Logstash Integration

ZABBIX

+

Origins

- Jordan Sissel
- Started in 2009
- Open Source (Apache License)
- Jordan joined **Elastic** in August 2013
- Still Open Source
- Will always be Open Source

What is it?

- A tool for receiving, processing and outputting logs, and other data streams.
- Pipeline
  - Input
  - Filter
  - Output

Inputs

- couchdb_changes
- drupal_dblog
- elasticsearch
- exec
- eventlog
- file
- ganglia
- gelf
- generator
- graphite
- github
- heartbeat
- heroku
- http
- http_poller
- irc
- imap
- jdbc
- jmx
- kafka
- log4j
- lumberjack
- meetup
- pipe
- puppet_facter
- relp
- rss
- rackspace
- rabbitmq
- redis
- snmptrap
- stdin
- sqlite
- s3
- sqs
- stomp
- syslog
- tcp
- twitter
- unix
- udp
- varnishlog
- wmi
- websocket
- xmpp
- zenoss
- zeromq

Filters

- aggregate
- alter
- anonymize
- collate
- csv
- cidr
- clone
- cipher
- checksum
- date
- dns
- drop
- elasticsearch
- extractnumbers
- environment
- elapsed
- fingerprint
- geoip
- grok
- i18n
- json
- json_encode
- kv
- mutate
- metrics
- multiline
- metaevent
- prune
- punct
- ruby
- range
- syslog_pri
- sleep
- split
- throttle
- translate
- uuid
- urldecode
- useragent
- xml
- zeromq

Outputs

- boundary
- circonus
- csv
- cloudwatch
- datadog
- datadog_metrics
- email
- elasticsearch
- exec
- file
- google_bigquery
- google_cloud_storage
- ganglia
- gelf
- graphtastic
- graphite
- hipchat
- http
- irc
- influxdb
- juggernaut
- jira
- kafka
- lumberjack
- librato
- loggly
- mongodb
- metriccatcher
- nagios
- null
- nagios_nsca
- opentsdb
- pagerduty
- pipe
- riemann
- redmine
- rackspace
- rabbitmq
- redis
- riak
- s3
- sqs
- stomp
- statsd
- solr_http
- sns
- syslog
- stdout
- tcp
- udp
- webhdfs
- websocket
- xmpp
- **zabbix**
- zeromq

Configuration

```
input {
  plugin_name { settings... }
}
filter {
  plugin_name { settings... }
}
output {
  plugin_name { settings... }
}
```

Inputs

file

Read events from a file in real-time, like `tail`

file

```
file {
  path => "/path/to/logfile"
}
```

tcp

Read from TCP socket

tcp

```
tcp {
  host => "ip or hostname"
  port => 12345
}
```

irc

Capture all or part of the discussion in one or more IRC channels.

irc

```
irc {
    channels => [ "#zabbix" ]
    host => "irc.freenode.org"
    nick => "my_nickname"
    port => 6667
}
```

Inputs

- couchdb_changes
- drupal_dblog
- elasticsearch
- exec
- eventlog
- file
- ganglia
- gelf
- generator
- graphite
- github
- heartbeat
- heroku
- http
- http_poller
- irc
- imap
- jdbc
- jmx
- kafka
- log4j
- lumberjack
- meetup
- pipe
- puppet_facter
- relp
- rss
- rackspace
- rabbitmq
- redis
- snmptrap
- stdin
- sqlite
- s3
- sqs
- stomp
- syslog
- tcp
- twitter
- unix
- udp
- varnishlog
- wmi
- websocket
- xmpp
- zenoss
- zeromq

Filters

grok

Parse arbitrary text and structure it.

grok

- Parse unstructured log data into something structured.
- Perfect for syslog, webserver, & db logs, and in general, any log format that is generally written for humans.
- Ships with 120+ patterns. You can add your own trivially.
- For help building patterns to match your logs:
  - http://grokconstructor.appspot.com/
  - http://grokdebug.herokuapp.com

grok

55.3.244.1 GET /index.html 15824 0.043

```
filter {
  grok {
    match => { "message" => "%{IP:client} %{WORD:method}
%{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" }
  }
}
```

grok

- client: 55.3.244.1
- method: GET
- request: /index.html
- bytes: 15824
- duration: 0.043

grok

Oniguruma

- `(?<field_name>the pattern here)`
- `(?<queue_id>[0-9A-F]{10,11})`

Custom `patterns_dir`

- `# contents of ./patterns/postfix:`
  `POSTFIX_QUEUEID [0-9A-F]{10,11}`

grok

Jan 1 06:25:43 mailserver14 postfix/cleanup[21403]: BEF25A72965: message-id=<20130101142543.5828399CCAF@mailserver14.example.com>

```
filter {
  grok {
    patterns_dir => "./patterns"
    match => { "message" => "%{SYSLOGBASE}
%{POSTFIX_QUEUEID:queue_id}: %{GREEDYDATA:syslog_message}" }
  }
}
```

grok

- timestamp: Jan 1 06:25:43
- logsource: mailserver14
- program: postfix/cleanup
- pid: 21403
- queue_id: BEF25A72965
- syslog_message: message-id=<20130101142543.5828399CCAF@mailserver14.example.com>

date

Convert string-based date formats to date object for easy conversion and export.

date

- syslog events usually have timestamps like this:

  `Apr 17 09:32:01`

- You would use the date format `MMM dd HH:mm:ss` to parse this.

- http://www.joda.org/joda-time/apidocs/org/joda/time/format/DateTimeFormat.html

- Overwrites `@timestamp` by default

date

```
filter {
  # ...grok, etc.
  date {
    match => [ "timestamp", "MMM dd HH:mm:ss" ]
    remove_field => { "timestamp" }
    locale => "en"
  }
  # ...other filters
}
```

date

- **ISO8601** - should parse any valid ISO8601 timestamp, such as 2011-04-19T03:44:01.103Z
- **UNIX** - will parse float or int value expressing unix time in seconds since epoch like 1326149001.132 as well as 1326149001
- **UNIX_MS** - will parse int value expressing unix time in milliseconds since epoch like 1366125117000
- **TAI64N** - will parse tai64n time values

geoip

Look up geographic information by IP

geoip

```
geoip {
  source => "clientip"
}
```

useragent

Parse useragent strings into fields.

useragent

```
useragent {
  source => "useragent"
}
```

OR

```
if [useragent] != "" {
  useragent { source => "useragent" }
}
```

Filters

- aggregate
- alter
- anonymize
- collate
- csv
- cidr
- clone
- cipher
- checksum
- date
- dns
- drop
- elasticsearch
- extractnumbers
- environment
- elapsed
- fingerprint
- geoip
- grok
- i18n
- json
- json_encode
- kv
- mutate
- metrics
- multiline
- metaevent
- prune
- punct
- ruby
- range
- syslog_pri
- sleep
- split
- throttle
- translate
- uuid
- urldecode
- useragent
- xml
- zeromq

Conditionals

if/then/else

```
if EXPRESSION {
  ...
} else if EXPRESSION {
  ...
} else {
  ...
}
```

expressions

Comparison operators:

- equality: **==**, **!=**, **<**, **>**, **<=**, **>=**
- regexp: **=~**, **!~**
- inclusion: **in**, **not in**

Supported boolean operators:

- **and**, **or**, **nand**, **xor**

Supported unary operators:

- **!**

expressions

```
filter {
  if [action] == "login" {
    mutate { remove => "secret" }
  }
}
```

expressions

```
output {
  # Send production errors to Zabbix
  if [loglevel] == "ERROR" and [deployment] ==
"production" {
    zabbix {
    ...
    }
  }
}
```

expressions

```
if [foo] in [foobar] {
if [foo] in "foo" {
if "hello" in [greeting] {
if [foo] in ["hello", "world", "foo"] {
if [missing] in [alsomissing] {
if !("foo" in ["hello", "world"]) {
```

sprintf

- Reference field values within a string:

```
add_field => { "foo" => "%{bar}" }
add_field => { "foo_%{bar}" => "%{baz}" }
```

- Nested fields are referenced with square braces:

```
add_field => {
  "foo" => "%{[@metadata][bar]"
}
```

zabbix

You know, for monitoring.

zabbix

- https://github.com/logstash-plugins/logstash-output-zabbix
- https://www.elastic.co/guide/en/logstash/current/plugins-outputs-zabbix.html
- Community plugin
- Deterministic (derives Zabbix host and key values from events)
- Installation:

```
bin/plugin install logstash-output-zabbix
```

zabbix

- `zabbix_sender` protocol
- Uses `@timestamp`
- Supports sending multiple values per event (most recently added feature)
- Uses native ruby TCP calls (old version used zabbix_sender binary)
- Does not support batching (don't overload your trappers)

options

- zabbix_host
- zabbix_key
- zabbix_value
- zabbix_server_host
- zabbix_server_port
- multi_value
- timeout

zabbix_host

- Type: String
- A single field name which holds the value you intend to use as the Zabbix host name.
- Required value.

zabbix_key

- Type: String
- A single field name which holds the value you intend to use as the Zabbix item key.
- Ignored if using `multi_value`, otherwise required.

zabbix_value

- Type: String
- A single field name which holds the value you intend to send to `zabbix_host`'s `zabbix_key`.
- Default: `"message"` (the whole, original log line)
- Ignored if using `multi_value`, otherwise required.

server

- `zabbix_server_host`

  The IP or resolvable hostname where the Zabbix server is running

  Default: `"localhost"`

- `zabbix_server_port`

  The port on which the Zabbix server is running

  Default: `10051`

multi_value

- Type: Array
- Ignores `zabbix_key` and `zabbix_value`.
- This can be visualized as:

  `[ key1, value1, key2, value2, ... keyN, valueN ]`
- ...where `key1` is an instance of `zabbix_key`, and `value1` is an instance of `zabbix_value`.
- If the field referenced by any `zabbix_key` or `zabbix_value` does not exist, that entry will be ignored.

timeout

- Type: Number
- The number of seconds to wait before giving up on a connection to the Zabbix server.
- Default: 1
- This number should be very small, otherwise delays in delivery of other outputs could result.

zabbix

```
output {
  zabbix {
    zabbix_server_host => "zabbix.example.com"
    zabbix_host => "host_field"
    zabbix_key => "key_field"
    zabbix_value => "value_field"
  }
  # ... Other outputs
}
```

zabbix

```
output {
  if [type] == "zabbix" {
    zabbix {
      zabbix_server_host => "zabbix.example.com"
      zabbix_host => "host_field"
      zabbix_key => "key_field"
      zabbix_value => "value_field"
    }
  }
}
```

zabbix

```
output {
  if [type] == "zabbix" {
    zabbix {
      zabbix_server_host => "zabbix.example.com"
      zabbix_host => "host_field"
      multi_value => [ "k1", "v1", "k2", "v2" ]
    }
  }
}
```

use cases

It's play time!

IRC

- Monitor IRC for catch word or phrase
- Send to Zabbix if the word is given

input

```
input {
  irc {
    channels => [ "#zabbix" ]
    host => "irc.freenode.org"
    nick => "howdy"
    port => 6667
    type => "irc"
  }
}
```

filter

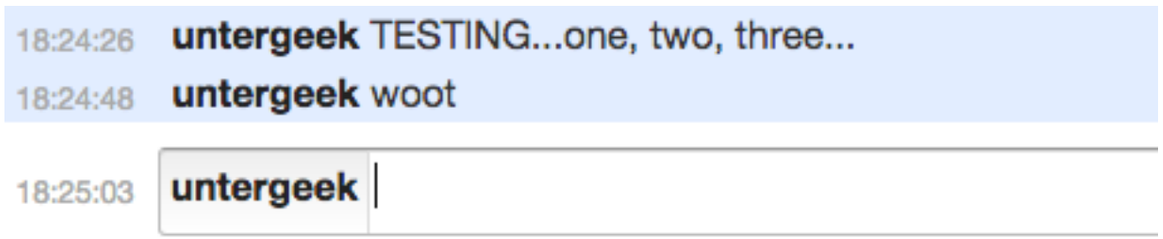
```
if [type] == "irc" {
 if [message] =~ /^.*TESTING.*$/ {
  mutate {
   add_field => { "[@metadata][irc_key]" =>
"message" }
   add_field => { "[@metadata][zabbix_host]" =>
"irc" }
   add_tag => "testing"
 }
}
```

output

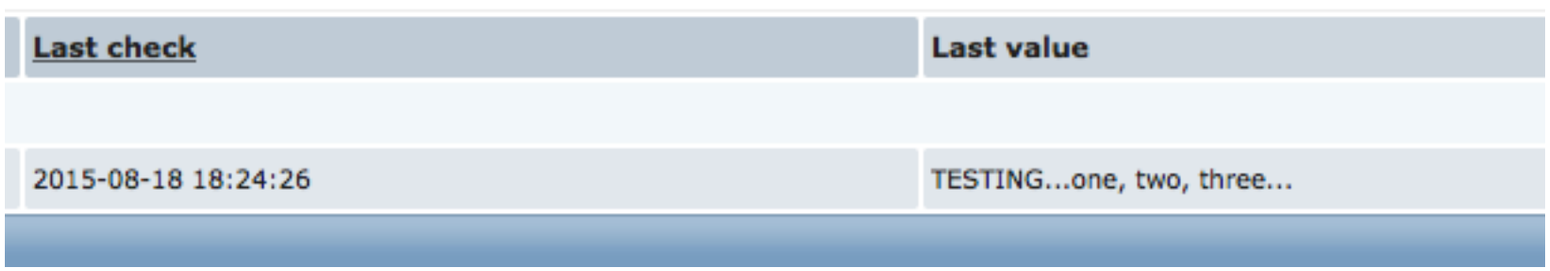
```
if [type] == "irc" and "testing" in [tags] {
  zabbix {
    zabbix_server_host => "localhost"
    zabbix_host => "[@metadata][zabbix_host]"
    zabbix_key => "[@metadata][irc_key]"
    zabbix_value => "message"
  }
}
```

Result

Input (IRCCloud)



Output (Zabbix Frontend)



Last check	Last value
2015-08-18 18:24:26	TESTING...one, two, three...

NGINX

- Capture NGINX logs for virtual hosts
- Watch for error codes (400 - 599)
- Send to Zabbix when one comes in
- Bonus: Send the client IP that generated the code

input

```
input {
  file {
    path => "/path/to/nxinx.log"
    type => "nginx_json"
  }
}
```

filter - pt.1

```
json {
    source => "message"
    remove_field => "message"
}
if [type] == "nginx_json" {
  mutate {
    replace => { "host" => "%{vhost}" }
    remove_field => "vhost"
  }
```

filter - pt.2

```
geoip { source => "clientip" }
if [useragent] != ""  {
  useragent { source => "useragent" }
}
if [referrer]   == "-" {
  mutate { remove_field => "referrer" }
}
```

filter - pt.3

```
if [status] >= 400 and [host] != "localhost" {
  mutate {
    add_field => {
      "[@metadata][status_key]" => "status"
    }
    add_field => {
      "[@metadata][clientip_key]" => "clientip"
    }
```

filter - pt.4

```
    add_field => {
      "[@metadata][error]" => "error[%{status},]"
    }
      add_field => {
        "[@metadata][counter]" => "1"
      }
    }
  }
}
```

output - 1

```
if [type] == "nginx_json" {
    if [status] >= 400 {
      zabbix {
        zabbix_server_host => "localhost"
        zabbix_host => "host"
        zabbix_key => "[@metadata][error]"
        zabbix_value => "[@metadata][counter]"
}
```

	zabbix host	key	value
fieldname	host	[@metadata][error]	[@metadata][counter]
value	untergeek.com	error[404,]	1

output - 2

```
zabbix {
  zabbix_server_host => "localhost"
  zabbix_host => "host"
  multi_value => [
   "[@metadata][status_key]", "status",
   "[@metadata][clientip_key]", "clientip"
  ]
}
```

Result

- Two kinds here:

Name	Last check	Last value
IP (1 Item)		
Client IP	2015-08-18 21:42:13	91.200
Status (5 Items)		
HTTP Error Code 404	2015-08-18 21:16:08	1
HTTP Error Code 405	2015-08-18 19:53:50	1
HTTP Error Code 429	2015-08-18 20:40:00	1
HTTP Error Code 499	2015-08-18 21:42:13	1
HTTP Status	2015-08-18 21:42:13	499

Result

Timestamp	Value
2015-08-18 21:42:13	91.200.
2015-08-18 21:29:57	91.200.
2015-08-18 21:26:46	91.200.
2015-08-18 21:24:32	91.200.
2015-08-18 21:21:38	52.10.2
2015-08-18 21:16:08	66.249.
2015-08-18 21:15:55	66.249.
2015-08-18 21:14:25	66.249.
2015-08-18 21:14:04	123.125
2015-08-18 21:10:01	91.200.

Timestamp	Value
2015-08-18 21:42:13	499
2015-08-18 21:29:57	499
2015-08-18 21:26:46	499
2015-08-18 21:24:32	499
2015-08-18 21:21:38	499
2015-08-18 21:16:08	404
2015-08-18 21:15:55	404
2015-08-18 21:14:25	404
2015-08-18 21:14:04	404
2015-08-18 21:10:01	499

Result

- Just 404s

Timestamp	Value
2015-08-18 21:55:47	1
2015-08-18 21:16:08	1
2015-08-18 21:15:55	1
2015-08-18 21:14:25	1
2015-08-18 21:14:04	1

Conclusion

- https://www.elastic.co/guide/en/logstash/current/index.html
- https://github.com/elastic/logstash
- https://github.com/logstash-plugins/logstash-output-zabbix
- https://discuss.elastic.co/c/logstash
- #logstash on irc.freenode.org