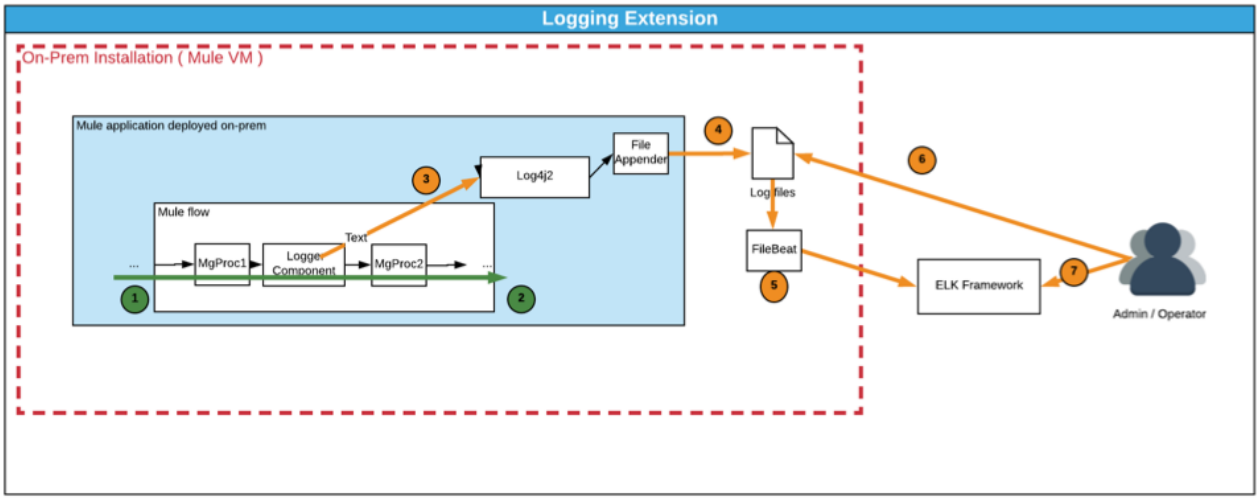
Logging Framework Extension Using ELK

1 Introduction

This guide provides a high-level overview of creating a logging framework extension using ELK (Elastic – Logstash – Kibana) and Filebeat that can be used with a Hybrid deployment model.

Hybrid deployment models do not necessarily require analytics and insights to be pushed to ELK since the cloud management plane can be used for the purpose. This guide covers the steps to set-up log aggregation and visualizations using Filebeat and ELK. This guide does not cover the installation of the ELK stack and provides necessary guidance on Filebeat and Logstash configurations.

The logging framework is explained below:



1) The message enters in a mule flow and it is executed by the message processors chain

2) The message exits the mule flow.

3) At a certain point, a log component is placed in the flow and the log message is routed to the logging framework.

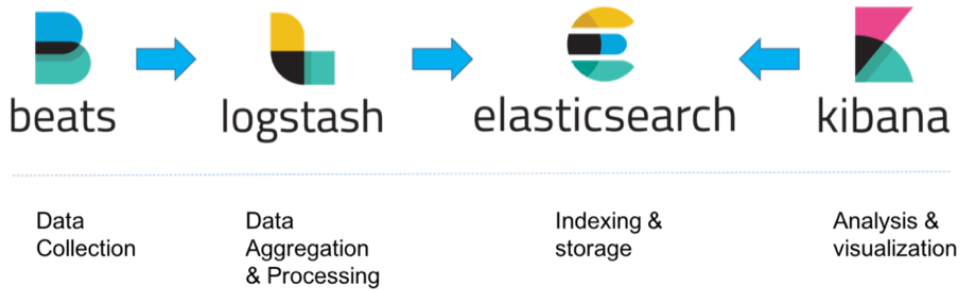
4) The logging framework, based on the Log4j2 component will receive the message and, asynchronously, the message is routed to the Log4j2 appenders. The Log4j2 component is configured to write files on-prem.

5) FileBeat is configured to push the on-premise logs to the ELK Framework.

6) The operator can see the logs files created in the Mule Runtimes on prem.

7) The operator / admin can also use ELK user interface to see the information contained in the logs.

The Elastic Stack Architecture is summarized as below:



Please note :

1. Filebeat is installed in the Mule server.
2. ELK stack is generally installed in a separate server.

2 Product Versions

Filebeat : 6.6.1

Logstash : 6.6.0

ElasticSearch : Version: 6.6.0, Build: default/tar/a9861f4/2019-01-24T11:27:09.439740Z, JVM: 1.8.0_191

Kibana: 6.6.0

3 Configurations

3.1 Filebeat

Version: *6.6.1*

Configuration File Path :

The filebeat configurations are stored in a ymal file :

<*filebeat_install_dir*>/filebeat.yml

Instructions:

1. Add the mule runtime logs to the filebeat.inputs

```
#========================== Filebeat inputs =============================
filebeat.inputs:
# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input specific configurations.
- type: log
  # Change to true to enable this input configuration.
  enabled: true
  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /Users/ndas/Documents/installation/mule-enterprise-standalone-4.1.3.2/logs/*.log. 1
```

Replace 1 with the Mule Runtime logs path

2. Add the multi-line support and define the pattern for log aggregation

```
### Multiline options
  fields: 2
    type: mule-runtime
  multiline.pattern: ^(INFO|WARN|ERROR|DEBUG|FATAL)  [0-9]{4}-[0-9]{2}-[0-9]{2} |[0-9]{4}-[0-9]{2}-[0-9]{2}
[0-9]{2}:[0-9]{2}:[0-9]{2}
  multiline.negate: true
  multiline.match: after
```

3. Additionally, we set the following processor (drop_event) to drop truncated lines on rotating logs


```
#============================== Processors =====================================
# Configure processors to enhance or manipulate events generated by the beat.
processors:
 - add_host_metadata: ~
 - add_cloud_metadata: ~
 - drop_event: 3
     when:
       not:
         regexp:
           message: '^(INFO|WARN|ERROR|DEBUG|FATAL)  [0-9]{4}-[0-9]{2}-[0-9]{2} |[0-9]{4}-[0-9]{2}-[0-9]
{2} [0-9]{2}:[0-9]{2}:[0-9]{2}'
```

4. For the output, we will change the host and ssl configurations of logstash

```
#------------------------- Logstash output -------------------------------
output.logstash:
 # The Logstash hosts
 hosts: ["<your-hostname>:5044"] 4

 # Optional SSL. By default is off.
 # List of root certificates for HTTPS server verifications
 #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

 # Certificate for SSL client authentication
 #ssl.certificate: "/etc/pki/client/cert.pem"

 # Client Certificate Key
 #ssl.key: "/etc/pki/client/cert.key"
```

Replace 4 with the host and port of the logstash instance

3.2 Logstash

Logstash will be configured

1. To set up a Beats -> Logstash -> Elasticsearch pipeline
2. To add necessary filters (date and grok) to preprocess the Mule runtime log formats

Configuration File Path :

<logstash_install_dir>/config/logstash.conf

Instructions :

1. Update the filebeat port in the input

2. Update the elasticsearch host and user/password in the output

3. Update the date and grok filters to pre-process mule runtime logs format


```
# Sample Logstash configuration for creating a simple
# Beats -> Logstash -> Elasticsearch pipeline.

input {
beats {. (1)
port => 5044
}
}

output {
elasticsearch {
hosts => ["http://<your-hostname>:9200"] (2)
index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
#user => "elastic"
#password => "changeme"
}
}


filter { (3)
date {
match => [ "logdate", "YYYY-MM-dd HH:mm:ss,SSS" ]
target => "logdate"
}
grok {
match => { "message" => "%{LOGLEVEL:log-level}  %{TIMESTAMP_ISO8601:logdate} \[%
{DATA:component}\] \[event: %{DATA:logcorrelationID}\] %{JAVACLASS:class}:%
{GREEDYDATA:message}" }
overwrite => ["message"]
}
}
```

Note:

The Mule runtime default log format is assumed for the above filters to work as expected.

```
INFO 2019-03-19 16:52:58,760 [AsyncHttpClient-Callback] [event: ]
org.mule.runtime.core.internal.context.DefaultMuleContext: Lets test multiline and logstash.
And view the expected results.
```

Alternatively , if you using JSON logger instead to log information , the below filter expression should work. The pattern checks if the message is of JSON format before applying the JSON filter , so as to handle mixed logs formats.


```
# Sample Logstash configuration for creating a simple
# Beats -> Logstash -> Elasticsearch pipeline.

input {
beats {. (1)
port => 5044
}
}

output {
elasticsearch {
hosts => ["http://<your-hostname>:9200"] (2)
index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
#user => "elastic"
#password => "changeme"
}
}


filter { (3)
date {
match => [ "logdate", "YYYY-MM-dd HH:mm:ss,SSS" ]
target => "logdate"
}
grok {
match => { "message" => "%{LOGLEVEL:log-level}  %{TIMESTAMP_ISO8601:logdate} \[%
{DATA:component}\] \[event: %{DATA:logcorrelationID}\] %{JAVACLASS:class}:%
{GREEDYDATA:message}" }
overwrite => ["message"]
}
if [message] =~ "\s*\{.+\}\s*" {
json {
   source => "message"
   skip_on_invalid_json => true
  }
 }
}
```

Sample Log Format:

```
INFO 2019-04-01 14:05:49,081 [[MuleRuntime].cpuLight.10: [testtruststore].testFlow.CPU_LITE
@45b48a62] [event: 0-3146e5b0-5444-11e9-bd52-8c85904803da]
org.mule.extension.jsonlogger.JsonLoggerExtension: Overriding the CategoryName
INFO 2019-04-01 14:05:49,297 [[MuleRuntime].cpuLight.10: [testtruststore].testFlow.CPU_LITE
@45b48a62] [event: 0-3146e5b0-5444-11e9-bd52-8c85904803da]
org.mule.extension.jsonlogger.JsonLogger: {
"applicationName" : "testtruststore",
"applicationVersion" : "1.0.0-SNAPSHOT",
"category" : "vg.vp.log",
"clientIP" : "/127.0.0.1:58450",
"content" : "{\n \"payload\": \"\",\n \"attributes\": {\n \"clientCertificate\": null,\n \"listenerPath\": \"/api/
hello\",\n \"requestPath\": \"/api/hello\",\n \"queryString\": \"\",\n \"remoteAddress\": \"/
```


127.0.0.1:58450\",\n \"method\": \"GET\",\n \"scheme\": \"http\",\n \"relativePath\": \"/hello\",\n \"localAddress\": \"localhost/127.0.0.1:8089\",\n \"requestUri\": \"/api/hello\",\n \"uriParams\": {\n \n },\n \"queryParams\": {\n \n },\n \"headers\": {\n \"host\": \"localhost:8089\",\n \"accept\": \"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\",\n \"upgrade-insecure-requests\": \"1\",\n \"cookie\": \"splunkweb_csrf_token_8000=2186996807111800484; BAMBOO-MAX-DISPLAY-LINES=25; _ga=GA1.1.517657043.1534402393; _ga=GA1.1.517657043.1534402393; AJS.conglomerate.cookie=\\\"|tabContainer.tabContainer.selectedTab=Capabilities|config.sidebar.planNavigator.expanded=false|tabContainer.remote-agents-tabs.selectedTab=Online%20remote%20agents|tabContainer.relies_36_41_0323.selectedTab=Agents|tabContainer.relies_36_43_0334.selectedTab=Agents\\\"; BAMBOO-BUILD-FILTER=LAST_25_BUILDS\",\n \"user-agent\": \"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1 Safari/605.1.15\",\n \"accept-language\": \"en-us\",\n \"accept-encoding\": \"gzip, deflate\",\n \"connection\": \"keep-alive\"\n },\n \"version\": \"HTTP/1.1\"\n }\n}",
"correlationId" : "0-3146e5b0-5444-11e9-bd52-8c85904803da",
"elapsed" : "2",
"endPointURL" : "/api/hello",
"environment" : "dev",
"headers" : {
"accept" : "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
"accept-encoding" : "gzip, deflate",
"accept-language" : "en-us",
"connection" : "keep-alive",
"cookie" : "splunkweb_csrf_token_8000=2186996807111800484; BAMBOO-MAX-DISPLAY-LINES=25; _ga=GA1.1.517657043.1534402393; _ga=GA1.1.517657043.1534402393; AJS.conglomerate.cookie=\"|tabContainer.tabContainer.selectedTab=Capabilities|config.sidebar.planNavigator.expanded=false|tabContainer.remote-agents-tabs.selectedTab=Online%20remote%20agents|tabContainer.relies_36_41_0323.selectedTab=Agents|tabContainer.relies_36_43_0334.selectedTab=Agents\"; BAMBOO-BUILD-FILTER=LAST_25_BUILDS",
"host" : "localhost:8089",
"upgrade-insecure-requests" : "1",
"user-agent" : "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_6) AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.1 Safari/605.1.15"
},
"httpMethod" : "GET",
"locationInfo" : {
"component" : "json-logger:logger",
"fileName" : "testtruststore.xml",
"lineInFile" : "24",
"location" : "testFlow/processors/0",
"rootContainer" : "testFlow"
},
"message" : "Start of the Flow test",
"priority" : "INFO",
"threadName" : "[MuleRuntime].cpuLight.10: [testtruststore].testFlow.CPU_LITE @45b48a62",
"timestamp" : "2019-04-01T06:05:48.982Z",
"tracePoint" : "START"
}

4. Restart logstash

5. Optional , if visualization is not available on the new fields created as part of the grok filter above – refresh the field list (*Management → Index Pattern → Refresh FieldList*)

Kibana

Once Kibana is running the only required configuration will be to specify the index template. In this case, we will use **filebeat-***

Optionally , you can create visualizations and dashboards as required by the operational and monitoring requirements.

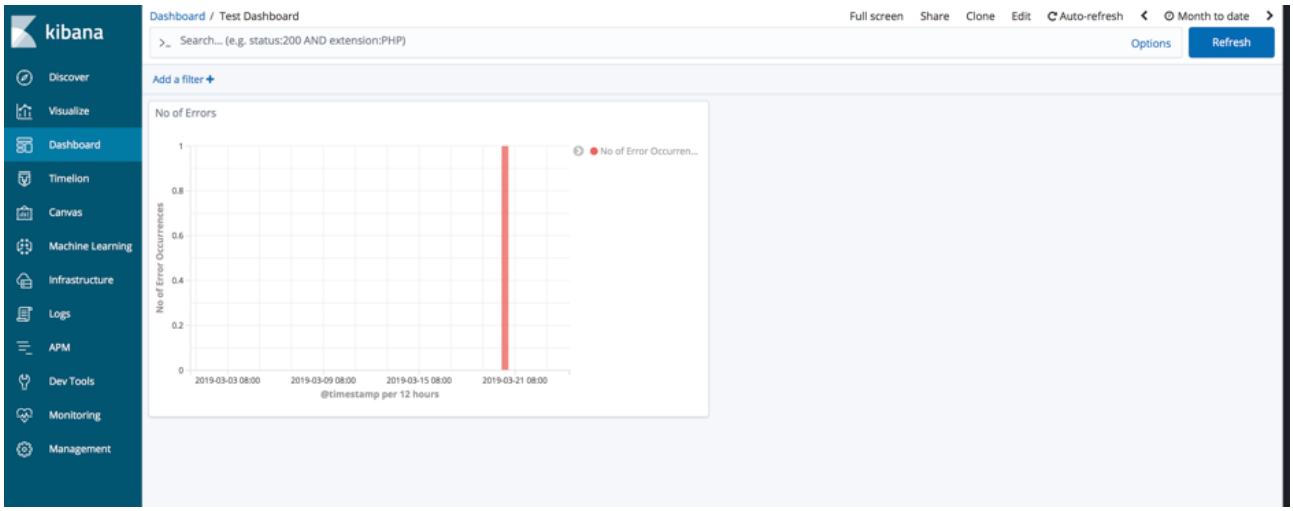


Figure 1: Custom Visualization and Dashboard in Kibana

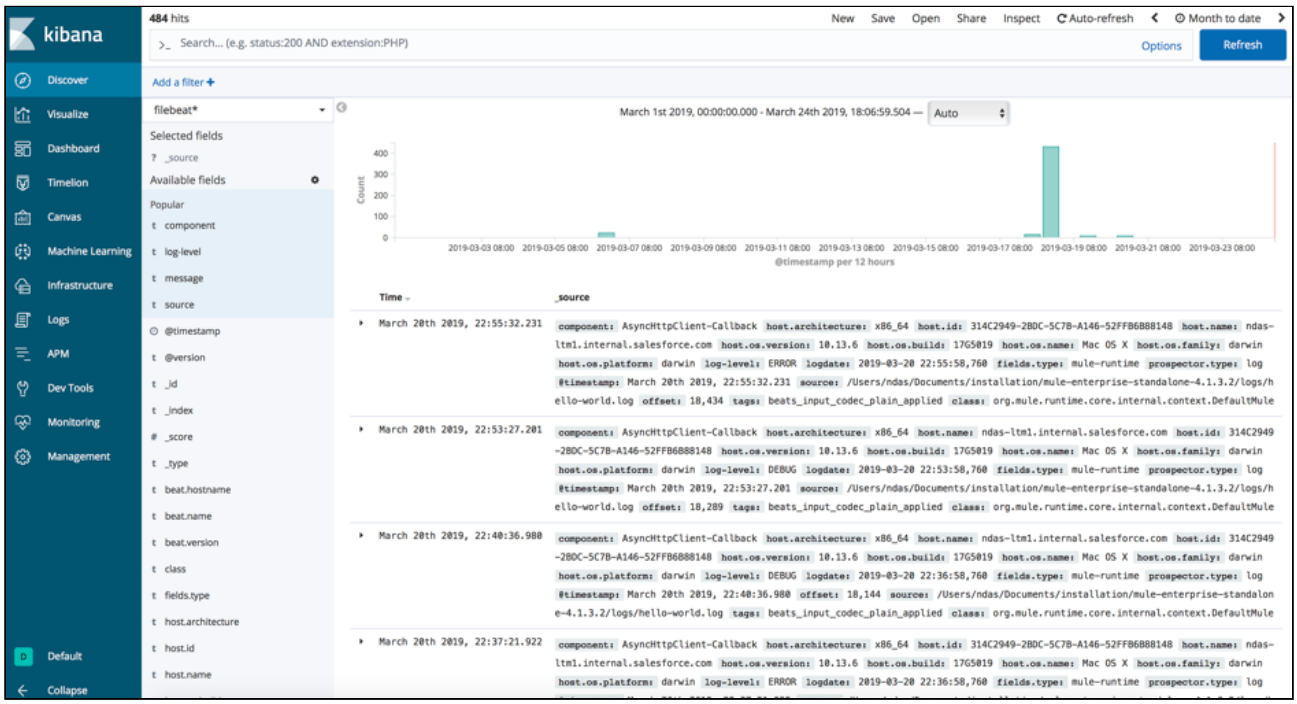


Figure 2: Aggregated Mule Runtime logs with pre-processed fields