Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

Using Telegraf to Parse Custom Logs

objective:

- read custom logs with telegraf
- parse custom logs with telegraf
- generate an output from custom logs to influxdb

The TIG stack [telegraf, influxdb, grafana]

Why Telegraf?

There are many benefits with Telegraf, some are :

- open-source and purely written in golang
- many input and output plugins
- logparser input plugin which supports grok patterns from elastic search's logstash
- light log shipper and analyzer
- analyze logs before exporting them to database
- processor plugins

Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

Create docker compose file

docker-compose.yml

```
version: "2"
services:
  influxdb:
    container_name: influxdb
    image: influxdb:latest
    ports:
      - "8083:8083"
      - "8086:8086"
    volumes:
      - /home/navidx/dockerTIGstack/core/volumes/influxdb:/var/lib/influxdb
    restart: always

  grafana:
    container_name: grafana
    image: grafana/grafana:latest
    ports:
      - "3000:3000"
    links:
      - influxdb
    restart: always

  telegraf:
    container_name: telegraf
    image: telegraf:latest
    network_mode: "host"
    volumes:
      - /home/navidx/dockerTIGstack/telegraf.conf:/etc/telegraf/telegraf.conf
      - /var/run/docker.sock:/var/run/docker.sock
    restart: always
```

This will bring up the TIG stack. For ease of use the telegraf.conf file influxdb volume are mapped to the host machine.

Just type this command in the directory : *docker-compose up*

Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

Sample log file format we want to parse

```
Lmmdd hh:mm:ss.uuuuuu threadid file:line] msg…


where the fields are defined as follows:
  L              A single character, representing the log level (eg 'I' for INFO)
  INFO —> I
  WARNING —> W
  ERROR —> E
  FATAL —> F
  mm               The month (zero padded; ie May is '05')
  dd             The day (zero padded)
  hh:mm:ss.uuuuuu  Time in hours, minutes and fractional seconds
  threadid       The space-padded thread ID as returned by GetTID()
  file           The file name
  line           The line number
  msg            The user-supplied message
```

Example:

```
E1112 09:50:08.004566   10803 ztp_handshake_codec_type.go:25] constructor error: CRC32_UNKNOWN
```

First we need to read the logfile with Telegraf:

https://github.com/influxdata/telegraf/blob/release-1.8/plugins/inputs/file/README.md

https://github.com/influxdata/telegraf/blob/release-1.8/plugins/inputs/tail/README.md

but this is not enough, we also want to parse the custom_log format into meaningful fields for InfluxDB to further query them…

so here the logparser comes in hand

simply logparser uses grok style patterns like ElasticSearch's Logstash which also supports regex patterns.

https://github.com/influxdata/telegraf/tree/master/plugins/inputs/logparser

The Telegraf grok parser uses a slightly modified version of logstash "grok" patterns, with the format ==> `%{<capture_syntax>[:<semantic_name>][:<modifier>]}`

Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

note:

Telegraf has many of its own built-in patterns, as well as support for most of logstash's builtin patterns. Golang regular expressions do not support lookahead or lookbehind. logstash patterns that depend on these are not supported.

If you need help building patterns to match your logs, you will find the https://grokdebug.herokuapp.com application quite useful!

Grok pattern tutorial:

https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html

https://grokdebug.herokuapp.com/patterns#

https://github.com/influxdata/telegraf/blob/master/plugins/inputs/logparser/grok/patterns/influx-patterns

https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns

For learning regex use :

https://regexone.com/

https://regexr.com/

some notes:

Output file bug:

https://github.com/influxdata/telegraf/issues/4018

Excluding from grok:

https://stackoverflow.com/questions/43150897/using-grok-to-skip-parts-of-message-or-logs

A multi-line literal string allows us to encode the pattern:

```
[[inputs.logparser]]
  [inputs.logparser.grok]
    patterns = ['''
          \|%{NUMBER:value:int}\|%{UNICODE_ESCAPE:escape}\|'%{WORD:name}'\|
        ''']
```

Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

```
custom_patterns = 'UNICODE_ESCAPE (?:\\u[0-9A-F]{4})+'
```

GOLANG for parsing custom time formats (magic numbers):

https://stackoverflow.com/questions/14106541/go-parsing-date-time-strings-which-are-not-standard-formats

https://golang.org/src/time/format.go

Back to the original problem

So here we have two main objectives

- parse the custom time format to meaningful timestamp for influxdb
- parse the rest of the fields to influxdb

For parsing custom time formats

https://github.com/influxdata/telegraf/tree/master/plugins/inputs/logparser#timestamp-examples

using custom time formats with golang magic numbers here is the pattern for the time format:

```
%{MY_TIME:timestamp:ts-"0102 15:04:05.000000"}
```

the overall logparser input config will be like this :

```
[[inputs.logparser]]
  files = ["/home/data.log"]
  from_beginning = true

  [inputs.logparser.grok]
    #patterns = ["%{COMBINED_LOG_FORMAT}"]
    patterns = ['''
    ^%{TYPE_NV:type}%{MY_TIME:timestamp:ts-"0102 15:04:05.000000"}\s+%
{MY_PID:PID:int}\s%{MY_FILE:file}:%{MY_LINENUM:lineNum:int}] %
{MY_MESSEGE:messege}
''']
    ## Name of the outputted measurement name.
    measurement = "new_custom_log"
```

Navid Malek
navidmalekedu@gmail.com
navidmalek.blog.ir

```
## Full path(s) to custom pattern files.
#custom_pattern_files = []

## Custom patterns can also be defined here. Put one pattern per line.
custom_patterns = '''
  TYPE_NV [A-Z]
  #MY_TIME %{MONTHNUM}%{MONTHDAY} %{TIME}
  #MY_TIME %{MONTHNUM:mounth:int}%{MONTHDAY:day:int} %{TIME:hhmmss}
  MY_NANOSEC (\d{2}:\d{2}:\d{2}\.\d{6})
  MY_TIME %{MONTHNUM}%{MONTHDAY} %{MY_NANOSEC}
  MY_PID (\d+)
  MY_FILE (.*)
  MY_LINENUM (\d+)
  MY_MESSEGE (.*)
'''
```

The generated output in file format:

```
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG PID=10803I,FILE="APP.GO",LINENUM=34I,MESSEGE="CREA$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG TYPE="I",PID=10803I,FILE="GRPC_CLIENT.GO",LINENUM=$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG PID=10803I,FILE="APP.GO",LINENUM=38I,MESSEGE="APP $ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG FILE="APP.GO",LINENUM=43I,MESSEGE="STARTED...",TYP$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG FILE="TCP_SERVER.GO",LINENUM=108I,MESSEGE="REGISTE$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG PID=10803I,FILE="TCP_SERVER.GO",LINENUM=111I,MESSE$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG FILE="DAEMON.GO",LINENUM=17I,MESSEGE="ONSERVERNEWC$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG PID=10803I,FILE="ZPROTO_SERVER.GO",LINENUM=70I,MES$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG TYPE="W",PID=10803I,FILE="DAEMON.GO",LINENUM=21I,M$ + timestamp
NEW_CUSTOM_LOG,HOST=NAVIDX-THINKPAD-E460,PATH=/HOME/DATA.LOG MESSEGE="CONSTRUCTOR ERROR: CRC32_UNKNOWN",TYPE="E$ + timestamp
```

The generated output in influxdb measurement (query on time and type):

```
> select time,type from new_custom_log
name: new_custom_log
time                        type
----                        ----
2018-11-12T09:49:23.80004Z  I
2018-11-12T09:49:23.800544Z I
2018-11-12T09:49:23.8009Z   I
2018-11-12T09:49:23.802324Z I
2018-11-12T09:49:23.803011Z I
2018-11-12T09:49:23.803027Z I
2018-11-12T09:49:33.47316Z  I
2018-11-12T09:50:08.003999Z I
2018-11-12T09:50:08.004139Z W
2018-11-12T09:50:08.004566Z E
>
```