Logfile management with logrotate

Logfiles are essential components of a Linux system to keep track of events happening on different services on the system. This helps to debug issues as well as provide statistics on the live machine. Management of logfiles is required because as time passes, the size of a logfile gets bigger and bigger. Therefore, we use techniques called **rotation** such that we limit the size of the logfile and if the logfile reaches a size beyond the limit, it will strip the logfile with that size and store the older entries in the logfile archived in log directories. Hence, older logs can be stored and kept for future references. Let's see how to rotate logs and store them.

Getting ready

logrotate is a command every Linux system admin should know. It helps to restrict the size of the logfile to the given SIZE. In a logfile, the logger appends information to the log file. Hence, the recent information appears at the bottom of the log file. logrotate will scan specific logfiles according to the configuration file. It will keep the last 100 kilobytes (for example, specified SIZE = 100 k) from the logfile and move rest of the data (older log data) to a new file logfile_name.1 with older entries. When more entries occur in the logfile (logfile_name.1) and it exceeds the SIZE, it updates the logfile with recent entries and creates logfile_name.2 with older logs. This process can easily be configured with logrotate. logrotate can also compress the older logs as logfile_name.1.gz, logfile_name2.gz, and so on. The option of whether older log files are to be compressed or not is available with the logrotate configuration.

How to do it...

logrotate has the configuration directory at /etc/logrotate.d. If you look at this directory by listing its contents, many other logfile configurations can be found.

We can write our custom configuration for our logfile (say /var/log/program.log) as follows:

```
$ cat /etc/logrotate.d/program
/var/log/program.log {
missingok
notifempty
size 30k
  compress
weekly
  rotate 5
create 0600 root root
}
```

Now the configuration is complete. `/var/log/program.log` in the configuration specifies the logfile path. It will archive old logs in the same directory path.

How it works...

Let's see what each of the parameters in the configuration mean:

Parameter	Description
`missingok`	Ignore if the logfile is missing and return without rotating the log.
`notifempty`	Only rotate the log if the source logfile is not empty.
`size 30k`	Limit the size of the logfile for which the rotation is to be made. It can be 1 M for 1 MB.
`compress`	Enable compression with gzip for older logs.
`weekly`	Specify the interval at which the rotation is to be performed. It can be weekly, yearly, or daily.
`rotate 5`	It is the number of older copies of logfile archives to be kept. Since 5 is specified, there will be `program.log.1.gz`, `program.log.2.gz`, and so on up to `program.log.5.gz`.
`create 0600 root root`	Specify the mode, user, and the group of the logfile archive to be created.

The options specified in the table are optional; we can specify the required options only in the `logrotate` configuration file. There are numerous options available with `logrotate`, please refer to the man pages (`http://linux.die.net/man/8/logrotate`) for more information on `logrotate`.

Logging with syslog

Usually, logfiles related to different daemons and applications are located in the `/var/log` directory, as it is the common directory for storing log files. If you read through a few lines of the logfiles, you can see that lines in the log are in a common format. In Linux, creating and writing log information to logfiles at `/var/log` are handled by a protocol called **syslog**, handled by the `syslogd` daemon. Every standard application makes use of syslog for logging information. In this recipe, we will discuss how to make use of `syslogd` for logging information from a shell script.