

CLOUDADMIN

System administration technologies brought to you from the coalface of Linux.



Jonathan Roberts dropped out of an MA in Theology to work with Linux. A Fedora advocate and systems administrator, we hear his calming tones whenever we're stuck with something hard.

I've said it before and I'll say it again: as a system administrator, all I want is a quiet life. I want to be able to walk out of work, on call, and know that I'm not going to be woken in the night. Unfortunately, that's just not going to happen. Accidents happen and hardware fails. It's a fact of life, and a perfectly quiet life is not realistic. If not a quiet life, then, we ought at least to aim for an anxiety free life.

Although it passed me by at the time, I now see that 31 March was World Backup Day – a friendly reminder that we should all be backing up our stuff. As a system administrator, just knowing that you're backing up all your important data, checking that the backups are actually running and that you can restore them in case of disaster, will go a long way to reducing your anxiety. In an age of configuration management, Amazon Web Services and the like, you should go one step further: you ought to know that you can quickly, accurately and automatically rebuild every single one of your systems.

Destroy to rebuild

Ideally, this means taking advantage of all that redundancy you've built in to your platform. Running a web farm behind a load balancer? When you release new software, don't just push it out to the existing servers; build new servers and destroy the old ones. Running a database cluster and doing an OS upgrade? Don't just do a yum update and reboot; rebuild the server, restore your data and destroy the old one.

Having done this a thousand times, knowing that you can rebuild your systems and restore critical data to them will cut out so much anxiety from your life as a system administrator.

Logs

Get started with your distro's built-in log management tools.

After three months looking at exciting new technologies, this issue I want to step back and look at something far less glamorous: logging. Often an afterthought to developers and system administrators alike, collecting, monitoring and using logs is one of the best things you can do for your infrastructure. Let's start with the basics – what are logs and why should you care?

Logs are how non-interactive components of your system communicate with you, the user. Each time an action takes place a note will get written to a file (the 'log') somewhere on your hard drive. Most logs contain one entry per line in the file. In the case of a web server, for example, an entry will probably be in the Common Log Format:

```
192.168.79.2 - - [27/Apr/2014:13:55:36 +0000] "GET /lv.html HTTP/1.1" 200 2326
```

Here, you see:

- The IP address of the user who visited the website.
- (Two empty fields not logged in this example, represented by dashes).
- The date time and timezone when the server finished dealing with the request.

- The request itself.
- The status code (the result of the request).
- How many bytes were returned to the clients.

It only takes a little bit of imagination to see how storing all this information can be useful. You can map IP addresses to geographic location, letting you know where in the world your site's visitors are coming from; you can see which web pages are most popular; you can even see if any pages experience more errors than others and at what times those errors occur.

Other logs on your system can help you keep things secure, too. For instance, on Red Hat-based systems, you can look in the `/var/log/secure` file and see a record of who logged in and when. You will also be able to see failed login attempts, commands executed with sudo and more besides. In the event of a security incident, your logs can be an invaluable source of information to help you understand how it happened, what damage was done, and how to stop it happening again in the future.

Now you know what logs are and why you should care about them, let's look at how you

The screenshot shows a browser window titled "rsyslog documentation - Mozilla Firefox". The address bar shows "file:///usr/share/doc/rsyslog-doc/manual.html". The page content includes the title "RSyslog - Documentation" and a detailed introduction to the software, its features, and how to get help. It also lists several links for troubleshooting and configuration.

RSyslog comes with a comprehensive set of documentation if you install the `rsyslog-doc` package. You'll find it installed in `/usr/share/doc/rsyslog-docs/`.

manage them. While not quite universal, you'll find many of your system's logs (on Red Hat based systems, at least) are managed by a program called RSyslog – the 'rocket-fast system for log processing'. It implements the syslog standard, enabling you to store logs locally or ship them remotely and store them in different formats.

RSyslog

In syslog-based systems, each message gets categorised with a facility and a priority. The facility describes which subsystem generated the messages, and can be one of auth, authpriv, cron, daemon, kern, lpr, mail, news, syslog, user, uucp, and local0 through local7. The local0–7 facilities are used for custom applications and reserved for the user to specify.

The priority describes whether the message is just informational or whether it relates to one of various error states. Available priorities are debug, info, notice, warning, err, crit, alert, and emerg.

The syslog daemon your computer is running will be configured to store messages of different facility and priority combinations in different locations. On a Red Hat-based system, you can see what the default configuration is like by looking in **/etc/rsyslog.conf**. For now, just skip to the **##### RULES #####** section.

Here, you can see lines defining where different messages go. To explain the syntax, look at this example:

```
*.info /var/log/messages
```

On the left is a filter to match log messages: in this example, messages of any facility with the priority **info** will be matched. On the right is the action – that is, what should happen to the message. In this example, the matched messages will be sent to the file **/var/log/messages** (note that most log messages will be stored in the **/var/log** directory by default – it's always a good place to start when looking for a log).

If you want to, you can use this file to redirect certain log messages to different files. Be sure to restart the **rsyslog** daemon after you make changes. Remember that many application might define their own logging arrangements, so you could find, for example, that your web server has its own log configuration in **/etc/httpd/conf/httpd.conf** or in individual virtual host definitions. Keep this in mind if you can't find the log file you're looking for.

If you want to interact with the syslog implementation from shell scripts you've

```
terminal - root@localhost:~
Apr 29 17:23:02 localhost systemd: Started dnf makecache.
Apr 29 17:28:37 localhost dnclient[1528]: DHCPREQUEST on wlp2s0 to 192.168.1.254 port 67 (xid=0x39e23547)
Apr 29 17:28:37 localhost dnclient[1528]: DHCPACK from 192.168.1.254 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> (wlp2s0): DHCPV4 state changed bound -> renew
Apr 29 17:28:37 localhost dnclient[1528]: bound to 192.168.1.83 -- renewal in 38623 seconds.
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> address 192.168.1.83
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> plen 24 (255.255.255.0)
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> gateway 192.168.1.254
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> server identifier 192.168.1.254
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> lease time 86400
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> nameserver 192.168.1.254
Apr 29 17:28:37 localhost NetworkManager[1029]: <info> domain name 'lan'
Apr 29 17:28:37 localhost dbus[917]: [system] Activating via systemd: service name='org.freedesktop.nm_dispatcher' unit='dbus-org.freedesktop.nm-dispatcher.service'
Apr 29 17:28:37 localhost dbus[917]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'
Apr 29 17:28:37 localhost NetworkManager: DHCPREQUEST on wlp2s0 to 192.168.1.254 port 67 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager: DHCPACK from 192.168.1.254 (xid=0x39e23547)
Apr 29 17:28:37 localhost NetworkManager: bound to 192.168.1.83 -- renewal in 38623 seconds.
Apr 29 17:28:37 localhost dbus-daemon: dbus[917]: [system] Activating via systemd: service name='org.freedesktop.nm_dispatcher' unit='dbus-org.freedesktop.nm-dispatcher.service'
Apr 29 17:28:37 localhost dbus-daemon: dbus[917]: [system] Successfully activated service 'org.freedesktop.nm_dispatcher'
Apr 29 17:28:37 localhost systemd: Starting Network Manager Script Dispatcher Service...
Apr 29 17:28:37 localhost systemd: Started Network Manager Script Dispatcher Service.
Apr 29 17:40:19 localhost dbus-daemon: dbus[917]: [system] Activating via systemd: service name='net.reactivated.Fprint' unit='fprintd.service'
Apr 29 17:40:19 localhost dbus-daemon: dbus[917]: [system] Successfully activated service 'net.reactivated.Fprint'
Apr 29 17:40:19 localhost systemd: Starting Fingerprint Authentication Daemon...
Apr 29 17:40:19 localhost systemd: Started Fingerprint Authentication Daemon.
Apr 29 17:40:19 localhost fprintd: Launching FprintObject
```

/var/log/messages is overwhelming at first, but it contains a treasure trove of information. Next time you're trying to figure out why something doesn't work, make **/var/log** your first stop.

written (backup scripts etc), then you can use the **logger** command. Use is simple:

```
logger -p local2.info "Database backup started."
```

This will send your defined message to syslog with the facility **local2** and priority **info**. You can then configure where such messages end up by adding a new entry to the **/etc/rsyslog.conf** file.

What's great about using logger is that you don't have to worry about formatting your message: logger will automatically append date, host and user information to your messages.

Try playing with this to see if you can generate a message in **/var/log/messages** – on most systems, if you run logger without the **-p** option, you'll find your message in **/var/log/messages** due to a line similar to the following in **/etc/rsyslog.conf**:

```
*.info;mail.none;authpriv.none;cron.none /var/log/messages
```

Logrotate

As great as syslog is, if you ever find yourself managing a cluster of more than two servers, you'll quickly find yourself getting annoyed with logging in to many different servers just to get a picture of what's happening across your entire cluster.

Thankfully, RSyslog provides the means by which you can centralise your logs. For example:

```
*.* @192.168.79.10
```

Putting this line in your RSyslog configuration file will instruct RSyslog to forward messages of all facilities and priorities to the machine at 192.168.79.10. Pretty simple, and you can filter messages in exactly the same way as shown for local logging, too. Setting up the receiving server isn't much more difficult:

```
$ModLoad imudp
$UDPServerRun 514
...
$template web,"/var/log/remote/%fromhost%/%year%/%month%/%day%/error.log"
...
local7.error ?web
```

This snippet instructs RSyslog to load the **imudp** module, which lets it receive logs via the UDP protocol, and instructs it to listen on port 514. Below that, we define a template for the output file name. In this example, we've included a number of properties that will get expanded by RSyslog on receipt of messages to create the full path to the log file. Finally, we filter all **local7** facility error messages to go to this template (the **?** indicates that a template is being used in the action).

Remotely storing logs like this isn't just convenient, it can help improve security, too. In the event of a breach, you have a second set of logs kept elsewhere. If the attacker wants to cover their tracks, they now have to break into a further server if they're to get at all copies of the logs.

What next?

That was a brief introduction to logging in Linux and rsyslog in particular. There's much more you can do with logs, however, and next month's issue we'll introduce you to Logstash. This open source tool can be taught how to parse your logs, after which it will index them, make it easy for you to generate pretty graphs and extract useful information from all those text files – all without having to touch grep and with lightning speed.

Until then, run back to your desk and sort out your logging – you'll thank us! 📄