How to Manage Log Rotation in Linux Using Logrotate

Log rotation is an essential process in system administration, especially in a Linux environment. It helps manage log files by archiving and compressing them, thus preventing log files from consuming excessive disk space. In this article, we will explore how to use the logrotate utility to automate log rotation in Linux.

What is Logrotate?

logrotate is a utility designed to manage the automatic rotation and compression of log files. It is typically run by the cron daemon and is highly configurable, allowing system administrators to specify how often logs should be rotated, how many old log files should be kept, and whether the rotated logs should be compressed.

Installing Logrotate

Most Linux distributions come with logrotate pre-installed. However, if it is not installed on your system, you can install it using your package manager.

For Debian-based systems (e.g., Ubuntu):

```
sudo apt-get update
sudo apt-get install logrotate
```

For Red Hat-based systems (e.g., CentOS, Fedora):

```
sudo yum install logrotate
```

Basic Configuration

The primary configuration file for logrotate is /etc/logrotate.conf. This file can include directives and settings that apply globally or to specific log files. Additionally, individual configuration files can be placed in the /etc/logrotate.d/ directory.

Here is an example of a basic configuration in /etc/logrotate.conf:

```
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
```

```
rotate 4

# create new (empty) log files after rotating old ones
create

# use date as a suffix of the rotated file
dateext

# compress log files
compress

# include all configurations in /etc/logrotate.d
include /etc/logrotate.d
```

Creating a Logrotate Configuration for a Specific Log File

To create a custom log rotation configuration for a specific log file, you can create a new file in the /etc/logrotate.d/ directory. For example, to manage the rotation of /var/log/myapp.log, create a file named /etc/logrotate.d/myapp with the following content:

```
/var/log/myapp.log {
    daily
    rotate 7
    compress
    missingok
    notifempty
    create 0640 root adm
    sharedscripts
    postrotate
        /usr/bin/systemctl reload myapp.service > /dev/null 2>/dev/null ||
 true
    endscript
}
```

Explanation of Directives

- daily: Rotate the log file daily.
- rotate 7: Keep 7 days of log files before deleting the oldest one.
- compress: Compress the rotated log files to save space.
- missingok: Do not issue an error message if the log file is missing.
- notifempty: Do not rotate the log if it is empty.
- create 0640 root adm: Create a new log file with the specified permissions and ownership.
- sharedscripts: Run the postrotate script only once, not once per log file.
- postrotate ... endscript: Run the specified commands after the log file is rotated. In this case, it reloads the myapp service to ensure it continues logging to the new file.

Manually Testing Logrotate Configuration

To manually test your logrotate configuration, you can use the following command:

```
sudo logrotate -d /etc/logrotate.conf
```

The -d option runs logrotate in debug mode, showing what actions would be taken without actually performing them. To execute the rotation, use:

```
sudo logrotate -f /etc/logrotate.conf
```

The -f option forces the rotation regardless of the log file's age.

Conclusion

Managing log rotation is crucial for maintaining a healthy Linux system. By using logrotate, you can automate the process, ensuring that log files are rotated, compressed, and managed efficiently. This not only helps in conserving disk space but also in keeping the system organized and logs accessible for troubleshooting.