GNU/Linux Administration - Support #424

Centralized Log Collection & Analysis With Logstash on Ubuntu

07/15/2014 07:55 AM - Daniel Curtis

Status:	Closed	**Start date:**	07/15/2014
Priority:	Normal	**Due date:**	
Assignee:	Daniel Curtis	**% Done:**	90%
Category:	Logging Server	**Estimated time:**	3.00 hours
Target version:		**Spent time:**	4.00 hours

Description

In this tutorial, we will go over the installation of Logstash 1.4.2 and Kibana 3, and how to configure them to gather and visualize the syslogs of our systems in a centralized location. Logstash is an open source tool for collecting, parsing, and storing logs for future use. Kibana 3 is a web interface that can be used to search and view the logs that Logstash has indexed. Both of these tools are based on Elasticsearch. Elasticsearch, Logstash, and Kibana, when used together is known as an ELK stack.

Centralized logging can be very useful when attempting to identify problems with your servers or applications, as it allows you to search through all of your logs in a single place. It is also useful because it allows you to identify issues that span multiple servers by correlating their logs during a specific time frame.

It is possible to use Logstash to gather logs of all types, but we will limit the scope of this tutorial to syslog gathering.

The Goal

The goal of the tutorial is to set up Logstash to gather syslogs of multiple servers, and set up Kibana to visualize the gathered logs.

Our Logstash / Kibana setup has four main components:

1. **Logstash**: The server component of Logstash that processes incoming logs
2. **Elasticsearch**: Stores all of the logs
3. **Kibana**: Web interface for searching and visualizing logs
4. **Logstash Forwarder**: Installed on servers that will send their logs to Logstash, Logstash Forwarder serves as a log forwarding agent that utilizes the lumberjack networking protocol to communicate with Logstash

We will install the first three components on a single server, which we will refer to as our Logstash Server. The Logstash Forwarder will be installed on all of the servers that we want to gather logs for, which we will refer to collectively as our Servers.

Prerequisites

To complete this tutorial, you will require root access to an Ubuntu 14.04 VPS. Instructions to set that up can be found here (steps 3 and 4): Initial Server Setup with Ubuntu 14.04.

The amount of CPU, RAM, and storage that your Logstash Server will require depends on the volume of logs that you intend to gather. For this tutorial, we will be using a VPS with the following specs for our Logstash Server:

- OS: Ubuntu 14.04
- RAM: 1GB
- CPU: 2

In addition to your Logstash Server, you will want to have a few other servers that you will gather logs from.

Let's get started on setting up our Logstash Server!

Install Java 7

Elasticsearch and Logstash require Java 7, so we will install that now. We will install Oracle Java 7 because that is what Elasticsearch recommends. It should, however, work fine with OpenJDK, if you decide to go that route.

- Add the Oracle Java PPA to apt:

```
sudo add-apt-repository ppa:webupd8team/java
```

1. **NOTE**: If add-apt-repository is not available, then install the python-software-properties and software-properties-common packages. This may be necessary for minimal Ubuntu installations:

   ```
   sudo apt-get install software-properties-common python-software-properties
   ```

- Update your apt package database:

  ```
  sudo apt-get update
  ```

- Install the latest stable version of Oracle Java 7 with this command (and accept the license agreement that pops up):

  ```
  sudo apt-get install oracle-java7-installer
  ```

Now that Java 7 is installed, let's install ElasticSearch.

Install Elasticsearch

NOTE: Logstash 1.4.2 recommends Elasticsearch 1.1.1.

- Import the Elasticsearch public GPG key into apt:

  ```
  wget -O - http://packages.elasticsearch.org/GPG-KEY-elasticsearch | sudo apt-key add -
  ```

- Create the Elasticsearch source list:

  ```
  echo 'deb http://packages.elasticsearch.org/elasticsearch/1.1/debian stable main' | sudo tee /
  etc/apt/sources.list.d/elasticsearch.list
  ```

- Update your apt package database:

  ```
  sudo apt-get update
  ```

- Install Elasticsearch 1.1.1 with this command:

  ```
  sudo apt-get install elasticsearch=1.1.1
  ```

- Elasticsearch is now installed. Let's edit the configuration:

  ```
  sudo vi /etc/elasticsearch/elasticsearch.yml
  ```

  1. Add the following line somewhere in the file, to disable dynamic scripts:

  script.disable_dynamic: true

1. You will also want to restrict outside access to your Elasticsearch instance (port 9200), so outsiders can't read your data or shutdown your Elasticseach cluster through the HTTP API. Find the line that specifies network.host and uncomment it so it looks like this:

   network.host: localhost

Save and exit elasticsearch.yml.

- Now start Elasticsearch:

```
sudo service elasticsearch restart
```

- Then run the following command to enable Elasticsearch on boot up:

```
sudo update-rc.d elasticsearch defaults 95 10
```

Now that Elasticsearch is up and running, let's install Kibana.

Install Kibana

NOTE: Logstash 1.4.2 recommends Kibana 3.0.1

- Download Kibana to your home directory with the following command:

```
cd ~; wget https://download.elasticsearch.org/kibana/kibana/kibana-3.0.1.tar.gz
```

- Extract Kibana archive with tar:

```
tar xvf kibana-3.0.1.tar.gz
```

- Open the Kibana configuration file for editing:
  sudo vi ~/kibana-3.0.1/config.js
  1. In the Kibana configuration file, find the line that specifies the elasticsearch, and replace the port number (9200 by default) with 80:

  elasticsearch: "http://"+window.location.hostname+":80",

This is necessary because we are planning on accessing Kibana on port 80 (i.e. http://logstash_server_public_ip/).

- We will be using Nginx to serve our Kibana installation, so let's move the files into an appropriate location. Create a directory with the following command:

```
sudo mkdir -p /var/www/kibana3
```

- Now copy the Kibana files into your newly-created directory:

```
sudo cp -R ~/kibana-3.0.1/* /var/www/kibana3/
```

Before we can use the Kibana web interface, we have to install Nginx. Let's do that now.

Install Nginx

- Use apt to install Nginx:

```
sudo apt-get install nginx
```

Because of the way that Kibana interfaces the user with Elasticsearch (the user needs to be able to access Elasticsearch directly), we need to configure Nginx to proxy the port 80 requests to port 9200 (the port that Elasticsearch listens to by default). Luckily, Kibana provides a sample Nginx configuration that sets most of this up.

- Download the sample Nginx configuration from Kibana's github repository to your home directory:

```
cd ~; wget https://github.com/elasticsearch/kibana/raw/master/sample/nginx.conf
```

- Open the sample configuration file for editing:

```
vi nginx.conf
```

1. Find and change the values of the server_name to your FQDN (or localhost if you aren't using a domain name) and root to where we installed Kibana, so they look like the following entries:

server_name FQDN;
root /var/www/kibana3;

Save and exit.

- Now copy it over your Nginx default server block with the following command:

```
sudo cp nginx.conf /etc/nginx/sites-available/default
```

- Now we will install apache2-utils so we can use htpasswd to generate a username and password pair:

```
sudo apt-get install apache2-utils
```

- Then generate a login that will be used in Kibana to save and share dashboards (substitute your own username):

```
sudo htpasswd -c /etc/nginx/conf.d/kibana.myhost.org.htpasswd user
```

Then enter a password and verify it. The htpasswd file just created is referenced in the Nginx configuration that you recently configured.

- Now restart Nginx to put our changes into effect:

```
sudo service nginx restart
```

Kibana is now accessible via your FQDN or the public IP address of your Logstash Server i.e. http://logstash_server_public_ip/. If you go there in a web browser, you should see a Kibana welcome page which will allow you to view dashboards but there will be no logs to view because Logstash has not been set up yet. Let's do that now.

Install Logstash

- The Logstash package is available from the same repository as Elasticsearch, and we already installed that public key, so let's create the Logstash source list:

```
echo 'deb http://packages.elasticsearch.org/logstash/1.4/debian stable main' | sudo tee /etc/a
pt/sources.list.d/logstash.list
```

- Update your apt package database:

```
sudo apt-get update
```

- Install Logstash 1.4.2 with this command:

```
sudo apt-get install logstash=1.4.2-1-2c0f5a1
```

Logstash is installed but it is not configured yet.

Generate SSL Certificates

- Since we are going to use Logstash Forwarder to ship logs from our Servers to our Logstash Server, we need to create an SSL certificate and key pair. The certificate is used by the Logstash Forwarder to verify the identity of Logstash Server. Create the directories that will store the certificate and private key with the following commands:

```
sudo mkdir -p /etc/pki/tls/certs
sudo mkdir /etc/pki/tls/private
```

- Now generate the SSL certificate and private key, in the appropriate locations (/etc/pki/tls/...), with the following command:

```
cd /etc/pki/tls; sudo openssl req -x509 -batch -nodes -newkey rsa:2048 -keyout private/logstas
h-forwarder.key -out certs/logstash-forwarder.crt
```

The **logstash-forwarder.crt** file will be copied to all of the servers that will send logs to Logstash but we will do that a little later. Let's complete our Logstash configuration.

Configure Logstash

Logstash configuration files are in the JSON-format, and reside in /etc/logstash/conf.d. The configuration consists of three sections: inputs, filters, and outputs.

- Let's create a configuration file called 01-lumberjack-input.conf and set up our "lumberjack" input (the protocol that Logstash Forwarder uses):

```
sudo vi /etc/logstash/conf.d/01-lumberjack-input.conf
```

1. Insert the following input configuration:

```
input {
  lumberjack {
    port => 5000
    type => "logs"
    ssl_certificate => "/etc/pki/tls/certs/logstash-forwarder.crt"
    ssl_key => "/etc/pki/tls/private/logstash-forwarder.key"
  }
}
```

Save and quit. This specifies a lumberjack input that will listen on tcp port 5000, and it will use the SSL certificate and private key that we created earlier.

- Now let's create a configuration file called 10-syslog.conf, where we will add a filter for syslog messages:

```
sudo vi /etc/logstash/conf.d/10-syslog.conf
```

1. Insert the following syslog filter configuration:

```
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp} %{SYSLOGHOST:syslog_hos
tname} %{DATA:syslog_program}(?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}"
}
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
```

Save and quit. This filter looks for logs that are labeled as "syslog" type (by a Logstash Forwarder), and it will try to use "grok" to parse incoming syslog logs to make it structured and query-able.

- Lastly, we will create a configuration file called 30-lumberjack-output.conf:

```
sudo vi /etc/logstash/conf.d/30-lumberjack-output.conf
```

1. Insert the following output configuration:

```
output {
  elasticsearch { host => localhost }
  stdout { codec => rubydebug }
}
```

Save and exit. This output basically configures Logstash to store the logs in Elasticsearch.

With this configuration, Logstash will also accept logs that do not match the filter, but the data will not be structured (e.g. unfiltered Nginx or Apache logs would appear as flat messages instead of categorizing messages by HTTP response codes, source IP addresses, served files, etc.).

If you want to add filters for other applications that use the Logstash Forwarder input, be sure to name the files so they sort between the input and the output configuration (i.e. between 01 and 30).

- Restart Logstash to put our configuration changes into effect:

```
sudo service logstash restart
```

Now that our Logstash Server is ready, let's move onto setting up Logstash Forwarder.

Set Up Logstash Forwarder

Note: Do these steps for each server that you want to send logs to your Logstash Server.

Copy SSL Certificate and Logstash Forwarder Package

- On Logstash Server, copy the SSL certificate to Server (substitute with your own login):

```
scp /etc/pki/tls/certs/logstash-forwarder.crt user@server_private_IP:/tmp
```

Install Logstash Forwarder Package

- On Server, download the Logstash Forwarder Debian package to your home directory:

```
cd ~; wget http://packages.elasticsearch.org/logstashforwarder/debian/pool/main/l/logstashforw
arder/logstash-forwarder_0.3.1_amd64.deb
```

- Then install the Logstash Forwarder package:

```
sudo dpkg -i ~/logstash-forwarder_0.3.1_amd64.deb
```

- Next, you will want to install the Logstash Forwarder init script, so it starts on bootup:

```
cd /etc/init.d/; sudo wget https://raw.github.com/elasticsearch/logstash-forwarder/master/logs
tash-forwarder.init -O logstash-forwarder
sudo chmod +x logstash-forwarder
sudo update-rc.d logstash-forwarder defaults
```

- Now copy the SSL certificate into the appropriate location (/etc/pki/tls/certs):

```
sudo mkdir -p /etc/pki/tls/certs
sudo cp /tmp/logstash-forwarder.crt /etc/pki/tls/certs/
```

Configure Logstash Forwarder

- On Server, open the Logstash Forwarder configuration file, which is formatted as JSON, for editing:

```
sudo vi /etc/logstash-forwarder
```

1. Now add the following lines into the file, substituting in your Logstash Server's private IP address for logstash_server_private_IP:

```
{
  "network": {
    "servers": [ "logstash_server_private_IP:5000" ],
    "timeout": 15,
    "ssl ca": "/etc/pki/tls/certs/logstash-forwarder.crt"
  },
  "files": [
    {
      "paths": [
        "/var/log/syslog",
        "/var/log/auth.log"
       ],
      "fields": { "type": "syslog" }
    }
   ]
}
```

Save and quit. This configures Logstash Forwarder to connect to your Logstash Server on port 5000 (the port that we specified an input for earlier), and uses the SSL certificate that we created earlier. The paths section specifies which log files to send (here we specify syslog and auth.log), and the type section specifies that these logs are of type "syslog* (which is the type that our filter is looking for).

Note that this is where you would add more files/types to configure Logstash Forwarder to other log files to Logstash on port 5000.

- Now restart Logstash Forwarder to put our changes into place:

```
sudo service logstash-forwarder restart
```

Now Logstash Forwarder is sending syslog and auth.log to your Logstash Server! Repeat this process for all of the other servers that you wish to gather logs for.

Connect to Kibana

When you are finished setting up Logstash Forwarder on all of the servers that you want to gather logs for, let's look at Kibana, the web interface that we installed earlier.

In a web browser, go to the FQDN or public IP address of your Logstash Server. You should see a Kibana welcome page.

Click on Logstash Dashboard to go to the premade dashboard. You should see a histogram with log events, with log messages below (if you don't see any events or messages, one of your four Logstash components is not configured properly).

Here, you can search and browse through your logs. You can also customize your dashboard. This is a sample of what your Kibana instance might look like:

Kibana 3 Example Dashboard

Try the following things:

1. Search for "root" to see if anyone is trying to log into your servers as root
2. Search for a particular hostname
3. Change the time frame by selecting an area on the histogram or from the menu above
4. Click on messages below the histogram to see how the data is being filtered

Kibana has many other features, such as graphing and filtering, so feel free to poke around!
Conclusion

Now that your syslogs are centralized via Logstash, and you are able to visualize them with Kibana, you should be off to a good start with centralizing all of your important logs. Remember that you can send pretty much any type of log to Logstash, but the data becomes even more useful if it is parsed and structured with grok.

Note that your Kibana dashboard is accessible to anyone who can access your server, so you will want to secure it with something like htaccess.

Resources

https://www.digitalocean.com/community/tutorials/how-to-use-logstash-and-kibana-to-centralize-and-visualize-logs-on-ubuntu-14-04

History

#1 - 07/16/2014 12:46 PM - Daniel Curtis

- Status changed from In Progress to Closed

- % Done changed from 20 to 90

#2 - 02/15/2015 08:40 PM - Daniel Curtis

- Project changed from 94 to GNU/Linux Administration

- Category set to Logging Server