DigitalOcean

Subscribe Share Contents

How To View and Configure Linux Logs on Ubuntu and Centos

19

Posted December 17, 2013 393k LOGGING LINUX BASICS CENTOS UBUNTU DEBIAN

By: Sadequl Hussain

Introduction

Linux system administrators often need to look at log files for troubleshooting purposes. In fact, this is the first thing any sysadmin would do.

Linux and the applications that run on it can generate all different types of messages, which are recorded in various log files. Linux uses a set of configuration files, directories, programs, commands and daemons to create, store and recycle these log messages. Knowing where the system keeps its log files and how to make use of related commands can therefore help save valuable time during troubleshooting.

In this tutorial, we will have a look at different parts of the Linux logging mechanism.

> ### Disclaimer
>
> The commands in this tutorial were tested in plain vanilla installations of CentOS 6.4, Ubuntu 12 and Debian 7.

Default Log File Location

The default location for log files in Linux is /var/log.

ROLL TO TOP

This is what I see in my CentOS system:

```
[root@TestLinux ~]# ls -l /var/log
```

```
total 1472
-rw-------. 1 root root   4524 Nov 15 16:04 anaconda.ifcfg.log
-rw-------. 1 root root  59041 Nov 15 16:04 anaconda.log
-rw-------. 1 root root  42763 Nov 15 16:04 anaconda.program.log
-rw-------. 1 root root 299910 Nov 15 16:04 anaconda.storage.log
-rw-------. 1 root root  40669 Nov 15 16:04 anaconda.syslog
-rw-------. 1 root root  57061 Nov 15 16:04 anaconda.xlog
-rw-------. 1 root root   1829 Nov 15 16:04 anaconda.yum.log
drwxr-x---. 2 root root   4096 Nov 15 16:11 audit
-rw-r--r--  1 root root   2252 Dec  9 10:27 boot.log
-rw-------  1 root utmp    384 Dec  9 10:31 btmp
-rw-------. 1 root utmp   1920 Nov 28 09:28 btmp-20131202
drwxr-xr-x  2 root root   4096 Nov 29 15:47 ConsoleKit
-rw-------  1 root root   2288 Dec  9 11:01 cron
-rw-------. 1 root root   8809 Dec  2 17:09 cron-20131202
-rw-r--r--  1 root root  21510 Dec  9 10:27 dmesg
-rw-r--r--  1 root root  21351 Dec  6 16:37 dmesg.old
-rw-r--r--. 1 root root 165665 Nov 15 16:04 dracut.log
-rw-r--r--. 1 root root 146876 Dec  9 10:44 lastlog
-rw-------  1 root root    950 Dec  9 10:27 maillog
-rw-------. 1 root root   4609 Dec  2 17:00 maillog-20131202
-rw-------  1 root root 123174 Dec  9 10:27 messages
-rw-------. 1 root root 458481 Dec  2 17:00 messages-20131202
-rw-------  1 root root   2644 Dec  9 10:44 secure
-rw-------. 1 root root  15984 Dec  2 17:00 secure-20131202
-rw-------  1 root root      0 Dec  2 17:09 spooler
-rw-------. 1 root root      0 Nov 15 16:02 spooler-20131202
-rw-------. 1 root root      0 Nov 15 16:02 tallylog
-rw-rw-r--. 1 root utmp  89856 Dec  9 10:44 wtmp
-rw-------  1 root root   3778 Dec  6 16:48 yum.log
```

Viewing Log File Contents

Here are some common log files you will find under /var/log:

- utmp
- dmesg
- messages
- maillog or mail.log
- spooler
- auth.log or secure

The wtmp and utmp files keep track of users logging in and out of the system. You cannot directly read the contents of these files using cat– there are specific commands for that.

We will now use some of these commands.

To see who is currently logged in to the Linux server, simply use the who command. This command gets its values from the /var/run/utmp file (for CentOS and Debian) or /run/utmp (for Ubuntu).

Here is an example from CentOS:

```
[root@TestLinux ~]# who



root     tty1         2013-12-09 10:44
root      pts/0        2013-12-09 10:29 (10.0.2.2)
sysadmin pts/1        2013-12-09 10:31 (10.0.2.2)
joeblog  pts/2        2013-12-09 10:39 (10.0.2.2)
```

In this particular case, I am the sole user of the system. I was running the server from an Oracle VirtualBox and accessing it as root from both the console and an SSH session. Two other user accounts (sysadmin and joebolg) were also accessing the system.

The last command tells us the login history of users:

```
[root@TestLinux ~]# last | grep sysadmin
```

ROLL TO TOP

```
sysadmin pts/0        10.0.2.2         Thu Nov 28 17:06 - 17:13  (00:06)
sysadmin pts/0        10.0.2.2         Thu Nov 28 16:17 - 17:05  (00:48)
sysadmin pts/0        10.0.2.2         Thu Nov 28 09:29 - crash  (06:04)
sysadmin pts/0        10.0.2.2         Wed Nov 27 16:37 - down   (00:29)
sysadmin tty1                          Wed Nov 27 14:05 - down   (00:36)
sysadmin tty1                          Wed Nov 27 13:49 - 14:04  (00:15)
```

In this example, I am trying to find the login history of the user sysadmin. As you can see, there were couple of instances where he managed to crash the system.

To find out when was the system last rebooted, we can run the following command:

```
[root@TestLinux ~]# last reboot
```

The result may look like this

```
reboot   system boot  2.6.32-358.el6.x Mon Dec  9 10:27 - 10:47  (00:19)
reboot   system boot  2.6.32-358.el6.x Fri Dec  6 16:37 - 10:47 (2+18:10)
reboot   system boot  2.6.32-358.el6.x Fri Dec  6 16:28 - 16:36  (00:08)    reboot   syst
reboot   system boot  2.6.32-358.el6.x Mon Dec  2 17:00 - 16:36 (3+23:36)
reboot   system boot  2.6.32-358.el6.x Fri Nov 29 16:01 - 16:36 (7+00:34)
reboot   system boot  2.6.32-358.el6.x Fri Nov 29 15:43 - 16:36 (7+00:53)
...
...
wtmp begins Fri Nov 15 16:11:54 2013
```

To see when did someone last log in to the system, use lastlog:

```
[root@TestLinux ~]# lastlog
```

In my system, the output looked like this:

```
Username         Port        From             Latest
root             tty1                          Mon Dec  9 10:44:30 +1100 2013
bin                                           **Never logged in**
daemon                                        **Never logged in**
```

ROLL TO TOP

```
sync                                       **Never logged in**
shutdown                                   **Never logged in**
halt                                       **Never logged in**
mail                                       **Never logged in**
uucp                                       **Never logged in**
operator                                   **Never logged in**
games                                      **Never logged in**
gopher                                     **Never logged in**
ftp                                        **Never logged in**
nobody                                     **Never logged in**
vcsa                                       **Never logged in**
saslauth                                   **Never logged in**
postfix                                    **Never logged in**
sshd                                       **Never logged in**
sysadmin         pts/1    10.0.2.2         Mon Dec  9 10:31:50 +1100 2013
dbus                                       **Never logged in**
joeblog          pts/2    10.0.2.2         Mon Dec  9 10:39:24 +1100 2013
```

For other text-based log files, you can use cat, head or tail commands to read the contents.

In the example below, I am trying to look at the last ten lines of /var/log/messages file in a Debian box:

```
debian@debian:~$ sudo tail /var/log/messages
```

Output:

```
Dec 16 01:21:08 debian kernel: [    9.584074] Bluetooth: BNEP (Ethernet Emulation) ver 1
Dec 16 01:21:08 debian kernel: [    9.584074] Bluetooth: BNEP filters: protocol multicast
Dec 16 01:21:08 debian kernel: [    9.648220] Bridge firewalling registered
Dec 16 01:21:08 debian kernel: [    9.696728] Bluetooth: SCO (Voice Link) ver 0.6
Dec 16 01:21:08 debian kernel: [    9.696728] Bluetooth: SCO socket layer initialized
Dec 16 01:21:08 debian kernel: [    9.832215] lp: driver loaded but no devices found
Dec 16 01:21:08 debian kernel: [    9.868897] ppdev: user-space parallel port driver
Dec 16 01:21:11 debian kernel: [   12.748833] [drm] Initialized drm 1.1.0 20060810
Dec 16 01:21:11 debian kernel: [   12.754412] pci 0000:00:02.0: PCI INT A -> Link[LNKB] -
Dec 16 01:21:11 debian kernel: [   12.754412] [drm] Initialized vboxvideo 1.0.0 20090303
```

ROLL TO TOP

At the heart of the logging mechanism is the rsyslog daemon. This service is responsible for listening to log messages from different parts of a Linux system and routing the message to an appropriate log file in the /var/log directory. It can also forward log messages to another Linux server.

The rsyslog Configuration File

The rsyslog daemon gets its configuration information from the `rsyslog.conf` file. The file is located under the /etc directory.

Basically, the rsyslog.conf file tells the rsyslog daemon where to save its log messages. This instruction comes from a series of two-part lines within the file.

This file can be found at `rsyslog.d/50-default.conf` on ubuntu.

The two part instruction is made up of a *selector* and an *action*. The two parts are separated by white space.

The selector part specifies what's the source and importance of the log message and the action part says what to do with the message.

The selector itself is again divided into two parts separated by a dot (.). The first part before the dot is called *acility (the origin of the message) and the second part after the dot is called priority (the severity of the message).

Together, the facility/priority and the action pair tell rsyslog what to do when a log message matching the criteria is generated.

Here is excerpt from a CentOS rsyslog.conf file:

```
# rsyslog v5 configuration file
...
...
# Include all config files in /etc/rsyslog.d/
IncludeConfig /etc/rsyslog.d/*.conf

#### RULES ####
# Log all kernel messages to the console.
# Logging much else clutters up the screen.
# Don't log private authentication messages!
*.info;mail.none;authpriv.none;cron.none                /var/log/messages

# The authpriv file has restricted access.
authpriv.*                                              /var/log/secure

# Log all the mail messages in one place.
mail.*                                                  -/var/log/maillog


# Log cron stuff
cron.*                                                  /var/log/cron

# Everybody gets emergency messages
*.emerg                                                 *

# Save news errors of level crit and higher in a special file.
uucp,news.crit                                          /var/log/spooler

# Save boot messages also to boot.log
local7.*                                                /var/log/boot.log
...
...
```

To understand what this all means, let's consider the different types of facilities recognized by Linux. Here is a list:

- **auth** or **authpriv**: Messages coming from authorization and security related events
- **kern**: Any message coming from the Linux kernel
- **mail**: Messages generated by the mail subsystem
- **cron**: Cron daemon related messages
- **daemon**: Messages coming from daemons
- **news**: Messages coming from network news subsystem
- **lpr**: Printing related log messages
- **user**: Log messages coming from user programs
- **local0 to local7**: Reserved for local use

And here is a list of priorities in ascending order:

- **debug**: Debug information from programs
- **info**: Simple informational message - no intervention is required
- **notice**: Condition that may require attention
- **warn**: Warning
- **err**: Error
- **crit**: Critical condition
- **alert**: Condition that needs immediate intervention
- **emerg**: Emergency condition

So now let's consider the following line from the file:

```
cron.*              /var/log/cron
```

This just tells the rsyslog daemon to save all messages coming from the cron daemon in a file called /var/log/cron. The asterix (*) after the dot (.) means messages of all priorities will be logged. Similarly, if the facility was specified as an asterix, it would mean all sources.

Facilities and priorities can be related in a number of ways.

In its default form, when there is only one priority specified after the dot, it means all events equal to or greater than that priority will be trapped. So the following directive causes any messages coming from the mail subsystem with a priority of warning or higher to be logged in a specific file under /var/log:

```
mail.warn           /var/log/mail.warn
```

This will log every message equal to or greater than the warn priority, but leave everything below it. So messages with err, crit, alert or emerg will also be recorded in this file.

Using an equal sign (=) after the dot (.) will cause only the specified priority to be logged. So if we wanted to trap only the info messages coming from the mail subsystem, the specification

ROLL TO TOP

```
mail.=info          /var/log/mail.info
```

Again, if we wanted to trap everything from mail subsystem except info messages, the specification would be something like the following

```
mail.!info          /var/log/mail.info
```

or

```
mail.!=info         /var/log/mail.info
```

In the first case, the mail.info file will contain everything with a priority lower than info. In the second case, the file will contain all messages with a priority above info.

Multiple facilities in the same line can be separated by commas.

Multiple sources (*facility.priority*) in the same line is separated by semicolon.

When an action is marked as an asterix (*), it means all users. This entry in my CentOS rsyslog.conf file is saying exactly that:

```
# Everybody gets emergency messages
*.emerg                                                 *
```

Try to see what's the rsyslog.conf is saying in your Linux system. Here is an excerpt from the Debian server I am running:

```
#  /etc/rsyslog.conf    Configuration file for rsyslog.
#
#           For more information see
#           /usr/share/doc/rsyslog-doc/html/rsyslog_conf.html
...
...
auth,authpriv.*         /var/log/auth.log
*.*;auth,authpriv.none      -/var/log/syslog
```

ROLL TO TOP

```
lpr.*               -/var/log/lpr.log
mail.*              -/var/log/mail.log
user.*              -/var/log/user.log

#
# Logging for the mail system.  Split it up so that
# it is easy to write scripts to parse these files.
#
mail.info           -/var/log/mail.info
mail.warn           -/var/log/mail.warn
mail.err            /var/log/mail.err
#
# Logging for INN news system.
#
news.crit           /var/log/news/news.crit
news.err            /var/log/news/news.err
news.notice         -/var/log/news/news.notice
```

As you can see, Debian saves all security/authorization level messages in `/var/log/auth.log` whereas CentOS saves it under `/var/log/secure`.

The configurations for rsyslog can come from other custom files as well. These custom configuration files are usually located in different directories under /etc/rsyslog.d. The rsyslog.conf file includes these directories using $IncludeConfig directive.

Here is what it looks like in Ubuntu:

```
#  Default logging rules can be found in /etc/rsyslog.d/50-default.conf
....
....
$IncludeConfig /etc/rsyslog.d/*.conf
```

The contents under the /etc/rsyslog.d directory looks like the following:

```
-rw-r--r-- 1 root root  311 Mar 17  2012 20-ufw.conf
-rw-r--r-- 1 root root  252 Apr 11  2012 21-cloudinit.conf
-rw-r--r-- 1 root root 1655 Mar 30  2012 50-default.conf
```

ROLL TO TOP

than one user needs to receive the message, their usernames are separated by commas. If the message needs to be broadcast to every user, it's specified by an asterix (*) in the action field.

Because of being part of a network operating system, rsyslog daemon can not only save log messages locally, it can also forward them to another Linux server in the network or act as a repository for other systems. The daemon listens for log messages in UDP port 514. The example below will forward kernel critical messages to a server called "texas".

```
kern.crit           @texas
```

Creating and Testing Your Own Log Messages

So now it's time for us to create our own log files.

To test this, we will do the following

- Add a log file specification in /etc/rsyslog.conf file
- Restart the rsyslog daemon
- Test the configuration using the logger utility

In the following example, I am adding two new lines in my CentOS Linux system's rsyslog.conf file. As you can see, each of them are coming from a facility called local4 and they have different priorities.

```
[root@TestLinux ~]# vi /etc/rsyslog.conf



....
....

# New lines added for testing log message generation

local4.crit                                             /var/log/local4crit.log
local4.=info                                            /var/log/local4info.log
```

ROLL TO TOP

```
[root@TestLinux ~]# /etc/init.d/rsyslog restart
Shutting down system logger:                               [  OK  ]
Starting system logger:                                    [  OK  ]
[root@TestLinux ~]#
```

To generate the log message now, the **logger** application is called:

```
[root@TestLinux ~]# logger -p local4.info " This is a info message from local 4"
```

Looking under the /var/log directory now shows two new files:

```
...
...
-rw-------  1 root root      0 Dec  9 11:21 local4crit.log
-rw-------  1 root root     72 Dec  9 11:22 local4info.log
```

The size of the local4info.log is non-zero. So when it's opened, I see the message has been recorded:

```
[root@TestLinux ~]# cat /var/log/local4info.log


Dec  9 11:22:32 TestLinux root:  This is a info message from local 4
```

Rotating Log Files

As more and more information is written to log files, they get bigger and bigger. This obviously poses a potential performance problem. Also, the management of the files become cumbersome.

Linux uses the concept of "rotating" log files instead of purging or deleting them. When a log is rotated, a new log file is created and the old log file is renamed and optionally compressed. A log file can thus have multiple old versions remaining online. These files will go back over a period of time and will represent the backlog. Once a certain number of backlogs have been generated, a new log rotation will cause the oldest log file to be deleted.

ROLL TO TOP

The logrotate Configuration File

Like rsyslog, logrotate also depends on a configuration file and the name of this file is logrotate.conf. It's located under /etc.

Here is what I see in the logrotate.conf file of my Debian server:

```
debian@debian:~$ cat /etc/logrotate.conf



# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

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

# uncomment this if you want your log files compressed
#compress

# packages drop log rotation information into this directory
include /etc/logrotate.d

# no packages own wtmp, or btmp -- we'll rotate them here
/var/log/wtmp {
    missingok
    monthly
    create 0664 root utmp
    rotate 1
}

/var/log/btmp {
    missingok
    monthly
    create 0660 root utmp
    rotate 1
}
```

ROLL TO TOP

The lines are fairly self-explanatory. By default, log files are to be rotated weekly with four backlogs remaining online at any one time. When the program runs, a new, empty log file will be generated and optionally the old ones will be compressed.

The only exception is for wtmp and btmp files. wtmp keeps track of system logins and btmp keeps track of bad login attempts. Both these log files are to be rotated every month and no error is returned if any previous wtmp or btmp file can be found.

Custom log rotation configurations are kept under etc/logrotate.d directory. These are also inluded in the logrotate.conf with the include directive. The Debian installation shows me the content of this directory:

```
debian@debian:~$ ls -l /etc/logrotate.d
```

```
total 44
-rw-r--r-- 1 root root 173 Apr 15  2011 apt
-rw-r--r-- 1 root root  79 Aug 12  2011 aptitude
-rw-r--r-- 1 root root 135 Feb 24  2010 consolekit
-rw-r--r-- 1 root root 248 Nov 28  2011 cups
-rw-r--r-- 1 root root 232 Sep 19  2012 dpkg
-rw-r--r-- 1 root root 146 May 12  2011 exim4-base
-rw-r--r-- 1 root root 126 May 12  2011 exim4-paniclog
-rw-r--r-- 1 root root 157 Nov 16  2010 pm-utils
-rw-r--r-- 1 root root  94 Aug  8  2010 ppp
-rw-r--r-- 1 root root 515 Nov 30  2010 rsyslog
-rw-r--r-- 1 root root 114 Nov 26  2008 unattended-upgrades
```

The contents of the rsyslog shows how to recycle a number of log files:

```
debian@debian:~$ cat /etc/logrotate.d/rsyslog
```

```
/var/log/syslog
{
    rotate 7
    daily
    missingok
    notifempty
```

ROLL TO TOP

```
    postrotate
        invoke-rc.d rsyslog reload > /dev/null
    endscript
}

/var/log/mail.info
/var/log/mail.warn
/var/log/mail.err
/var/log/mail.log
/var/log/daemon.log
/var/log/kern.log
/var/log/auth.log
/var/log/user.log
/var/log/lpr.log
/var/log/cron.log
/var/log/debug
/var/log/messages
{
    rotate 4
    weekly
    missingok
    notifempty
    compress
    delaycompress
    sharedscripts
    postrotate
        invoke-rc.d rsyslog reload > /dev/null
    endscript
}
```

As you can see, the syslog file will be reinitialized every day with seven days' worth of logs being kept online. Other log files are rotated every week.

Also worth noting is the postrotate directive. This specifies the action that happens after the whole log rotation has completed.

Testing the Rotation

Logrotate can be manually run to recycle one or more files. And to do that, we simply specify the relevant configuration file as an argument to the command.

ROLL TO TOP

```
[root@TestLinux ~]# ls -l /var/log


total 800
...
-rw-------  1 root root    359 Dec 17 18:25 maillog
-rw-------. 1 root root   1830 Dec 16 16:35 maillog-20131216
-rw-------  1 root root  30554 Dec 17 18:25 messages
-rw-------. 1 root root 180429 Dec 16 16:35 messages-20131216
-rw-------  1 root root    591 Dec 17 18:28 secure
-rw-------. 1 root root   4187 Dec 16 16:41 secure-20131216
...
...
```

The partial contents of the logrotate.conf file looks like this:

```
[root@TestLinux ~]# cat /etc/logrotate.conf



# see "man logrotate" for details
# rotate log files weekly
weekly

# keep 4 weeks worth of backlogs
rotate 4

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

Next we run the logrotate command:

```
[root@TestLinux ~]# logrotate -fv /etc/logrotate.conf
```

Messages scroll over as new files are generated, errors are encountered etc. When the dust settles, we try to check for new mail, secure or messages files:

ROLL TO TOP

```
-rw-------  1 root root    0 Dec 17 18:34 /var/log/maillog
-rw-------. 1 root root 1830 Dec 16 16:35 /var/log/maillog-20131216
-rw-------  1 root root  359 Dec 17 18:25 /var/log/maillog-20131217


[root@TestLinux ~]# ls -l /var/log/messages*
-rw-------  1 root root    148 Dec 17 18:34 /var/log/messages
-rw-------. 1 root root 180429 Dec 16 16:35 /var/log/messages-20131216
-rw-------  1 root root  30554 Dec 17 18:25 /var/log/messages-20131217


[root@TestLinux ~]# ls -l /var/log/secure*
-rw-------  1 root root    0 Dec 17 18:34 /var/log/secure
-rw-------. 1 root root 4187 Dec 16 16:41 /var/log/secure-20131216
-rw-------  1 root root  591 Dec 17 18:28 /var/log/secure-20131217
[root@TestLinux ~]#
```

As we can see, all three new log files have been created. The maillog and secure files are still empty, but the new messages file already has some data in it.

Last Words

Hopefully this tutorial has given you some ideas about Linux logging. You can try to look into your own development or test systems to have a better idea. Once you are familiar with the location of the log files and their configuration settings, use that knowledge for supporting your production systems. And then maybe you can create some aliases to point to these files to save some typing time as well.

Submitted by: Sadequl Hussain

By: Sadequl Hussain

♡ Upvote (19) Subscribe Share

ROLL TO TOP

Related Tutorials

How To Use Kibana Dashboards and Visualizations

How To Install and Use Logwatch Log Analyzer and Reporter on a VPS

How To Gather Infrastructure Metrics with Metricbeat on Ubuntu 18.04

How To Install Elasticsearch, Logstash, and Kibana (Elastic Stack) on CentOS 7

How To Set Up an Elasticsearch, Fluentd and Kibana (EFK) Logging Stack on Kubernetes

14 Comments

Leave a comment...

ROLL TO TOP

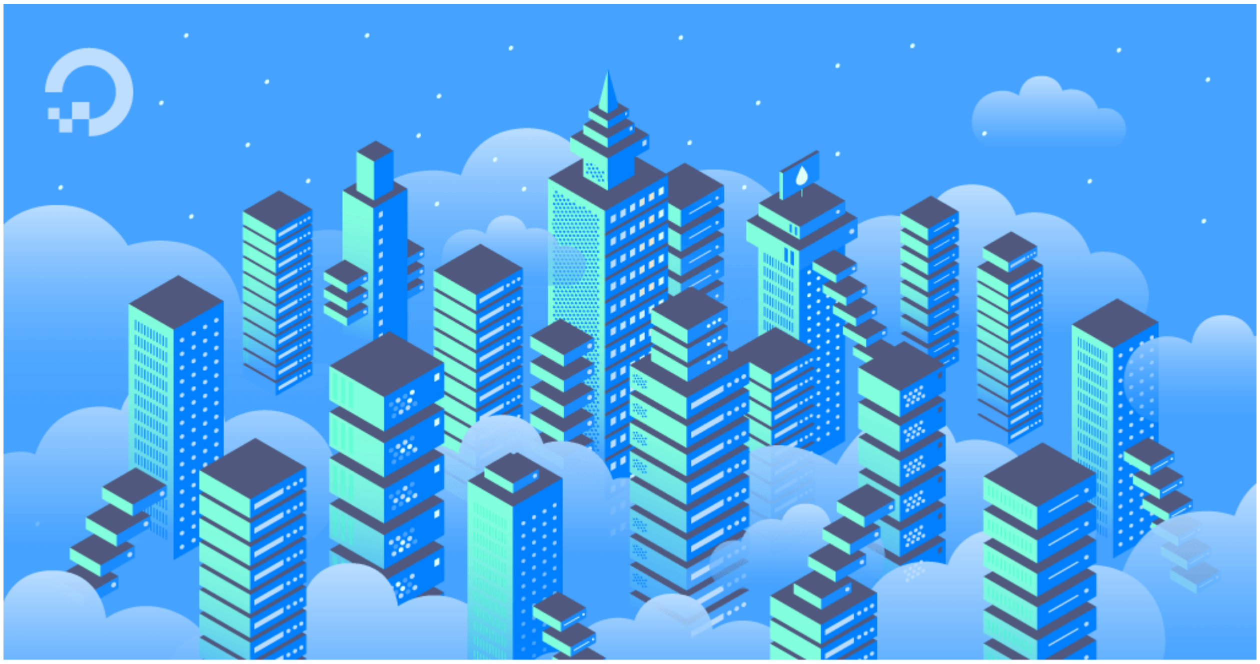
chris354008 *December 20, 2013*

0 You might also mention ways of having your logs delivered in a recap every day via email... like Logwatch. Thanks for this page though!

etel *December 20, 2013*

0 That's a great point. We also have an article on Logwatch here:

https://www.digitalocean.com/community/articles/how-to-install-and-use-logwatch-log-analyzer-and-reporter-on-a-vps



How To Install and Use Logwatch Log Analyzer and Reporter on a VPS

by O.S Tezer

Following in the footsteps of our previous articles on Linux system hardening, security monitoring and emailing alerts, in this DigitalOcean article we will talk about Logwatch: a very powerful log parser and analyzer which can make any dedicated system administrator's life a

chris354008 *December 20, 2013*

0 COOL! You guys rock so much!

ROLL TO TOP

Aww-- thanks :D

0

niranjan81 *December 21, 2013*

0 Superb, as always, Thank You very much !, please keep up the good work :-)

abhinavgupt04 *July 2, 2014*

0 I have been trying to look for basics of logging everywhere, this article made my day. Thank you so much. You are doing a great job :) :)

nightire *September 20, 2014*

0 Interesting, so many new thing to learn.

troy42 *December 2, 2014*

0 To Chris's question, I had that problem but wanted something which would work with more than one droplet (but only send one email) and would let me only get emailed about certain logs.

To do that, I built this: https://papertrailapp.com/

If it helps anyone else here, it's free for smaller volumes of logs and since it's a service, setting it up with rsyslog is literally copying and pasting one line.

balamurugansamy *February 2, 2015*

0 Its very helpful . I had configured a log server finally :) ..

Thanks..

melissa032215 *March 27, 2015*

0 Excellent advice! Thank you very much for your hard work.

melissa032215 *June 21, 2015*

ROLL TO TOP

myuvakishore32 *March 11, 2016*

0 Good Article. I really appreciate.

Some intelligent, Please guide me how to setup PHPMailer in Ubuntu.

I want to use PHPMailer to send mails from Webpage.

petec00c23c32c2 *September 13, 2016*

0 Great article as ever.
Where/how are iptable messages logged?

sayguno *November 27, 2016*

0 Hi, great informative post as always. With a SaaS log management solution logging with rsyslog is super easy. I use ZettaLogs (https://zettalogs.com). This one is a feature-rich and cost-effective. It has fast search, easy-to-use log parsing feature, real-time alerting, archiving, functional and clean UI, nice graphs and customizable dashboards.

ROLL TO TOP

Community Tutorials Questions Projects Tags Newsletter RSS

Distros & One-Click Apps Terms, Privacy, & Copyright Security Report a Bug Write for DOnations Shop

ROLL TO TOP