Hello World Template version: 1.0

[[doc-title]]

[[doc-type]]

Version: [[doc-version]]
Release Date: [[doc-release-date]]

1 Logstash

1.1 Logstash

Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash".

1.2 1.0 Instalation

Logstash requires Java 8 or Java 11.

1.2.1 1.1 Installing from a downloaded binary

Logstash binaries are available here. Just unzip the package into desired directory.

1.2.2 1.2 Installing from package repositories

There are repositories for APT and YUM based distributions. To install logstash using APT follow this link or if you use YUM follow this link.

1.2.3 1.3 Running logstash on docker

Docker images for logstash are available from the elastic docker registry. Obtaining logstash for docker can be done with following command

```
docker pull docker.elastic.co/logstash/logstash:7.5.0
```

1.3 2.0 How logstash works

The Logstash event processing pipeline has three stages:

- *inputs*
- *filters*
- *outputs*

Input generates events, filters modify them and outputs ships them elsewhere.

1.3.1 2.1 Inputs

You use inputs to get data into Logstash. Most common inputs are:

- *file: reads from a file on the filesystem*
- *syslog: listens on port514 for syslog messages*
- *redis: reads from a redis server*
- *beats: processes events sent by beats*
    1. *filebeat*
    2. *auditbeat*
    3. *metricbeat*
    4. *...*

Input examples:

```
input { beats { port => 5044 ssl => true ssl_certificate => "/etc/logstash/logstash.crt" ssl_key => "/etc/logstash/logstash.key" ssl_verify_mode => "none" } } input { file { path => "/var/log/*.log" } }
```

For more information about input plugins visit this page.

1.3.2 2.2 Filters

Filters are used to process events. You can combine filters with conditionals for more specific processing.

Some of most used filters are:

- *grok: parse unstructured log data into something structured*
- *mutate: perform transformations on event fields(rename, remove, replace)*
- *drop: drop an event*
- *geoip: add information about geo location of IP addresses*

Filter examples:

The following example will pull out fields from a simple log:

```
55.3.244.1 GET /index.html 15824 0.043 filter { grok { match => { "message" => "%{IP:client} %{WORD:method} %{URIPATHPARAM:request} %{NUMBER:bytes} %{NUMBER:duration}" } } }
```

After the filter the event will have additional fields in it:

- *client: 55.3.244.1*

- *# method: GET*
- *# request: /index.html*
- *# bytes: 15824*
- *# duration: 0.043*

For more information about filter plugins visit this page.

1.3.3 2.3 Outputs

Finally outputs are use to ship events to other services. One event can be sent to multiple outputs. Some of the outputs are:

- *# elasticsearch*
- *# file*

Output examples:

```
output{ elasticsearch{ hosts => "path.to.your.elastic.search.com" index => "auth-logs-%{+YYY.MM.dd}" user => "elastic" password => "xxxxxxxx" } }
```

For more information about availabe outputs visit this page.

1.3.4 2.4 Codecs

Codecs are stream filters that can operate as part of an input or output. A codec plugin changes the data representation of an event. Some of the codecs are:

- *# json: encode or decode data in the json format*
- *# multiline: merge multiple text lines into a single event (java exception and stacktrace).*

For more information about available codecs visit this page.

1.4 3.0 Logstash Configuration

There are two types of configuration files for logstash: pipeline configuration files and settings files.

1.4.1 3.1 Pipeline configuration

Pipeline configuration files define logstash processing pipeline. As it was said earlier logstash pipeline consists of three sections (stages), two of them are required (input, output) and one is optional (filter). In each section you can specify plugins that you need to use and also specify settings for them. You can reference specific event fields in configuration and use conditionals to process events based on criteria. To run logstash with specific configuration files use following command:

```
bin/logstash -f logstash.conf
```

where -f flag represents path to config file or path to folder where the config files are located. Config file can be separeted in mulitple files where each file can represent one section or part of a section.

1.4.1.1 3.1.1 Plugin configuration

Plugin configuration consists of the plugin name followed by a block of settings.

```
input { file { path => "/var/log/messages" type => "syslog" } }
```

More about was said earlier input plugins, , .

1.4.1.1.1 Value types

- *# Lists*

path => ["/var/log/messages", "/var/log/*.log"]

- *# Boolean*

ssl_enable => true

- *# Bytes*

my_bytes => "1113" # 1113 bytes my_bytes => "10MiB" # 10485760 bytes

- *# Codec*

codec => "json"

For more value types go here

1.4.1.2 3.1.2 Accessing Event Data and Fields

As it was said before pipeline consists out of 3 sections (stages): inputs, filters, outputs. Inputs generate events, filters modify them and outputs ship them. All events have properties or as they're called in logstash fields. In logstash you can reference field by a name. Syntax to access the field is [fieldname]. To access nested field full path must be specified [top-level field][nested field]. Click to see example of the event.

```
``` {  "message" => "127.0.0.1 - - [11/Dec/2013:00:01:45 -0800] \"GET /xampp/status.php HTTP/1.1\" 200 3891 \"http://cadenza/xampp/navi.php\" \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"",  "@timestamp" => "2013-12-11T08:01:45.000Z", "@version" => "1", "host" => "cadenza", "clientip" => "127.0.0.1", "ident" => "-", "auth" => "-", "timestamp" => "11/Dec/2013:00:01:45 -0800",  "verb" => "GET",  "request" => "/xampp/status.php",  "httpversion" => "1.1",  "response" => "200", "bytes" => "3891",  "referrer" => "\"http://cadenza/xampp/navi.php\"",  "agent" => "\"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9; rv:25.0) Gecko/20100101 Firefox/25.0\"" } ```
```

Fields values can be accessed from within other strings.

```
output {  file {  path => "/var/log/%{type}.%{+yyyy.MM.dd.HH}"  } }
```

1.4.1.2.1 Conditionals

Conditionals act the same as they do in programming languages. Conditionals support if, else if and else statements and can be nested. It support following comparison operators:

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

and these boolean operators:

- *and*
- *or*
- *nand*
- *xor and unary operator*
- *!*

Examples of conditional uses:

```
output {  if [loglevel] == "ERROR" and [deployment] == "production" {  pagerduty {  ...  }  } }  filter{  if [foo] in ["hello", "world", "foo"] {  mutate { add_tag => "field in list" }  } }  output {  if "_grokparsefailure" not in [tags] {  elasticsearch { ... }  } }
```

1.4.1.3 3.1.3 Configuration example

```
input {  file {  path => "/tmp/access_log"  start_position => "beginning"  } }  filter {  if [path] =~ "access" {  mutate { replace => { "type" => "apache_access" } }  grok {  match => { "message" => "%{COMBINEDAPACHELOG}" }  }  }  date {  match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]  } }  output {  elasticsearch {  hosts => ["localhost:9200"]  }  stdout { codec => rubydebug } }
```

1.4.2 3.2 Settings files

With logstash installation comes a couple of config files:

- *logstash.yml*
    1. *contains config flags, values can be set instead of passing them from command line. More info.*
- *pipelines.yml*
    1. *contains instructions for running multiple pipelines in a single logstash instance. More info.*
- *jvm.options*
    1. *contains jvm configuration flags. Most used for setting heap space.*
- *log4j.properties*
    1. *contains settings for log4j. More info*
- *startup.options*

1.4.3 3.3 Configuring logstash for docker

1.4.3.1 3.3.1 Pipeline configuration

Pipeline configuration should be placed in "/usr/share/logstash/pipeline/" which is the default place where container looks for configuration files. Configuration files can be provided by following command:

```
docker run --rm -it -v ~/pipeline/:/usr/share/logstash/pipeline/ docker.elastic.co/logstash/logstash:7.5.0
```

Every file in the host directory ~/pipeline/ will then be parsed by Logstash as pipeline configuration. If no configuration files are provided logstash uses minimal configuration.
```

1.4.3.2 3.3.2 Settings configuration

Similar to pipeline configuration, settings files are expected to be in "/usr/share/logstash/config/" folder. They can also be provided by docker run command:

```
docker run --rm -it -v ~/settings/:/usr/share/logstash/config/ docker.elastic.co/logstash/logstash:7.5.0
```

this command provide whole folder, a single file can also be provided.

1.4.3.3 3.3.3 Custom images

Bind-mounted configuration is not the only option. Custom image with custom configuration can be created using dockerfile:

```
FROM docker.elastic.co/logstash/logstash:7.5.0 RUN rm -f /usr/share/logstash/pipeline/logstash.conf ADD pipeline/ /usr/share/logstash/pipeline/ ADD config/ /usr/share/logstash/config/
```

Be sure to replace or delete logstash.conf in your custom image.

For more info visit this page.

1.5 4.0 Running logstash

Logstash can be run from command line, as a service or on docker.

1.5.1 4.1 Running from command line

Logstash can be run with a simple command

```
bin/logstash [options]
```

where options are flags that can specified. All options can be specified in logstash.yml file.

1.5.2 4.2 Running logstash on docker

Docker images for logstash are available from the elastic docker registry. Obtaining logstash for docker can be done with following command

```
docker pull docker.elastic.co/logstash/logstash:7.5.0
```


PUZZLES
CLOUD