Nagios Log Server – Configuring Input Filters

The Industry Standard in IT Infrastructure Monitoring

Purpose

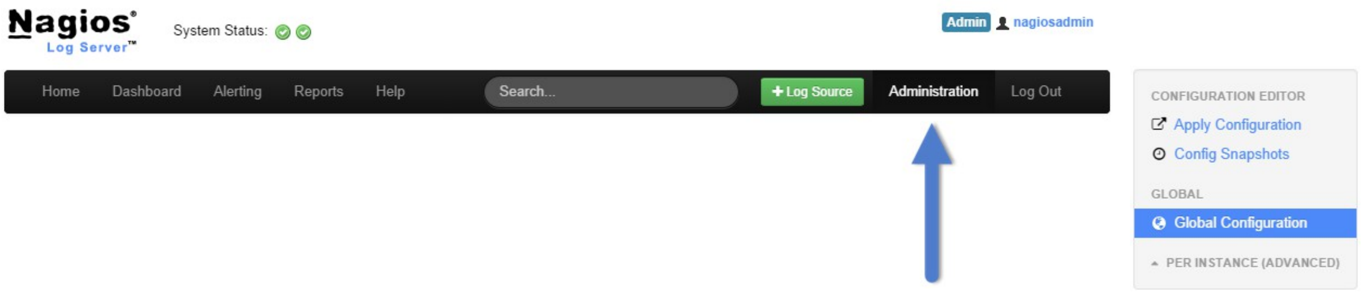
This document describes how to customize Nagios Log Server Input Filters.

Target Audience

This document is intended for use by Nagios Log Server Administrators and Users who want to customize their Nagios Log Server Input Filters.

Navigate

First, Select the Administration section in Nagios Log Server:



Then Select the **'Global Configuration'** Global Configuration option from the first navigation window on the left side of the page.

Input Filters

On the Global configuration page there are two main tables named **Inputs** and **Filters** and for this document we will be configuring filters that are used by Logstash for logs that are incoming into your Nagios Log Server. To be able to verify you configured a filter correctly you will need to have logs being sent to Nagios Log Server to test the filter.

When log stash configurations are saved and applied the input, filter and output are all in the same configuration file. The structure of the filters are in JSON format and look like this:

```
filter {
        <filter_type> {
                <filter_action> => [ "<selected_field>", "<selected_pattern>" ]
        }
}
```

`<filter_type>` is the type of filter **plugin** that will be used to match the field and pattern you are looking for. Logstash allows a large amount of possible filter types, but we will explore some that are more useful for manipulating logs. For our example we will be using **grok** to do our filtering. To use any of the possible filter plugins you can import them from the Logstash website. The **filters** section column on the Logstash docs page would be where plugins like **date, grep, grok** and **elasticsearch** are located. **Mutate** will also be useful for changing any field in the logs we receive so you can customize logs as much as you want.

`<filter_action>` is the action that will be taken using the filter type. The most common using our **grok** plugin are **match, add_field, add_tag.** Here is an explanation and more details about **grok** from Logstash: http://Logstash.net/docs/1.4.2/filters/grok.

Grok Filter

Grok is a plugin that is used by Logstash for making specific filters using regex and matching. The plugin allows a lot of customization and will be used heavily in making custom filters in your Nagios Log Server configuration.

`<matching_field>` will be the log field that we are attempting to match. You can choose things from 'hostname' to 'message' and customize whatever is needed.

`<matching_pattern>` relates to the pattern that will be looked for when the field is given. For example the field 'message' will be populated with whatever log message was included. For PHP a log message may look like the following example:

```
[Thu Oct 02 16:05:21 2014] [error] [client 127.0.0.1] PHP Notice:  Undefined variable: _SESSION
in /path/to/file on line 202, referer: http://<address>/page/request/
```

This message from the apache error_log is showing a PHP Notice that there is an undefined variable. It shows the time the notice was posted, the type which is [error] here, the client sending the log posting, the PHP log type and message, the variable and path to the offending file, the line number the offending code is on and the referring page that generated the request that resulted in the notice.

This is a lot of information to take advantage of and this specific PHP notice isn't even as large as many of them could potentially be. This is what will be present in the **message** field. To take control of this message and filter the whole message or parts of it you will use a grok pattern that relates to the message.

Here is a **grok** filter example searching for any logs with the **mysqld** program name. It will match the message with the following matching in the message field:

```
filter {
       if [program] == "mysqld" {
              grok {
                     match => [ "message", "^%{NUMBER:date} *%{NOTSPACE:time}" ]
              }
              mutate {
                     replace => [ "type", "mysql_log" ]
              }
       }
}
```

This filter is designed to match mysqld message contents and replace the log **type** with **mysql_log.**

Mutate, as mentioned before, is another very useful plugin used in Logstash filtering. This allows you to replace or append a value in any field with a new value that may replace the whole field or add, delete or append portions of it. In our example above we are changing the log type which would be 'syslog' and replace it with 'mysql_log' so that we can differentiate between our normal syslogs and our mysql syslogs. It knows which syslogs to mutate because the [program] must be mysqld and must match the grok filter message pattern.

Filtering Time Stamps

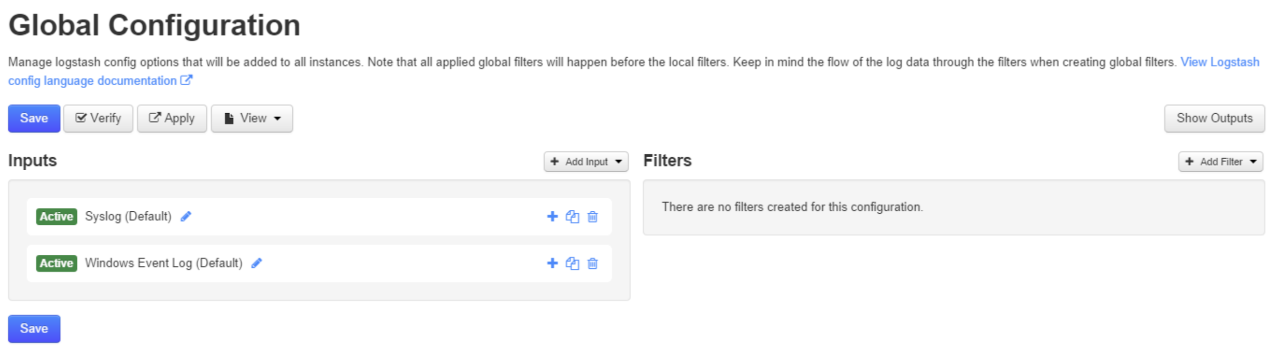
Filtering time stamps will be a very common use case when using grok or any other type of filter since time stamps have so many different formats and you may want to make all your logs share a single format. Here is an example for a time stamp that will be matched and the proper date format set so Logstash can parse the date:

```
date  {
       match => [ "timestamp", "MMM dd HH:mm:ss", "MMM d HH:mm:ss" ]
       add_tag => [ "dated" ]
}
```

This will be added below the grok filter section where the patterns are matched. The add tag “dated” is designed to tell the user a log has had its date formatted.

Adding Filters

Now that we know how filters are formatted and how they can be used in manipulating log data we want to add them to Nagios Log Server. Here is a view of the global configuration page when no filters exist:



Global Configuration

Manage logstash config options that will be added to all instances. Note that all applied global filters will happen before the local filters. Keep in mind the flow of the log data through the filters when creating global filters. View Logstash config language documentation

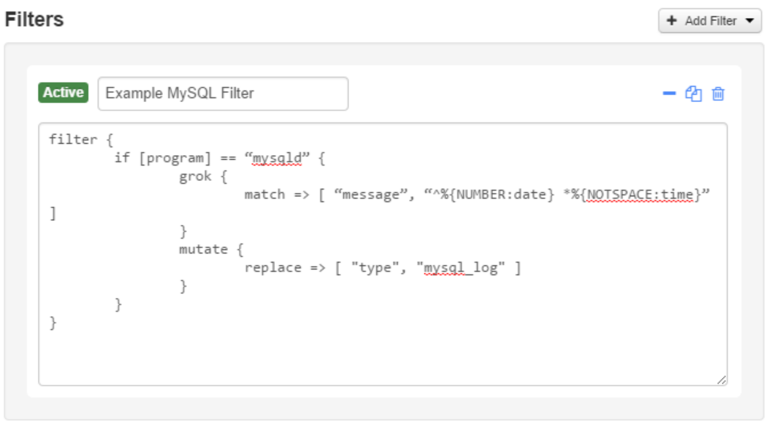
There are no filters created for this configuration.

Add

On the left is the default inputs that are present when Nagios Log Server is installed. On the right we see the **Filters** table with nothing added yet so we are going to add our MySQL example. To add a filter,

Click the **'Add Filter'** Add Filter button and select **'Custom'** Custom from the drop down.

The table will pop out and display a form (right):



Example MySQL Filter

```
filter {
        if [program] == “mysqld” {
                grok {
                        match => [ “message”, “^%{NUMBER:date} *%{NOTSPACE:time}”
]
                }
                mutate {
                        replace => [ "type", "mysql_log" ]
                }
        }
}
```

The **active** Active button will toggle if the filter is active or inactive so you can save filters even if you don't want to use them right away.

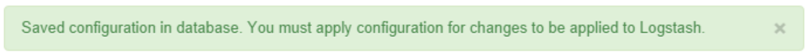
Then set the unique **name** of the filter. Here we named it 'Example MySQL Filter.'

Now you will add the actual filter that will be included in the Logstash configuration file. We will use our mysql filter we went over before and see how it works.

Save

Now that we have our nice MySQL filter in place we have to take a couple steps to save, verify and write the Logstash configuration file.

First we need to save the filter by clicking the **'Save'** Save button to save the filter we added. If it was successfully saved you will see this message:

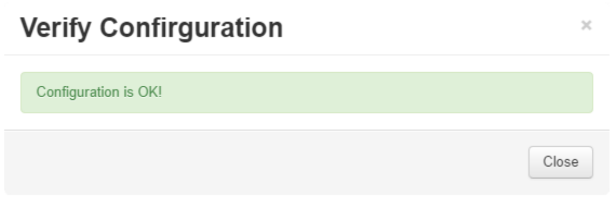


Saved configuration in database. You must apply configuration for changes to be applied to Logstash.

The message then tells us to apply the configuration, but first we will **verify** that the configuration syntax we saved will work.

Verify

Now select the **'Verify'** Verify button to check our newly added filter. If the filter syntax was verified successfully you will see a message in a window like this:


Verify Confirguration

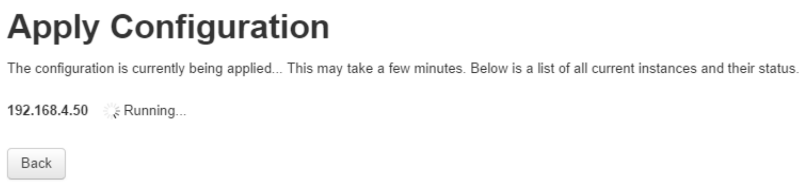
Configuration is OK!

Now that we know that the syntax of our input, filter and output will be accepted by Logstash we can finally apply the configuration so that it will be written.

Apply

Click the **'Apply'** Apply button and it will bring you to the Apply Configuration Apply Configuration view which is the first option of the configuration editor on the Administration page. This button will apply the saved configuration to ALL instances.

While the configuration is being applied you will see this screen:


Apply Configuration

The configuration is currently being applied... This may take a few minutes. Below is a list of all current instances and their status.

192.168.4.50 Running...

It shows which server the configuration is being applied to and will show a loading icon. Once the configuration has been applied successfully you will see the following next to each instance that was configured: 192.168.4.50 ✓ Completed!

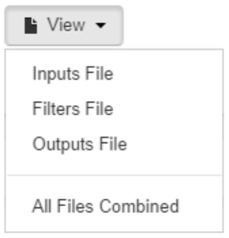
This will be your indication that the configuration was applied with no errors and should now be working on your instance if it is active. Lastly it will restart all the Logstash service on all instances.

View

To view the current configuration Inputs and Filters navigate back to the **'Global Configuration'** Global Configuration page.

Select the **'View'** View button which will show you each part of the overall configuration file.

To see all the files together and get an overview select All Files Combined



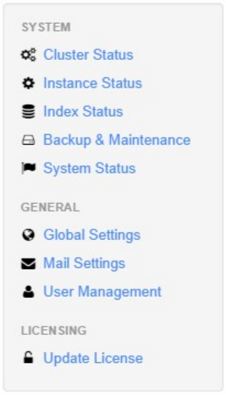
This will open a new modal window where the combined file can be viewed or copied.

Here is our current combined configuration file after adding our example MySQL filter:

```
#
# Logstash Configuration File
# Dynamically created by Nagios Log Server
#
# DO NOT EDIT THIS FILE. IT WILL BE OVERWRITTEN.
#
# Created Fri, 03 Oct 2014 11:40:51 -0500
#

#
# Global Configuration
#

input {
    syslog {
        type => 'syslog'
        port => 5544
    }
    tcp {
        type => 'eventlog'
        port => 3515
        codec => json {
            charset => 'CP1252'
        }
    }
}

filter {
    if [program] == "mysqld" {
        grok {
            match => [ "message", "^%{NUMBER:date} *%{NOTSPACE:time}"]
        }
        mutate {
            replace => [ "type", "mysql_log" ]
        }
    }
}

#
# Local Configuration
#
```

Finishing Up

There are more sections that make up the configuration and administration menu and you can look at the documentation and master your Nagios Log Server. Learning each part will allow administrators full control of all the features that are in Nagios Log Server.



If you have questions about Nagios Log Server or of its capabilities, contact our support team via our online form at:

http://support.nagios.com/forum