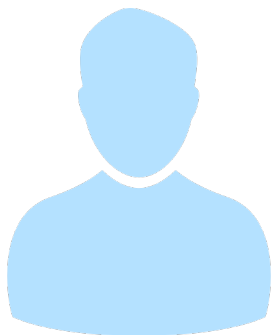


Extending syslog-ng in Python: Best of both worlds

Peter Czanik / syslog-ng, a One Identity business



About me



Peter Czanik from Hungary

Evangelist at One Identity: syslog-ng upstream

syslog-ng packaging, support, advocacy

syslog-ng originally developed by Balabit, now part of One Identity

Overview

- What is syslog-ng
- The four roles of syslog-ng
- Configuring syslog-ng for Python
- Python source, parser, destination

syslog-ng

Logging

Recording events, such as:

Jan 14 11:38:48 linux-0jbu sshd[7716]: Accepted publickey for root from 127.0.0.1 port 48806 ssh2

syslog-ng

Enhanced logging daemon with a focus on portability and high-performance central log collection. Originally developed in C.

Python

Makes syslog-ng slower but gives easy development and flexibility.

Main syslog-ng roles



Collector



Processor



Filter



Storage
(or forwarder)

Role: data collector

Collect system and application logs together: contextual data for either side

A wide variety of platform-specific sources:

- /dev/log & co
- Journal, Sun streams

Receive syslog messages over the network:

- Legacy or RFC5424, UDP/TCP/TLS

Logs or any kind of text data from applications:

- Through files, sockets, pipes, application output, etc.

Python source: Jolly Joker

HTTP server, Amazon CloudWatch fetcher, Kafka source, etc.

Role: processing

Classify, normalize, and structure logs with built-in parsers:

- CSV-parser, PatternDB, JSON parser, key=value parser

Rewrite messages:

- For example: anonymization

Reformatting messages using templates:

- Destination might need a specific format (ISO date, JSON, etc.)

Enrich data:

- GeoIP
- Additional fields based on message content

Python parser: all of above, enrich logs from databases and also filtering

Role: data filtering

Main uses:

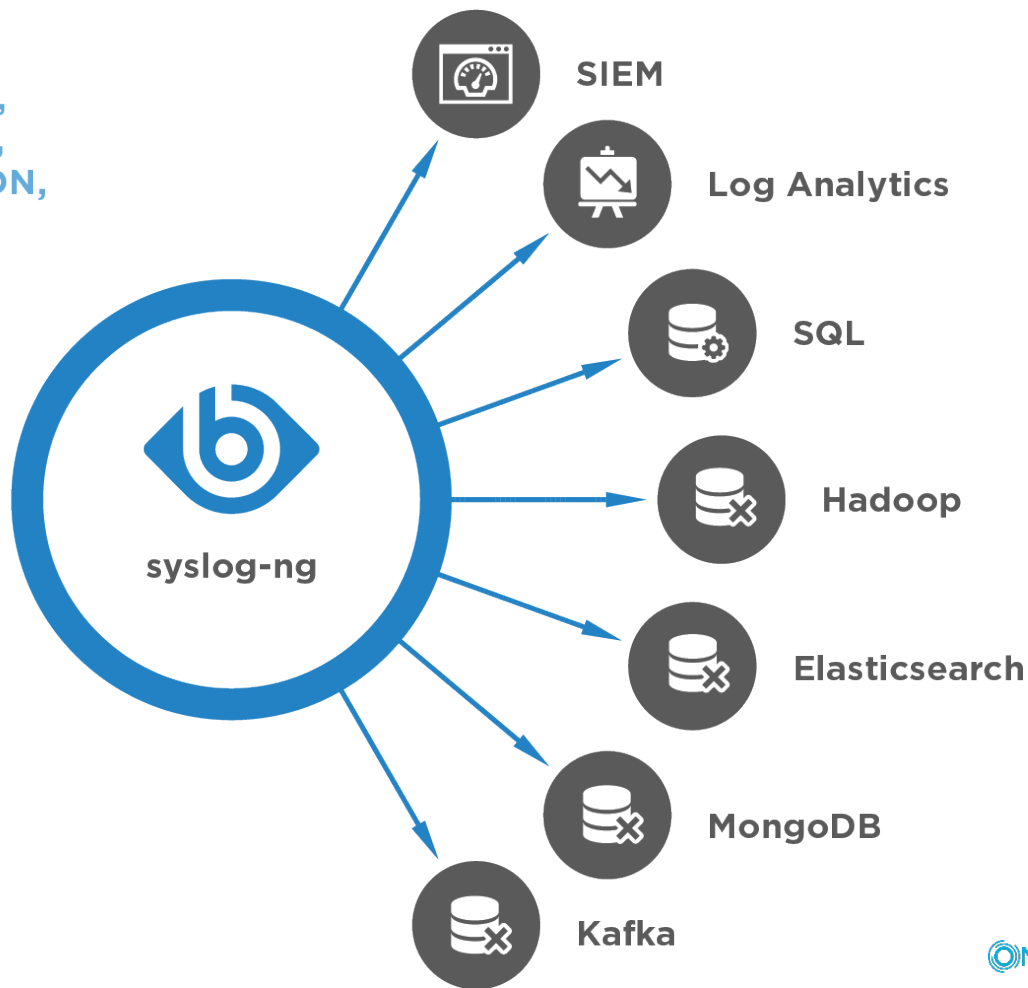
- Discarding surplus logs (not storing debug-level messages)
- Message routing (login events to SIEM)

Many possibilities:

- Based on message content, parameters, or macros
- Using comparisons, wildcards, regular expressions, and functions
- Combining all of these with Boolean operators

Role: destinations

syslog-ng,
EventLog,
Journal, JSON,
TXT, CSV

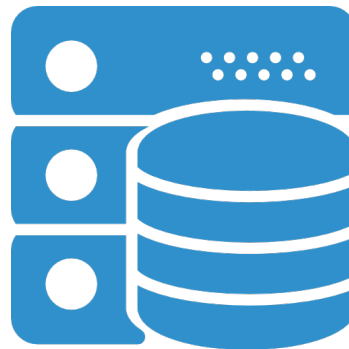


Freeform log messages

Most log messages are: date + hostname + text

Mar 11 13:37:56 linux-6965 sshd[4547]: Accepted keyboard-interactive/pam for root from 127.0.0.1 port 46048 ssh2

- Text = English sentence with some variable parts
- Easy to read by a human
- Difficult to create alerts or reports



Solution: structured logging

Events represented as name-value pairs. For example, an ssh login:

```
app=sshd user=root source_ip=192.168.123.45
```

syslog-ng: name-value pairs inside

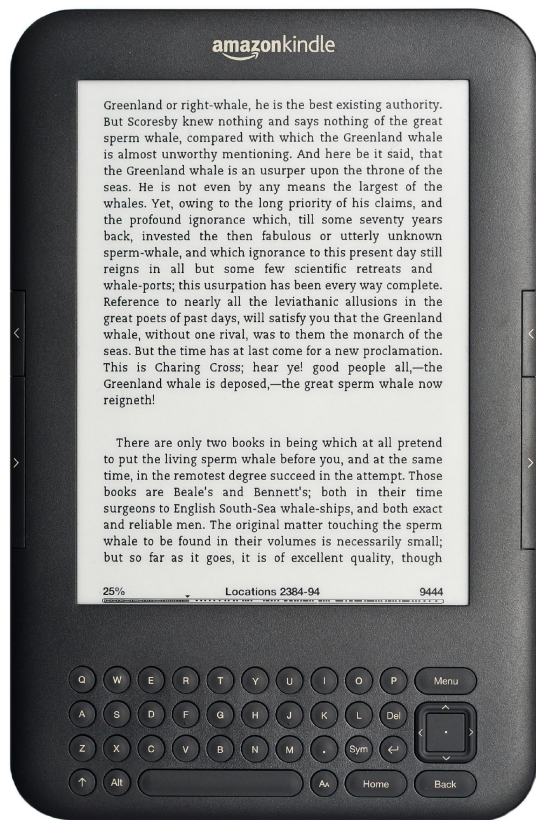
Date, facility, priority, program name, pid, etc.

Parsers in syslog-ng can turn unstructured and some structured data (CSV, JSON) into name-value pairs

Python bindings fully support name-value pairs

Which is the most used version?

- Project started in 1998
- RHEL EPEL has version 3.5
- Latest stable version is 3.19, released a month ago



Kindle e-book reader

Version 1.6

Configuration

- “Don't Panic”
- Simple and logical, even if it looks difficult at first
- Pipeline model:
 - Many different building blocks (sources, destinations, filters, parsers, etc.)
 - Connected into a pipeline using “log” statements

syslog-ng.conf: getting started

```
@version:3.19
```

```
@include "scl.conf"
```

```
# this is a comment :)
```

```
options {flush_lines (0); keep_hostname (yes);};
```

```
source s_sys { system(); internal();};
```

```
destination d_mesg { file("/var/log/messages"); };
```

```
filter f_default { level(info..emerg) and not (facility(mail)); };
```

```
log { source(s_sys); filter(f_default); destination(d_mesg); };
```

Python in syslog-ng

- Python bindings: configuration + code
- Can pass parameters to Python code
- Only the class name is mandatory in config
- Python code can be in-line in a python {} block, or stored in external file(s)

Python destination: mandatory

- Only the class name is mandatory in config
- Only send() method is mandatory
- Name-value pairs as
 - object – all
 - dict – only those configured

Python destination: optional

- Many non-mandatory options, like disk-buffer, etc.
- `init()` and `deinit()`
 - When syslog-ng started or reloaded
- `open()` and `close()`
 - start/reload or when sending fails

Python.conf: a simple file destination

```
destination d_python_to_file {  
    python(  
        class("TextDestination")  
    );  
};  
  
log {  
    source(src);  
    destination(d_python_to_file);  
};  
  
python {  
class TextDestination(object):  
    def send(self, msg):  
        self.outfile = open("/tmp/example.txt", "a")  
        self.outfile.write("MESSAGE = %s\n" % msg["MESSAGE"])  
        self.outfile.flush()  
        self.outfile.close();  
        return True  
};
```

Python parser

- [illegible]

Python parser: config

```
parser my_python_parser{
  python(
    class("SngRegexParser")
    options("regex", "seq: (?P<seq>\\d+), thread: (?P<thread>\\d+), runid: (?P<runid>\\d+), stamp: (?P<stamp>[^ ]+) (?P<padding>.*$)")
  );
};

log {
  source { tcp(port(5555)); };
  parser(my_python_parser);
  destination {file("/tmp/regexparser.log.txt" template("seq: $seq thread: $thread runid: $runid stamp: $stamp my_counter: $MY_COUNTER\\n"));
};
};
```

Python parser: code

```
python {
```

```
import re
```

```
class SngRegexParser(object):
```

```
def init(self, options):
```

```
    """
```

```
    Initializes the parser
```

```
    """
```

```
    pattern = options["regex"]
```

```
    self.regex = re.compile(pattern)
```

```
    self.counter = 0
```

```
    return True
```

Python parser: code continued

```
def deinit(self):  
    pass  
def parse(self, log_message):  
    decoded_msg = log_message['MESSAGE'].decode('utf-8')  
    match = self.regex.match(decoded_msg)  
    if match:  
        for key, value in match.groupdict().items():  
            log_message[key] = value  
        log_message['MY_COUNTER'] = str(self.counter)  
        self.counter += 1  
        return True  
    return False  
};
```

Python source

- Options, like time zone handling
- Name-value pairs as object
- Two modes
 - server
 - fetcher (syslog-ng handles the eventloop)
- The run() and request_exit() methods are mandatory (for the server)

Simple source

```
source s_python {  
    python(  
        class("MySource")  
        options(  
            "option1" "value1",  
            "option2" "value2"  
        )  
    );  
};
```

```
destination d_file { file("/var/log/python.txt"); };
```

```
log { source(s_python); destination(d_file); };
```

Simple source continued

```
python {  
    from syslogng import LogSource  
    from syslogng import LogMessage  
  
    class MySource(LogSource):  
        def init(self, options): # optional  
            print("init")  
            print(options)  
            self.exit = False  
            return True  
  
        def run(self): # mandatory  
            print("run")  
            while not self.exit:  
                msg = LogMessage("this is a log message")  
                self.post_message(msg)  
  
        def request_exit(self): # mandatory  
            print("exit")  
            self.exit = True  
};
```

Debugging

- Logging to internal() from Python code
- Coming up in syslog-ng 3.20

```
import syslogng  
logger = syslogng.Logger()  
logger.error("plain text message: ERROR")  
logger.warning("plain text message: WARNING")  
logger.info("plain text message: INFO")  
logger.debug("plain text message: DEBUG")
```

Further examples

- MQTT destination: <https://www.syslog-ng.com/community/b/blog/posts/writing-python-destination-in-syslog-ng-how-to-send-log-messages-to-mqtt>
- Parsers: <https://www.syslog-ng.com/community/b/blog/posts/parsing-log-messages-with-the-syslog-ng-python-parser>
- HTTP source:
<https://www.syslog-ng.com/community/b/blog/posts/creating-an-http-source-for-syslog-ng-in-python>

Join the community!

- syslog-ng: <http://syslog-ng.org/>
- Source on GitHub: <https://github.com/balabit/syslog-ng>
- Mailing list: <https://lists.balabit.hu/pipermail/syslog-ng/>
- Gitter: <https://gitter.im/balabit/syslog-ng>



Questions?

syslog-ng blog: <https://syslog-ng.com/community/>

My e-mail: peter.czanik@oneidentity.com

Twitter: <https://twitter.com/PCzanik>

