SYSADMIN

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.

When it comes to troubleshooting, there's no substitute for knowing how the components of your system actually work. That said, there are some common steps that you should take every time you sit down to solve a problem, and it doesn't hurt to have an occasional refresher. So let's do just that: First, make sure you understand the problem – be able to describe what should be happening, what is happening and why this is wrong.

If a user comes to you and says their VPN connection is broken, you should not start by looking at their VPN logs, but instead ask "can you show me what you were trying to do when you found the problem?". Nine times out of ten, they'll show you what happens as they try to connect to a remote box with SSH or through a web browser and you'll reformulate the problem – "why can't the user connect via SSH to devbox1.test?".

Once you understand the problem, you can begin to think about solving it. Learn which log files are important, how you can increase their verbosity and how to read them carefully. Too many times, I've failed to solve a problem only to ask a colleague for help, at which point they look in the same log file I just did and spot the problem straight away.

Now you can formulate some hypotheses about what the cause could be. Write these down and come up with a way to test each one. Finally, work through all your hypotheses, testing one at a time, and only changing one thing at a time, until you find the answer. If you get through all of them without finding the answer, then you need to go back and make sure you understood the problem correctly and come up with new hypotheses. You are a philosopher.

Logstash

Understand the troves of information gathered by your logs.

Last issue, we talked about why logs are important and offered a brief introduction to collecting logs with syslog. This month, we want to take you a few steps beyond those basics and introduce a log management solution.

There are plenty of options out there, including the very well regarded Splunk, but we'll be looking at one of the main open source options – Logstash. It accepts input (log entries and other kinds of events) from many sources, stores them in a searchable back-end, and then enables you to query all that information through a web interface.

In short, Logstash helps you to actually use your logs and gain valuable operational insights from them.

In this month's issue, we're going to show you how to set up Logstash in conjunction with Elasticsearch, a storage backend, and Kibana, a fancy web interface which will help you query your data without learning the ins-and-outs of Elasticsearch.

Getting started

The first job, as ever, is to install all the components. Elasticsearch, the company behind all three components we're looking at today, provides a repository for Logstash and Elasticsearch, details of which are on its website (**www.elasticsearch.org/blog/apt-and-yum-repositories**).

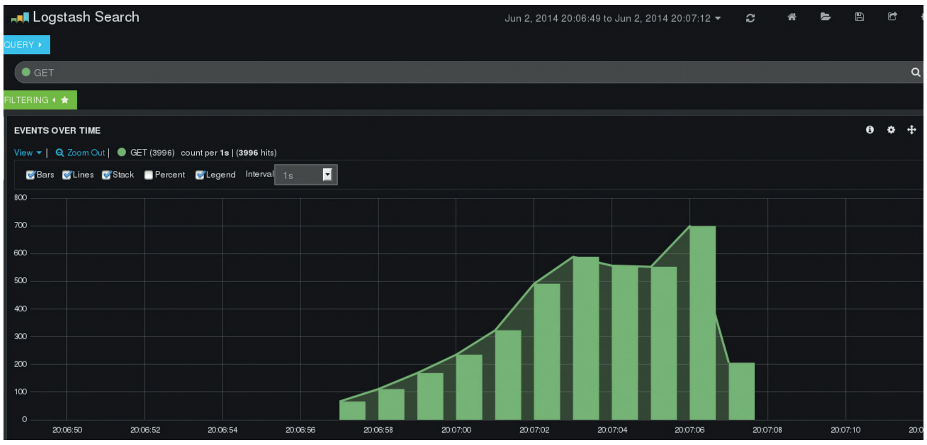
Once you've installed the package, start the Elasticsearch server with **service elasticsearch start** or equivalent.

Kibana has no package, but since its latest incarnation is just a collection of HTML and JavaScript files, installation isn't too complicated:

- Install a web server (httpd)
- Download Kibana (**https://download.elasticsearch.org/kibana/kibana/kibana-3.1.0.tar.gz**)
- Extract the files to your document root

We'll be running Kibana on the same server as Elasticsearch, so you shouldn't need to make any changes to its configuration.

With all that in place, you should be able to go to your Kibana installation in a web browser and see the Kibana welcome page. If you don't see a black page, Kibana isn't installed properly – check your httpd configuration – and if you see a red bar saying 'Couldn't connect to Elasticsearch' - check the service is running.



Kibana is an excellent web interface that makes it really easy to query your log data in Elasticsearch – even managers can understand it!

logstash - open source log management - Mozilla Firefox
Firefox
logstash - open source...
logstash - open source...
logstash/patterns/grok...
Kibana 3 - Logstash S...
File input for logstash ...
logstash filter - Google...
logstash.net
logstash filter
home
docs
about
cookbook
bugs
Download
Learn
Cookbook

logstash

logstash is a tool for managing events and logs. You can use it to collect logs, parse them, and store them for later use (like, for searching). Speaking of searching, logstash comes with a web interface for searching and drilling into all of your logs.

It is fully free and fully open source. The license is Apache 2.0, meaning you are pretty much free to use it however you want in whatever way.

logstash is now a part of the Elasticsearch family! This allows us to build better software much faster as well as offering production support

Logstash is just one component of the Elasticsearch, Logstash and Kibana stack – but it does have the coolest logo of them all, which is why we've included it in the screenshot.

Now we have a way to inspect our data without having to learn the ins-and-outs of the Elasticsearch API let's get some data in to Logstash and check everything is working as expected.

Getting your data

First, let's test Logstash in the simplest way possible:

```
cd /opt/logstash java -jar ./logstash.jar agent -e
'input { stdin { } } output { stdout { } }'
```

That command will start the Logstash process, accepting input from **stdin** and sending its output to **stdout**, rather than Elasticsearch. After running this, type anything you like into the console and you should see a structured reply from Logstash indicating that everything worked OK. (The **-e** argument simply lets you pass a configuration file on the command line – you can easily put this configuration in **/etc/logstash/conf.d**.)

If all works, let's stop using stdout for the output and instead use Elasticsearch:

```
java -jar ./logstash.jar agent -e 'input { stdin { } }
output { elasticsearch_http { host => localhost } }'
```

If you now type **hello linux voice** into the console and then refresh your Kibana window, you should see your message logged towards the bottom of the page. We're making great progress!

Logstash end to end

Let's look at the three main parts of Logstash configuration: inputs, filters and outputs. Inputs and outputs you've met already. These blocks simply define where input will come from and where you want to store it. In each section, you specify which plugins you want to deal with your input or output (such as **stdin**, **file**, **elasticsearch_http** etc) and pass any parameters to configure the plugin as you wish. Filters are more interesting. These get applied to the input, modifying them in some way, before they get stored by the output plugin. For example, with the Grok filter you can take any plain text file, such as an Apache or MySQL log file, and map regular expression matches to fields. This lets you split your output into separate, searchable fields, making it much easier for you to answer questions like 'do we get more 500 errors in summer than winter?'.

Grok comes with lots of common patterns already defined, so you rarely even have to write the regular expressions yourself. You can find the list of included patterns at **https://github.com/elasticsearch/logstash/blob/master/patterns/grok-patterns**.

More inputs

There are loads of input, filter and output plugins available for Logstash, but for a clearer example, let's look at a configuration file that would allow you to parse an Apache log file. First, the input:

```
input { file { path => '/var/log/httpd/access_log'
start_position => beginning } }
```

Here, we're specifying the default Apache access log as the input to Logstash. If you point this at the log file for your Kibana installation, you can use Logstash to track Kibana usage. By giving the **start_position** parameter, Logstash will look through the whole of the file, enabling you to analyse historical data, before operating more like the **tail -f** command, appending new input as it arrives.

Next, the filter:

```
filter { grok { match => { 'message' =>
'%{COMBINEDAPACHELOG}' } } }
```

This time, we're using the Grok filter plugin. The **match** parameter does the work of mapping a regular expression match to a field name – in our example, we're taking anything that matches the **COMBINEDAPACHELOG** pattern (the default format used in **httpd.conf**) and mapping it to the **message** field.

Finally, the output:

```
output { elasticsearch_http { host => localhost }
stdout { } }
```

While this may look familiar, notice that we've included two output plugins. Logstash will send the result of the filter to both standard output and Elasticsearch, making it a bit easier to debug.

Put those three sections into a configuration file and start Logstash with the **-f** option, passing it the location of the configuration file. Once Logstash has started, go to Kibana and refresh. As you browse around Kibana, you should find new log entries being recorded.

Kibana

Now you've actually got some logs being stored, you can use Kibana to start asking questions of that data.

By default, most of the Kibana interface is filled with a large graph. As your data makes its way in, this graph will fill up with bars representing activity at different points in time. You can click and drag on the graph to zoom in and get a higher resolution look at a particular moment in time.

You can also click the View button in the top-left of the graph to adjust what kind of chart is drawn, over what period the data is sampled at and much more besides.

Above the chart is the Query bar, where you can enter relatively free-form patterns to search for in the data. Enter an IP address, and it will filter all matching results. There's so much you can do with Kibana. We'd love to hear how you're using Logstash on your systems, so please drop us a line if you're using it! LV

Logstash beyond logs

As system administrators, we immediately think about how Logstash can be used to help us make use of the mountains of syslog data we're all storing, but it can be just as useful for developers who are trying to build more intelligent businesses.

For instance, if you're running a large retail company, you could easily use the TCP input to keep track of product sales – every time a product is sold, have your point of sale terminals send a message to your Logstash server, and it will index the data, graph it and help your management teams spot and quickly respond to emerging trends.