# How to Auto-Start Services on Boot in RHEL/CentOS 7?



By [Ankush Thakur](https://geekflare.com/author/ankush/) on August 2, 2020

Posted in [Sysadmin](https://geekflare.com/category/sysadmin/)

* **Wondering how to manage services in the background or on boot?**

The mechanism for managing and starting processes on boot has been changed. Until RHEL/CentOS  6.x, you would have created a script in /etc/init.d/ and enabled with the help of chkconfig but things are different on [RHEL 7](https://click.linksynergy.com/deeplink?id=jf7w44yEft4&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Flinux-administration-bootcamp%2F).

It’s replaced by systemd and since it is more or less the default process manager on major Linux versions, [System Admin](https://geekflare.com/linux-books-videos/) versed in other flavors will feel right at home. In this article, we will explore what systemd is, what the reasons to the switch were, and how to use systemd to set up, run and manage background processes with it.

### What is systemd?

Since every process in Linux is transparently visible, let’s see where systemd is lurking. On my system, I get the following:

~$ ps -ef | grep systemd

root 1 0 0 Nov11 ? 00:01:02 /lib/systemd/systemd --system --deserialize 22

message+ 768 1 0 Nov11 ? 00:05:46 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only

root 805 1 0 Nov11 ? 00:10:22 /lib/systemd/systemd-logind

ankush 1132 1 0 Nov11 ? 00:00:00 /lib/systemd/systemd --user

ankush 1176 1132 0 Nov11 ? 00:04:50 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only

ankush 9772 20029 0 21:11 pts/6 00:00:00 grep --color=auto systemd

systemd+ 17298 1 0 Nov19 ? 00:00:12 /lib/systemd/systemd-resolved

systemd+ 17303 1 0 Nov19 ? 00:00:00 /lib/systemd/systemd-timesyncd

root 17307 1 0 Nov19 ? 00:00:02 /lib/systemd/systemd-journald

root 18182 1 0 Nov19 ? 00:00:00 /lib/systemd/systemd-udevd

I bet you noticed it instantly. the first process in the listing was launched as the user root, and has the pid 1.

Sure enough, this was the first process that the system launched upon boot. Say hello to systemd. 🙂

So, quite simply, systemd is the “mother” process that launches, manages, and terminates other processes in the system, besides providing info about their logging, filesystem statuses, etc.

A note on the name, though. The name is indeed systemd and not System D or anything else. The “d” stands for daemon, a standard Linux process that works (lurks?) in the background and isn’t attached to any terminal session.

### Why RHEL switched to systemd?

As we already discussed, [systemd](https://geekflare.com/manage-systemd-services-with-systemctl/) is a system and process manager, and in RHEL 7, replaces the well-known program Upstart. Why did RHEL take this decision? Well, there are very good reasons for this, so let’s take a quick look.

Parallel service initialization

Unlikes the SysV init program, systemd is capable of launching services in parallel. The init program, by contrast, launches them one by one. In an era where even mobile devices have multi-core CPUs, the lack of parallel initialization is a big turn-off.

Dynamic (hot) service management

If you’ve noticed that USB drives need to be explicitly mounted on earlier Fedora systems while they would automatically pop open on Ubuntu and similar distributions, the reason is systemd. It is able to detect live changes in hardware and terminate/launch services as needed. Some can argue that it’s unnecessary, but to many, anything that reduces cognitive burden is most welcome.

Deferred service launch

systemd makes boot times shorter as it’s able to defer service launch to when it’s actually needed. A simple example is network file system-related services. If there’s no networked disk available, it doesn’t make sense to have a service up and running.

Faster process communication

The parallel capabilities of systemd carry over to inter-process communication. systemd is able to offer parallel access to sockets and system bus, significantly reducing process wait times for communication resources.

Automatic restart

If a service crashes, systemd can detect that and attempt to restart it. Most of the times, a simple restart is all that is needed for an application to begin functioning again, unless there are more fundamental issues.

Anyway, systemd makes the life of a sysadmin easier here.

### systemd in RHEL7 – What changes for Sysadmins?

If you have a nagging feeling that systemd isn’t going to be all bells and whistles, you’re right. There are a few significant incompatibilities that can catch RHEL sysadmin by surprise. Let’s take a quick look.

Limited runlevel support

systemd has a pretty shabby recognition of and support for runlevels. Not all the runlevels are supported, and for some targets there might even be none. In such cases, systemd returns “N” as a response to the runlevel commands, signifying that it has no corresponding runlevel to this target. All in all, Red Hat advises us to not use (!) the runlevel commands.

No custom commands

This one’s going to hurt. One big plus with SysV was the ability to define custom commands to provide better functionality for managing processes. With systemd, there’s no such option and you’re stuck with start, stop, status, restart, etc.

Family-only and non-interactive

systemd (of course) keeps a track of the processes it has launched and stores their PIDs. The challenge, however, is that systemd cannot deal with processes not launched by it. Further, it’s not possible for a user to interact with a process started by systemd. All output goes to /dev/null, effectively putting a stop on any hopes you might’ve had of capturing output.

No context

Unlike init services, those launched by systemd do not inherit any environment from any users in the system. In other words, information like PATH and other system variables aren’t available, and every new process is launched in an empty context.

If this list of limitations makes you cry, again, you’re not alone. systemd has been a polarizing force in the Linux world, and Googling on “systemd sucks” will unearth plenty of reading material. 😉

### How to Start Service Automatically When Down?

Here’s a pretty common use case in deployments. We need to daemonize a program in a language that doesn’t have long-running processes: PHP! Let’s assume I write a PHP script to handle incoming websocket connections (we built a chatting app, after all!) and the script is placed at /home/ankush/chat\_server/index.php.

Since websocket connections can hit the server any time, this process needs to be up at all times and monitor incoming connections. We can’t have the traditional PHP lifecycle here because WebSockets are stateful connections, and if the script dies, the connection is a list. Anyway, enough on websockets; let’s see how we’ll go about daemonizing this script via systemd.

All systemd services reside in /etc/systemd/system,  so let’s create a file there to describe our websocket server script. Assuming you are logged in as a root user.

# vi /etc/systemd/system/chat\_server.service

and then the following is needed.

[Unit]

Description=Chat Server Service

After=network.target

[Service]

Type=simple

User=ankush

ExecStart=php /home/ankush/chat\_server/index.php

Restart=on-abort

[Install]

WantedBy=multi-user.target

Save the file and the next step is to reload the systemd daemon

# systemctl daemon-reload

and to start the service we just created:

# systemctl start chat\_server

If you see no errors, that was it!

Let’s also quickly take a look at what the various directives in the file mean:

* The [Unit] part defines a new service unit for systemd. In the systemd parlance, all services are known as service units.
* The After directive (predictably) tells systemd to launch this service only after the networking services is launched (otherwise, who will do the lower-level handling of socket connections?!).
* The Type=simple tells systemd that this service isn’t supposed to fork itself. In other words, only one instance will be running any given time.
* User=ankush means this service will run as the user “ankush”. We could change this to “root”, but it’s highly unrecommended from a security perspective.
* ExecStart, as you can tell, is the actual command to run.
* Restart=on-abort means that the service should be restarted when it aborts. In PHP, long-running processes leak memory and eventually explode, so this is needed.
* The WantedBy= directive tells systemd which target (think of groups) this service is part of. This results in symbolic links being created inside that target to point to the service.

Generally, this much is enough for running background processes using systemd in RHEL 7.

### More option for Restart logic

In the above example, I’ve configured Restart=on-abort but that’s not the only option. There are more and choose based on the requirement.

* on-failure – will be restarted when unclean exit code or signal
* always – restart when found down, clean or unclean signal
* on-abnormal – unclean signal, watchdog or timeout
* on-success – only when it was stopped by a clean signal or exit code

### Configuring Service to Start on Boot

Once you are satisfied with the script and ensure it works, next you want to configure that so it trigger on boot and start.

Go to /etc/systemd/system and execute below enable command (don’t forget to change the .service file name with the one you have)

# systemctl enable chat\_server.service

You’ll see a confirmation that it has created a symlink.

Created symlink from /etc/systemd/system/multi-user.target.wants/chat\_server.service to /etc/systemd/system/chat\_server.service.

Restart your server and you should see service starts on the boot.

That was easy! Isn’t it?

## Help! I’m massively invested in Upstart. 🙁

I understand you trust me, your case is the norm rather than the exception. RHEL has been using Upstart for so long that the switch almost feels like a betrayal. But hey, systems keep changing, and we shouldn’t squabble over trifles. Red Hat recognizes that many people are stuck with older versions, and have created a [migration guide](https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/migration_planning_guide/index) that you should definitely refer to.

One saving grace in all of this is that systemd is compatible with the SysV init scripts, so for the most part, you’ll simply need to move your files and get the same services running.

Interested in learning more about Linux Administration and Troubleshooting? Check out this [online course](https://click.linksynergy.com/deeplink?id=jf7w44yEft4&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Flinux-administration%2F).