Linux Commands

How to Create a Service File in Linux

7 months ago • by Sam U

The systemd init system is now a part of almost all Linux distributions. As an administrator or developer, you create services that systemd can manage. For example, if you want to launch services on boot or want to manually control them, then it can be done through a custom service file.

In this tutorial, I will go through how to create a systemd service file on Linux.

What is a Service File

Before going any further, let's understand what a systemd service file is and how it's created on Linux.

A systemd service file contains instructions set for systemd to manage the service. It typically contains three sections:

- Unit
- Service
- Install

The **Unit** section contains basic information about the service such as a short description, documentation pages, and a path to dependencies. The **Install** section is optional, but typically it manages at what system state the service should be enabled.

The **Service** section is normally sandwiched between the Unit and Install section. It primarily contains the type of the service and path of the executables that essentially are commands to execute on invoking the service by systemd.

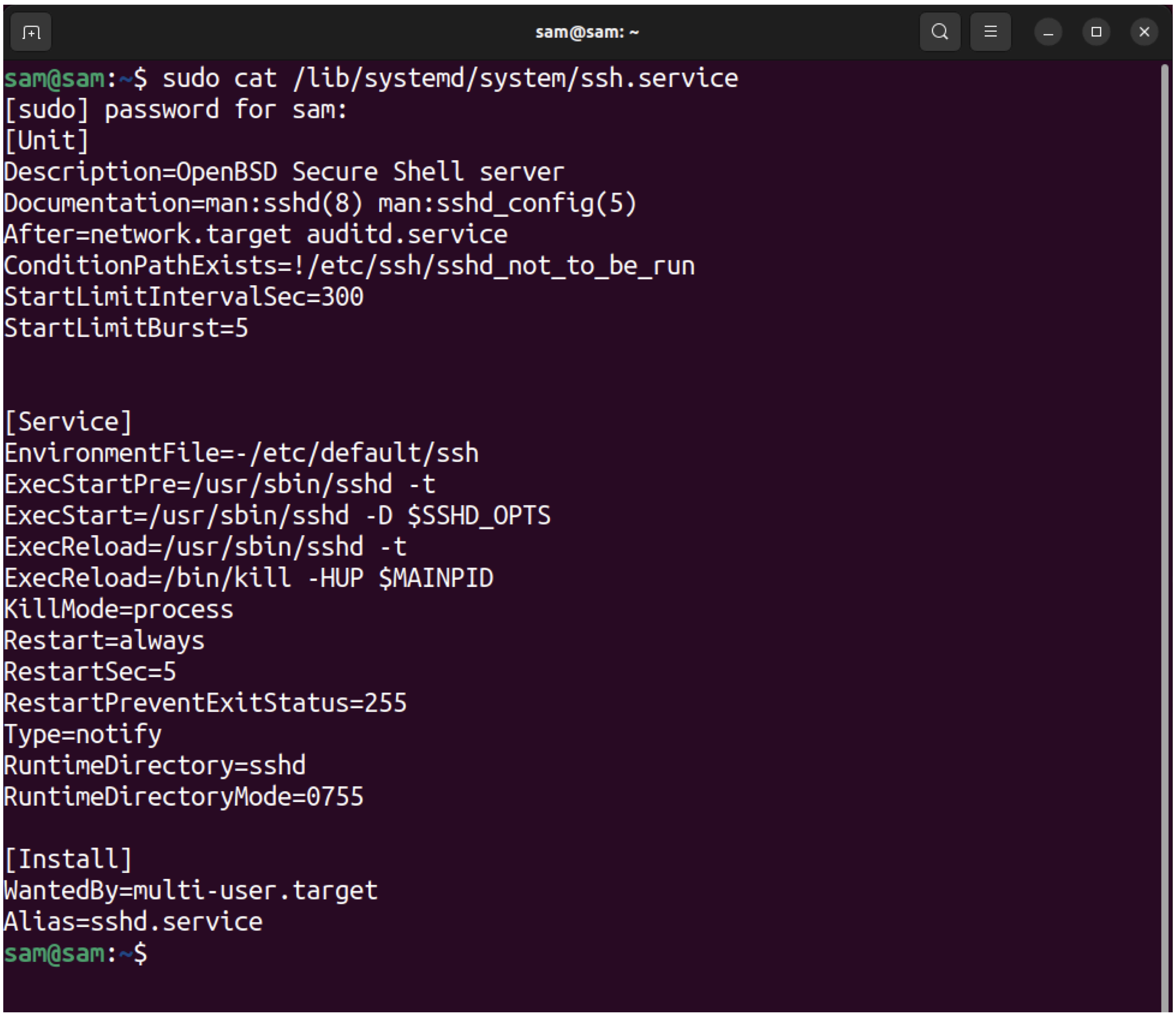
Here is what a typical service file structure looks like.

```
[Unit]
Directive1=instruction
Directive2=instruction
```

```
…
[Service]
Directive1=instruction
Directive2=instruction
…
[Install]
Directive1=instruction
Directive2=instruction
…
```

Here, directives are parameters that take their respective input. For example, the **Description** directive takes a string of the name of the service. In a similar manner, **ExecStart** takes into account the complete path of the executable.

A typical service file of **ssh.service** is given below.



```
sam@sam:~$ sudo cat /lib/systemd/system/ssh.service
[sudo] password for sam:
[Unit]
Description=OpenBSD Secure Shell server
Documentation=man:sshd(8) man:sshd_config(5)
After=network.target auditd.service
ConditionPathExists=!/etc/ssh/sshd_not_to_be_run
StartLimitIntervalSec=300
StartLimitBurst=5


[Service]
EnvironmentFile=-/etc/default/ssh
ExecStartPre=/usr/sbin/sshd -t
ExecStart=/usr/sbin/sshd -D $SSHD_OPTS
ExecReload=/usr/sbin/sshd -t
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=always
RestartSec=5
RestartPreventExitStatus=255
Type=notify
RuntimeDirectory=sshd
RuntimeDirectoryMode=0755

[Install]
WantedBy=multi-user.target
Alias=sshd.service
sam@sam:~$
```

How to Create a Service File

To create a systemd service, it is necessary to understand the key directives. In this guide, I will cover the essential directives that can help you to create a fully functional service file.

Creating a service file involves multiple steps, let's begin with creating the script file.

Note: To continue with the method mentioned below, you must have root privileges.

1. Creating a Script

The initial step involves the creation of the code that will be executed when the service begins its operation. For this tutorial, I am creating a bash script that will store the uptime of the Linux system and memory usage.

Let's create a script in the current directory with the name of **myscript.sh** using nano editor.

```
sudo nano myscript.sh
```

Now, add the script given below in the file and save it by pressing **ctrl+x** and then **y**.

```
#!/bin/bash
echo ">>Here is the Uptime of your System<<" > home/sam/myfile.txt
uptime >> home/sam/myfile.txt
echo ">>Here is the Memory Usage of your System<<" >> /home/sam/myfile.txt
free -m >> home/sam/myfile.txt
sleep 60
```

The script contains a couple of echo strings and **uptime** and **free** commands.

The **uptime** command on Linux is used to print how long a system has been running, and how many users are connected with an average system load of past 1, 5, and 15 minutes.

The **free** command is used to print the memory usage of the system, while the **-m** flag is used to print the output in **MBs**.

To save information in a text file, we use special operators called redirection operators. The **>** operator is used to insert text to the mentioned text file. However, if the file does not exist already, it will be created. While the **>>** operator is used to append the text in the file. The **sleep** command is utilized to maintain the service's activity for a minimum duration of one minute.

Now, make the script executable by granting the necessary permissions.

```
sudo chmod +x myscript.sh
```

Script now has the execution permission, let's proceed to the next section.

Note: To make the service file error-free, use the absolute path of the file in the bash script.

2. Creating a .service File

Next, create a service file with the **.service** extension. The service file must be created in the **/etc/systemd/system** directory. First, navigate to this directory using the **cd** command.

```
cd /etc/systemd/system
```

You can create the service file in any directory, and later move that file to this directory, nevertheless.

I am creating a service file with **myservice.service** name.

```
sudo nano myservice.service
```

Now, add the following lines to the file.

```
[Unit]
Description=My Service
[Service]
Type=simple
ExecStart=/bin/bash /home/sam/script.sh
Restart=on-failure
[Install]
WantedBy=multi-user.target
```

Note that **[Unit], [Service],** and **[Install]** are **case-sensitive**. The service file will not work if any of them is mentioned incorrectly, such as [UNIT], or [SERVICE].

The service name is specified as **My Service** in the **Description** directive of the **[Unit]** section.

The **Type** of the service is **simple** in the **[Service]** section, which is the default type. **Forking**, **one-shot**, **notify**, **dbus**, and idle are some of the other types.

If you want to make the service user-specific, then the **User** directive can also be used with the username of the user. Using this directive will make the service user permission dependent.

While the **ExecStart** directive contains the full path of the executable. In the above example, the script file **myscript.sh** is stored in the **/home/sam/** directory. This directive in fact manages what to execute when a service is invoked by systemd. If the command's full path is not specified, it will automatically be resolved to fix absolute paths such as **/usr/local/bin**, **/usr/bin/,** and **/bin**. It is perfectly fine to use the executable name as long as they are in the standard command directories, however, mention the absolute path otherwise. Note that Multiple commands can also be used that are separated by semicolon (;).

The **[Install]** section is optional; however, it indicates how the service is enabled. The **WantedBy** directive takes the run-level target files as parameters. Various target files indicated different run-levels of the system such as **poweroff**, **rescue**, **multi-user**, **graphical**, and **reboot**.

The **multi-user.target** means the service will be enabled when the system is in a state to allow multi-user non-graphical sessions.

3. Activating the Service

To activate the service, first, reload the systemd configurations using **systemctl** utility.

```
sudo systemctl daemon-reload
```

Next, activate the service again using **systemctl** command with **enable**.

```
sudo systemctl enable myservice.service
```

To verify, check the status of the service using the **systemctl status** command.

The service is successfully running.

Now, let’s read the text file **myfile.txt** the service created in the **/home** directory.

How to Create a Systemd Service File for Normal User

The procedure of creating the service file for a normal user is similar to the method of creating a service file by the administrator. However, the directory to save the service file for normal users is different. Normal users must place their service files in the **~/.config/systemd/user**. This directory must be created using the **mkdir** command.

```
mkdir ~/.config/systemd/user
```

To activate the service by a normal user **–user** command is inserted with **systemctl** instead of **sudo**.

```
systemctl --user daemon-reload

systemctl --user enable SERVICE-NAME.service

systemctl --user status SERVICE-NAME.service
```

The **–user** option is used to manage the user’s systemd service files.

How to Remove Service File

To remove the service file, firstly, the service needs to be stopped.

```
sudo systemctl stop SERVICE-NAME.service
```

Check the status using **systemctl status** command to know if the service is stopped or not. Then remove the service file using the **rm** command.

```
sudo rm /etc/systemd/system/SERVICE-NAME.service
```

Now, reload the **systemd** configuration.

```
sudo systemctl daemon-reload
```

Conclusion

The custom systemd service is advantageous in various scenarios. In this guide, we learned how to create a custom systemd service file for a system administrator and how normal users can create a service file. Furthermore, we also see the procedure of removing the service file.

ABOUT THE AUTHOR

Sam U

I am a professional graphics designer with over 6 years of experience. Currently doing research in virtual reality, augmented reality and mixed reality.
I hardly watch movies but love to read tech related books and articles.

View all posts

RELATED LINUX HINT POSTS