local-fs.target swap.target emergency.service emergency.target
```

With no requirements, this is the top-level target for the graphical.target.

So, once booted, the system looks for the default.target file and goes through all of the listed dependencies, requires, and wants of the target, applying them in the appropriate order based upon the target files.

Changing Boot Targets and Runlevels

Note: All Sysvinit commands from the "Change Runlevels and Boot Targets on a Sysvinit System" Hands-on Lab work on Systemd servers; however, this guide focuses on Systemd-specific commands.

Runlevels

Systemd uses the systemctl command to control the activation, starting, stopping, and management of services, as well as how the system itself is started, stopped, and rebooted.

For instance, we can use systemctl get-default to view the default target:

```
[root@ip] systemctl get-default
multi-user.target
```

We can also set the default target:

```
[root@ip] systemctl set-default graphical.target
Removed symlink /etc/systemd/system/default.target.
Created symlink from /etc/systemd/system/default.target to /usr/lib/
systemd/system/graphical.target.
```

Should we need to list all available targets, we can use:

```
[root@ip] systemctl list-units --type=target
UNIT                   LOAD   ACTIVE SUB    DESCRIPTION
basic.target           loaded active active Basic System
cloud-config.target    loaded active active Cloud-config availability
cryptsetup.target      loaded active active Encrypted Volumes
```

```
getty.target           loaded active active Login Prompts
local-fs-pre.target    loaded active active Local File Systems (Pre)
local-fs.target        loaded active active Local File Systems
multi-user.target      loaded active active Multi-User System
network-online.target  loaded active active Network is Online
network.target         loaded active active Network
nss-lookup.target      loaded active active Host and Network Name
Lookups
nss-user-lookup.target loaded active active User and Group Name Lookups
paths.target           loaded active active Paths
remote-fs.target       loaded active active Remote File Systems
slices.target          loaded active active Slices
sockets.target         loaded active active Sockets
swap.target            loaded active active Swap
sysinit.target         loaded active active System Initialization
timers.target          loaded active active Timers
```

`systemctl` also allows us to change runlevels, using the `isolate` property:

```
[root@ip] systemctl isolate multi-user.target
```

This is effectively the same as the using the `init 3` command, but Systemd specific.

Shutdown

Note: Running any of these commands on your lab server will end the lab.

There are also specific Systemd shutdown commands. For example, to reboot:

```
[root@ip] systemctl reboot
```

And to power off:

```
[root@ip] systemctl poweroff
```

We can also use halt:

```
[root@ip] systemctl halt
```

And shutdown:

```
[root@ip] systemctl shutdown
```

This lab is now complete!