**What is logrotate?**

It may surprise you to learn that logrotate is a program used to rotate logs. It’s true! The system usually runs logrotate once a day, and when it runs it checks rules that can be customized on a per-directory or per-log basis.

“Log rotation” refers to the practice of archiving an application’s current log, starting a fresh log, and deleting older logs. And while we’re explaining things, a “log” is a file where an application stores information that might be useful to an administrator or developer - what it’s been doing, what errors it’s run into, that sort of thing. So logs are good, you just usually don’t want to keep a ton of them around. That’s where logrotate comes in.

**The importance of log rotation**

Logs are wonderful things when you want to track usage or troubleshoot an application. Unfortunately the more information that gets logged, the more disk space the log uses. Over time it can really add up.

A log left unrotated can grow to a pretty unwieldy size. Running out of disk space because of a giant log is a problem of course, but a huge log file can also slow down the process of resizing or backing up your virtual server. Another practical consideration is that it’s hard to look for a particular event if you have a million log entries to skim through. So on the whole it’s a good idea to keep log files down to a manageable size, and to prune them when they get too old to be of much use.

Fortunately logrotate makes log rotation easy.

**How it works**

The system runs logrotate on a schedule, usually daily. In fact, on most distributions you’ll find the script that runs logrotate daily at:

/etc/cron.daily/logrotate

Some distributions use a variation on that theme. For example, on Gentoo the logrotate script is:

/etc/cron.daily/logrotate.cron

If you want logrotate to run more often (for hourly log rotation, for example) you’ll need to look into using cron to run logrotate through a script in /etc/cron.hourly.

When logrotate runs it reads its configuration files to determine where to find the log files it needs to rotate, and to check on details like how often the files should be rotated and how many archived logs to keep.

**logrotate.conf**

The main logrotate configuration file is located at:

/etc/logrotate.conf

If you look inside that file you’ll see the default parameters logrotate uses when it rotates logs. The file is nicely commented, so skim it to see how things are set up. We’ll talk about several of the specific commands in that file shortly.

Note that one line reads:

include /etc/logrotate.d

That’s where we’ll find most of the application-specific configuration files.

**logrotate.d**

Take a look inside the directory where you’ll store application-specific log settings:

ls /etc/logrotate.d

Depending on how much you’ve installed on your server there may be no files in this directory, or there may be several. In general, applications that are installed through your package manager will also create a config file in /etc/logrotate.d.

Most likely you will at least see a config file for your syslog service, which logrotate will read when it goes to rotate the system logs. If you look inside you’ll see an entry for various system logs along with some commands similar to what you saw in logrotate.conf.

**NOTE:** You won’t actually see an entry for a syslog service on versions of Ubuntu older than Karmic Koala (9.10). Prior to that release the system logs were rotated by a “savelog” command run from the “/etc/cron.daily/sysklogd” script.

**Inside an application file**

As an example, let’s take a look at the contents of a logrotate config file that might be put in place when you install apache on a Fedora system:

/var/log/httpd/\*log {

missingok

notifempty

sharedscripts

postrotate

/sbin/service httpd reload > /dev/null 2>/dev/null || true

endscript

}

We’ll look at what most of the specific directives in this file mean in a bit, but the short version is that when logrotate runs it will check for any files in /var/log/httpd that end in “log” and rotate them, so long as they aren’t empty. If it checks the httpd directory and doesn’t find any logfiles it won’t throw an error. Then it will run the command in the “postrotate/endscript” block (in this case, a command that will tell apache to restart), but only after it’s processed all the specified logs.

What you don’t see in that file are some settings you saw back in logrotate.conf. This is because the commands in logrotate.conf act as defaults for log rotation. You can specify different settings for any application where you want to override the defaults. For example, if you run a busy web server, you may want to include a “daily” command in apache’s config block so apache’s logs will rotate daily instead of the default weekly rotation.

That might be more clear if we talk about what some of the more commonly-used commands actually do in a logrotate config file. So let’s do that next.

**Configuration commands**

You can get a full list of commands used in logrotate configuration files by checking the man page:

man logrotate

We’ll go over more commonly-used commands here.

Remember, the config files for applications in /etc/logrotate.d inherit their defaults from the main /etc/logrotate.conf file.

**Log files**

A log file and its rotation behavior is defined by listing the log file (or files) followed by curly brackets. Most application configuration files will contain just one of these blocks, but it’s possible to put more than one in a file, or to add log file blocks to the main logrotate.conf file.

You can list more than one log file for a block either by using a wildcard in the name or by separating log files in the list with spaces. For example, to specify all files in the directory /var/foo that end in “.log”, as well as the file “/var/bar/log.txt”, you would set up the block like so:

/var/foo/\*.log /var/bar/log.txt {

blah blah blah

blah blah blah redux

}

Just not with as many blahs.

**Rotate count**

The “rotate” command determines how many archived logs will be kept around before logrotate starts deleting the older ones. For example:

rotate 4

That command tells logrotate to keep 4 archived logs at a time. If there are already four archived logs when the log is rotated again, the oldest one (the one with “.4” at the end, usually) will be deleted to make room for the new archive.

**Rotation interval**

You can specify a command that will tell logrotate how often to rotate a particular log. The possible commands include:

daily

weekly

monthly

yearly

If a rotation interval is not specified the log will be rotated whenever logrotate runs (unless another condition like “size” has been set).

If you want to use a time interval other than the keywords listed here you’ll have to get clever with cron and a separate config file. For example, if you wanted to rotate a particular log file hourly, you could create a file in “/etc/cron.hourly” (you may need to create that directory too) that would contain a line like:

/usr/sbin/logrotate /etc/logrotate.hourly.conf

Then put the configuration for that hourly run of logrotate (the log file location, whether or not to compress old files, and so on) into “/etc/logrotate.hourly.conf”.

**Size**

You can specify a file size that logrotate will check when determining whether or not to perform a rotation by using the “size” command. The format of the command tells logrotate what units you’re using to specify the size:

size 100k

size 100M

size 100G

The first example would rotate the log if it gets larger than 100 kilobytes, the second if it’s larger than 100 megabytes, and the third if it’s over 100 gigabytes. I don’t recommend using a limit of 100G, mind you, the example just got a little out of hand there.

The size command takes priority over and replaces a rotation interval if both are set.

**Compression**

If you want archived logfiles to be compressed (in gzip format) you can include the following command, usually in /etc/logrotate.conf:

compress

This is normally a good idea, since log files are usually all text, and text compresses very well. You might, however, have some archived logs you don’t want compressed, but still want compression to be on by default. In those cases you can include the following command in an application-specific config:

nocompress

One more command of note in regard to compression is:

delaycompress

This command can be useful if you want the archived logs to be compressed, but not right away. With “delaycompress” active an archived log won’t be compressed until the next time the log is rotated. This can be important when you have a program that might still write to its old logfile for a time after a fresh one is rotated in. Note that “delaycompress” only works if you also have “compress” in your config.

An example of a good time to use delaycompress would be when logrotate is told to restart apache with the “graceful” or “reload” directive. Since old apache processes would not be killed until their connections are finished, they could potentially try to log more items to the old file for some time after the restart. Delaying the compression ensures that you won’t lose those extra log entries when the logs are rotated.

**Postrotate**

The “postrotate” script is run by logrotate each time it rotates a log specified in a config block. You’ll usually want to use this to restart an application after the log rotation so the app can switch to a new log.

postrotate

/usr/sbin/apachectl restart > /dev/null

endscript

That “> /dev/null” bit at the end tells logrotate to pipe the command’s output to, well, nowhere. Otherwise the output of that command will be sent off to the console or the log or email or whatever, and in this case, you don’t really care about the output if everything restarted okay.

The “postrotate” command tells logrotate that the script to run will start on the next line, and the “endscript” command says that the script is done.

**Sharedscripts**

Normally logrotate will run the “postrotate” script every time it rotates a log. This is true for multiple logs using the same config block. So for example, a web server config block that refers to both the access log and the error log will, if it rotates both, run the “postrotate” script twice (once for each file rotated). So if both files are rotated, the web server will be restarted twice.

To keep logrotate from running that script for every log, you can include the command:

sharedscripts

That tells logrotate to wait until it’s checked all the logs for that config block before running the postrotate script. If one or both of the logs get rotated, the postrotate script still only gets run once. If none of the logs get rotated, the postrotate script won’t run at all.

**Where to go next**

You’ve seen an overview of what logrotate does and what kind of configuration options are available to you. You should be all set to go poking around in the existing configs and adapt them to your needs. To learn how to put an example config together (to rotate the logs for custom virtual hosts),and also cover some handy troubleshooting approaches, see [Sample logrotate configurations and troubleshooting](https://support.rackspace.com/how-to/sample-logrotate-configuration-and-troubleshooting).