beaver Documentation

Release 36.3.0

Jose Diaz-Gonzalez

Oct 14, 2018

Contents

Release v36.3.0. (*Installation*)

Beaver is an *MIT* licensed python daemon that munches on logs and sends their contents to logstash.

Contents:

CHAPTER 1

Introduction

1.1 Background

Beaver provides an lightweight method for shipping local log files to Logstash. It does this using redis, zeromq, tcp, udp, rabbit or stdout as the transport. This means you'll need a redis, zeromq, tcp, udp, amqp or stdin input somewhere down the road to get the events.

Events are sent in logstash's `json_event` format. Options can also be set as environment variables.

1.2 MIT License

A large number of open source projects you find today are GPL Licensed. While the GPL has its time and place, it should most certainly not be your go-to license for your next open source project.

A project that is released as GPL cannot be used in any commercial product without the product itself also being offered as open source.

The MIT, BSD, ISC, and Apache2 licenses are great alternatives to the GPL that allow your open-source software to be used freely in proprietary, closed-source software.

Beaver is released under terms of MIT License.

1.3 Beaver License

CHAPTER 2

Installation

This part of the documentation covers the installation of Beaver. The first step to using any software package is getting it properly installed.

2.1 Requirements

- Python 2.6+
- Optional zeromq support: install libzmq (`brew install zmq` or `apt-get install libzmq-dev`) and pyzmq (`pip install pyzmq==2.1.11`)

2.2 Distribute & Pip

Installing Beaver is simple with pip:

```
$ pip install beaver
```

or, with easy_install:

```
$ easy_install beaver
```

But, you really shouldn't do that.

2.3 Cheeseshop (PyPI) Mirror

If the Cheeseshop (a.k.a. PyPI) is down, you can also install Beaver from one of the mirrors. Crate.io is one of them:

```
$ pip install -i http://simple.crate.io/ beaver
```

2.4 Get the Code

Beaver is actively developed on GitHub, where the code is always available.

You can either clone the public repository:

```
git clone git://github.com/python-beaver/python-beaver.git
```

Download the tarball:

```
$ curl -OL https://github.com/python-beaver/python-beaver/tarball/master
```

Or, download the zipball:

```
$ curl -OL https://github.com/python-beaver/python-beaver/zipball/master
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

CHAPTER 3

Usage

The `beaver` binary contains a number of options:

usage:

```
beaver [-h] [-c CONFIG] [-C CONFD_PATH] [-d] [-D] [-f FILES [FILES ...]]
       [-F {json,msgpack,raw,rawjson,string}] [-H HOSTNAME] [-m {bind,connect}]
       [-l OUTPUT] [-p PATH] [-P PID]
       [-t {kafka,mqtt,rabbitmq,redis,sns,sqs,kinesis,stdout,tcp,udp,zmq,stomp}] [-v]␣
↪[--fqdn]
```

optional arguments:

```
-h, --help            show this help message and exit
-c CONFIG, --configfile CONFIG
                      main beaver ini config file path
-C CONFD_PATH         ini config directory path
-d, --debug           enable debug mode
-D, --daemonize       daemonize in the background
-f FILES [FILES ...], --files FILES [FILES ...]
                      space-separated filelist to watch, can include globs
                      (*.log). Overrides --path argument
-F {json,msgpack,raw,rawjson,string,gelf}, --format {json,msgpack,raw,rawjson,string,
↪gelf}
                      format to use when sending to transport
-H HOSTNAME, --hostname HOSTNAME
                      manual hostname override for source_host
-m {bind,connect}, --mode {bind,connect}
                      bind or connect mode
-l OUTPUT, --logfile OUTPUT, -o OUTPUT, --output OUTPUT
                      file to pipe output to (in addition to stdout)
-p PATH, --path PATH  path to log files
-P PID, --pid PID     path to pid file
-t {kafka,mqtt,rabbitmq,redis,sns,sqs,kinesis,stdout,tcp,udp,zmq,stomp}, --transport
↪{kafka,mqtt,rabbitmq,redis,sns,sqs,kinesis,stdout,tcp,udp,zmq,stomp}
                      log transport method
```

(continues on next page)

(continued from previous page)

```
-v, --version         output version and quit
--fqdn                use the machine's FQDN for source_host
```

3.1 Configuration File Options

Beaver can optionally get data from a `configfile` using the `-c` flag. This file is in `ini` format. Global configuration will be under the `beaver` stanza. The following are global beaver configuration keys with their respective meanings:

- kafka_client_id: Default `beaver-kafka`. Kafka client id
- kafka_hosts: Default `localhost:9092`. Seed list of hosts (host:port)separated by commas for kafka cluster
- kafka_async: Default `True`.
- kafka_topic: Default `logstash-topic`
- kafka_key: Optional. Defaults `None`. Target specific partition
- kafka_codec: Optional. Defaults `None`. GZIP supported with 0x01. SNAPPY requires to install python-snappy anbd use codec = 0x02
- kafka_ack_timeout: Default `2000`. Acknowledge timeout
- kafka_batch_n: Default `10`. Batch log message size
- kafka_batch_t: Default `10`. Batch log message timeout
- mqtt_host: Default `localhost`. Host for mosquitto
- mqtt_port: Default `1883`. Port for mosquitto
- mqtt_clientid: Default `paho`. Paho client id
- mqtt_keepalive: Default `60`. mqtt keepalive ping
- mqtt_topic: Default `/logstash`. Topic to publish to
- number_of_consumer_processes: Default `1`. Number of parallel consumer processes that read and process messages from the beaver queue.
- rabbitmq_arguments: Defaults `{}`. RabbitMQ arguments comma separated, colon separated key value pairs. i.e `rabbitmq_arguments: x-max-length:750000,x-max-length-bytes:1073741824`
- rabbitmq_host: Defaults `localhost`. Host for RabbitMQ
- rabbitmq_port: Defaults `5672`. Port for RabbitMQ
- rabbitmq_ssl: Defaults `0`. Connect using SSL/TLS
- rabbitmq_ssl_key: Optional. Defaults `None`. Path to client private key for SSL/TLS
- rabbitmq_ssl_cert: Optional. Defaults `None`. Path to client certificate for SSL/TLS
- rabbitmq_ssl_cacert: Optional. Defaults `None`. Path to CA certificate for SSL/TLS
- rabbitmq_vhost: Default `/`
- rabbitmq_username: Default `guest`
- rabbitmq_password: Default `guest`
- rabbitmq_queue: Default `logstash-queue`.
- rabbitmq_queue_durable: Default `0`.

- rabbitmq_exchange_type: Default `direct`.
- rabbitmq_exchange_durable: Default `0`.
- rabbitmq_key: Default `logstash-key`.
- rabbitmq_exchange: Default `logstash-exchange`.
- rabbitmq_timeout: Default `1`. The timeout in seconds for the connection to the RabbitMQ broker
- rabbitmq_delivery_mode: Default `1`. Message deliveryMode. 1: non persistent 2: persistent
- redis_url: Default `redis://localhost:6379/0`. Comma separated redis URLs
- redis_namespace: Default `logstash:beaver`. Redis key namespace
- redis_data_type: Default `list`, but can also be `channel`. Redis data type used for transporting log messages
- sns_aws_access_key: Can be left blank to use IAM Roles or AWS_ACCESS_KEY_ID environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- sns_aws_secret_key: Can be left blank to use IAM Roles or AWS_SECRET_ACCESS_KEY environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- sns_aws_profile_name: Can be left blank to use IAM Roles AWS_SECRET_ACCESS_KEY environment variable, or fixed keypair (above) (see: https://github.com/boto/boto#getting-started-with-boto)
- sns_aws_region: Default `us-east-1`. AWS Region
- sns_aws_topic_arn: Topic ARN (must exist)
- sqs_aws_access_key: Can be left blank to use IAM Roles or AWS_ACCESS_KEY_ID environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- sqs_aws_secret_key: Can be left blank to use IAM Roles or AWS_SECRET_ACCESS_KEY environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- sqs_aws_profile_name: Can be left blank to use IAM Roles AWS_SECRET_ACCESS_KEY environment variable, or fixed keypair (above) (see: https://github.com/boto/boto#getting-started-with-boto)
- sqs_aws_region: Default `us-east-1`. AWS Region
- sqs_aws_queue: SQS queue, or comma delimited list of queues (must exist)
- sqs_aws_queue_owner_acct_id: Optional. Defaults `None`. Account ID or Principal allowed to write to queue
- sqs_bulk_lines: Optional. Send multiple log entries in a single SQS message (cost saving benefit on larger environments)
- kinesis_aws_access_key: Can be left blank to use IAM roles or AWS_ACCESS_KEY_ID environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- kinesis_aws_secret_key: Can be left blank to use IAM Roles or AWS_SECRET_ACCESS_KEY environment variable (see: https://github.com/boto/boto#getting-started-with-boto)
- kinesis_aws_region: Default `us-east-1`. AWS Region
- kinesis_aws_stream: Optional. Defaults `None`. Name of the Kinesis stream to ship logs to
- kinesis_aws_batch_size_max: Default `512000`. Arbitrary flush size to limit size of logs in transit.
- tcp_host: Default `127.0.0.1`. TCP Host
- tcp_port: Default `9999`. TCP Port
- tcp_ssl_enabled: Defaults `0`. Connect using SSL/TLS
- tcp_ssl_key Optional. Defaults `None`. Path to client private key for SSL/TLS

- tcp_ssl_cert Optional. Defaults `None`. Path to client certificate for SSL/TLS
- tcp_ssl_cacert Optional. Defaults `None`. Path to CA certificate for SSL/TLS
- udp_host: Default `127.0.0.1`. UDP Host
- udp_port: Default `9999`. UDP Port
- zeromq_address: Default `tcp://localhost:2120`. Zeromq URL
- zeromq_hwm: Default None. Zeromq HighWaterMark socket option
- zeromq_bind: Default `bind`. Whether to bind to zeromq host or simply connect
- http_url: Default `None` http://someserver.com/path
- stomp_host: Default `localhost`
- stomp_port: Default `61613`
- stomp_user: Default `None`
- stomp_password: Default `None`
- stomp_queue: Default `queue/logstash`

The following are used for instances when a TransportException is thrown - Transport dependent

- respawn_delay: Default `3`. Initial respawn delay for exponential backoff
- max_failure: Default `7`. Max failures before exponential backoff terminates
- max_queue_size: Default `100`. Max log entries Beaver can store in it's queue before backing off until they have been transmitted

The following configuration keys are for SinceDB support. Specifying these will enable saving the current line number in an sqlite database. This is useful for cases where you may be restarting the Beaver process, such as during a logrotate.

- sincedb_path: Default `None`. Full path to an `sqlite3` database. Will be created at this path if it does not exist. Beaver process must have read and write access

Logstash 1.2 introduced a JSON schema change. The `logstash_version` needs to be set or Beaver will fail to start

- logstash_version: No default. Set to `0` for older versions, `1` for Logstash v1.2 and above

The following configuration keys are for building an SSH Tunnel that can be used to proxy from the current host to a desired server. This proxy is torn down when Beaver halts in all cases.

- ssh_key_file: Default `None`. Full path to `id_rsa` key file
- ssh_tunnel: Default `None`. SSH Tunnel in the format `user@host:port`
- ssh_tunnel_port: Default `None`. Local port for SSH Tunnel
- ssh_remote_host: Default `None`. Remote host to connect to within SSH Tunnel
- ssh_remote_port: Default `None`. Remote port to connect to within SSH Tunnel
- ssh_options: Default `None`. Comma separated list of SSH options to Pass through to the SSH Tunnel. See `ssh_config(5)` for more options

The following configuration keys are for multi-line events support and are per file.

- multiline_regex_after: Default `None`. If a line match this regular expression, it will be merged with next line(s).
- multiline_regex_before: Default `None`. If a line match this regular expression, it will be merged with previous line(s).

The following can also be passed via argparse. Argparse will override all options in the configfile, when specified.

- format: Default `json`. Options `[ json, msgpack, string, raw, rawjson, gelf ]`. Format to use when sending to transport
- files: Default `files`. Space-separated list of files to tail. (Comma separated if specified in the config file)
- path: Default `/var/log`. Path glob to tail.
- transport: Default `stdout`. Transport to use when log changes are detected
- fqdn: Default `False`. Whether to use the machine's FQDN in transport output
- hostname: Default `None`. Manually specified hostname

The following configuration key allows cleaning up the worker and transport sub-processes on an interval respawning

- refresh_worker_process: Default `None`. Interval between sub-process cleanup

The following configuration key allows the importing of OS environment data into the event.

- add_field_env: Default `None`. Format is fieldname1,ENVVARIABLE1[,fieldname2,ENVVARIABLE2, . . .]

The following configuration key allows to set a redis_namespace per files stanza. It will override the global [beaver] setting fo the same key.

- redis_namespace: Defaults to Null string. Redis key namespace

3.2 Examples

As you can see, `beaver` is pretty flexible as to how you can use/abuse it in production:

3.2.1 Default Usage

Listen to all files in the default path of /var/log on standard out as json:

```
beaver
```

3.2.2 Alternative output formats

Listen to all files in the default path of /var/log on standard out with json:

```
# adds data to a json object before shipping
beaver --format json
```

Listen to all files in the default path of /var/log on standard out with msgpack:

```
beaver --format msgpack
```

Listen to all files in the default path of /var/log on standard out as a raw:

```
# ships with no formatting
beaver --format raw
```

Listen to all files in the default path of /var/log on standard out as a raw:

```
# ships with no formatting
beaver --format raw
```

Listen to all files in the default path of /var/log on standard out as a [Raw Json Support](http://blog.pkhamre.com/2012/08/23/logging-to-logstash-json-format-in-nginx/):

```
# also adds any extra data specified in config
beaver --format raw
```

Listen to all files in the default path of /var/log on standard out as a string:

```
# Useful for stdout debugging
# Output format is:
#
#   '[{host}] [{timestamp}] {message}'
beaver --format string
```

3.2.3 Configuration files

Read config from config.ini and put to stdout:

```
# /etc/beaver/conf:
; follow a single file, add a type, some tags and fields
[/tmp/somefile]
type: mytype
tags: tag1,tag2
add_field: fieldname1,fieldvalue1[,fieldname2,fieldvalue2, ...]

; follow all logs in /var/log except those with `messages` or `secure` in the name.
; The exclude tag must be a valid python regular expression.
[/var/log/*log]
type: syslog
tags: sys
exclude: (messages|secure)

; follow /var/log/messages.log and /var/log/secure.log using file globbing
[/var/log/{messages,secure}.log]
type: syslog
tags: sys

# From the commandline
beaver -c /etc/beaver/conf -t stdout
```

Loading stanzas from /etc/beaver/conf.d/* support:

```
# /etc/beaver/conf
[beaver]
format: json

# /etc/beaver/conf.d/syslog
[/var/log/syslog]
type: syslog
tags: sys,main

# /etc/beaver/conf.d/nginx
```

(continues on next page)

(continued from previous page)

```
[/var/log/nginx]
format: rawjson
type: nginx
tags: nginx,server

# From the commandline
beaver -c /etc/beaver/conf -C /etc/beaver/conf.d
```

3.2.4 Shipping to a broker

Sending logs from /var/log files to a redis list:

```
# /etc/beaver/conf
[beaver]
redis_url: redis://localhost:6379/0

# From the commandline
beaver  -c /etc/beaver/conf -t redis
```

Sending logs from /var/log files to multiple redis servers using round robin strategy:

```
# /etc/beaver/conf
[beaver]
redis_url: redis://broker01:6379/0,redis://broker02:6379/0,redis://broker03:6379/0

# From the commandline
beaver  -c /etc/beaver/conf -t redis
```

Sending logs from /tmp/somefile files to a redis list, with custom namespace:

```
# /etc/beaver/conf
[beaver]
redis_url: redis://localhost:6379/0

[/tmp/somefile]
type: mytype
tags: tag1,tag2
redis_namespace: some:space

# From the commandline
beaver  -c /etc/beaver/conf -t redis
```

Zeromq listening on port 5556 (all interfaces):

```
# /etc/beaver/conf
[beaver]
zeromq_address: tcp://*:5556

# logstash indexer config:
input {
  zeromq {
    type => 'shipper-input'
    mode => 'client'
    topology => 'pushpull'
    address => 'tcp://shipperhost:5556'
```

(continues on next page)

(continued from previous page)

```
  }
}
output { stdout { debug => true } }

# From the commandline
beaver  -c /etc/beaver/conf -m bind -t zmq
```

Zeromq connecting to remote port 5556 on indexer:

```
# /etc/beaver/conf
[beaver]
zeromq_address: tcp://indexer:5556

# logstash indexer config:
input {
  zeromq {
    type => 'shipper-input'
    mode => 'server'
    topology => 'pushpull'
    address => 'tcp://*:5556'
  }
}
output { stdout { debug => true } }

# on the commandline
beaver -c /etc/beaver/conf -m connect -t zmq
```

Real-world usage of Redis as a transport:

```
# in /etc/hosts
192.168.0.10 redis-internal01
192.168.0.11 redis-internal02

# /etc/beaver/conf
[beaver]
redis_url: redis://redis-internal01:6379/0,redis://redis-internal02:6379/0
redis_namespace: app:unmappable

# logstash indexer01 config:
input {
  redis {
    host => 'redis-internal01'
    data_type => 'list'
    key => 'app:unmappable'
    type => 'app:unmappable'
  }
}
output { stdout { debug => true } }

# logstash indexer02 config:
input {
  redis {
    host => 'redis-internal02'
    data_type => 'list'
    key => 'app:unmappable'
    type => 'app:unmappable'
  }
```

(continues on next page)

(continued from previous page)

```
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -f /var/log/unmappable.log -t redis
```

RabbitMQ connecting to defaults on remote broker:

```
# /etc/beaver/conf
[beaver]
rabbitmq_host: 10.0.0.1

# logstash indexer config:
input { amqp {
    name => 'logstash-queue'
    type => 'direct'
    host => '10.0.0.1'
    exchange => 'logstash-exchange'
    key => 'logstash-key'
    exclusive => false
    durable => false
    auto_delete => false
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t rabbitmq
```

Kafka transport:

```
# /etc/beaver/conf
[beaver]
kafka_client_id: beaver-kafka-1
kafka_hosts: kafkahost1:9092,kafkahost2:9092
kafka_key: logstash
kafka_topic: mylogs-topic
kafka_batch_n: 10
kafka_batch_t: 10

# logstash indexer config:
input {
  kafka {
    zk_connect => 'zk1:2181' # string (optional), default: "localhost:2181"
    group_id => 'logstash' # string (optional), default: "logstash"
    topic_id => 'mylogs-topic' # string (optional), default: "test"
    reset_beginning => false # boolean (optional), default: false
    consumer_threads => 25 # number (optional), default: 1
    queue_size => 20 # number (optional), default: 20
    rebalance_max_retries => 4 # number (optional), default: 4
    rebalance_backoff_ms => 2000 # number (optional), default:  2000
    consumer_timeout_ms => -1 # number (optional), default: -1
    consumer_restart_on_error => true # boolean (optional), default: true
    consumer_restart_sleep_ms => 0 # number (optional), default: 0
    decorate_events => false # boolean (optional), default: false
    consumer_id => 'logstash-kafka-1' # string (optional) default: nil
    fetch_message_max_bytes => 1048576 # number (optional) default: 1048576
```

(continues on next page)

(continued from previous page)

```
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t kafka
```

TCP transport:

```
# /etc/beaver/conf
[beaver]
tcp_host: 127.0.0.1
tcp_port: 9999
format: raw

# logstash indexer config:
input {
  tcp {
    host => '127.0.0.1'
    port => '9999'
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t tcp
```

UDP transport:

```
# /etc/beaver/conf
[beaver]
udp_host: 127.0.0.1
udp_port: 9999

# logstash indexer config:
input {
  udp {
    type => 'shipper-input'
    host => '127.0.0.1'
    port => '9999'
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t udp
```

SNS Transport:

```
# /etc/beaver/conf
[beaver]
sns_aws_region: us-east-1
sns_aws_topic_arn: arn:aws:sns:us-east-1:123456789123:logstash-topic
sns_aws_access_key: <access_key>
sns_aws_secret_key: <secret_key>
sns_aws_profile_name: <proflie_name>
```

(continues on next page)

(continued from previous page)

```
# logstash indexer config:
input {
  sqs {
    queue => "sns-subscriber"
    type => "shipper-input"
    format => "json_event"
    access_key => "<access_key>"
    secret_key => "<secret_key>"
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t sns
```

SQS Transport:

```
# /etc/beaver/conf
[beaver]
sqs_aws_region: us-east-1
sqs_aws_queue: logstash-input1,logstash-input2
sqs_aws_access_key: <access_key>
sqs_aws_secret_key: <secret_key>

# logstash indexer config:
input {
  sqs {
    queue => "logstash-input"
    type => "shipper-input"
    format => "json_event"
    access_key => "<access_key>"
    secret_key => "<secret_key>"
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -t sqs
```

Kinesis Transport:

```
# /etc/beaver/conf
[beaver]
kinesis_aws_region: us-east-1
kinesis_aws_stream: logstash-stream
kinesis_aws_access_key: <access_key>
kinesis_aws_secret_key: <secret_key>

# ingest process (not via Logstash): https://github.com/awslabs/amazon-kinesis-
↪connectors

# From the commandline
beaver -c /etc/beaver/conf -t kinesis
```

Mqtt transport using Paho:

```
# /etc/beaver/conf
[beaver]
mqtt_client_id: 'beaver_client'
mqtt_topic: '/logstash'
mqtt_host: '127.0.0.1'
mqtt_port: '1318'
mqtt_keepalive: '60'

# logstash indexer config:
input {
  mqtt {
    host => '127.0.0.1'
    data_type => 'list'
    key => 'app:unmappable'
    type => 'app:unmappable'
  }
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -f /var/log/unmappable.log -t mqtt
```

HTTP transport

The HTTP transport simply posts the payload data for a log event to the url specified here. You can use this to post directly to elastic search, for example by creating an index and posting json to the index URL:

```
# Assuming an elastic search instance running on your localhost,
# create a 'logs' index:
curl -XPUT 'http://localhost:9200/logs/'

# A beaver config to post directly to elastic search:
# /etc/beaver/conf
[beaver]
format: json
logstash_version: 1
http_url: http://localhost:9200/logs/log

# From the commandline
beaver -c /etc/beaver/conf -F json -f /var/log/somefile -t http
```

GELF using HTTP transport

To ship logs directly to a Graylog server, start with this configuration:

```
# /etc/beaver/conf, GELF HTTP input on port 12200
[beaver]
http_url: 'http://graylog.example.com:12200/gelf'

# From the commandline
beaver -c /etc/beaver/conf -f /var/log/somefile -t http -F gelf
```

Stomp transport using Stomp.py:

```
# /etc/beaver/conf
[beaver]
stomp_host: 'localhost'
stomp_port: '61613'
```

(continues on next page)

(continued from previous page)

```
stomp_user: 'producer-user'
stomp_password : 'password'
stomp_queue : 'queue/logstash'

# logstash indexer config:
stomp {
    user => "consumer-user"
    password => "consumer-password"
    destination => "logstash"
    host => "localhost"
    port => "61613"
}
output { stdout { debug => true } }

# From the commandline
beaver -c /etc/beaver/conf -f /var/log/somefile.log -t stomp
```

3.2.5 Sincedb support using Sqlite3

Note that this will require R/W permissions on the file at sincedb path, as Beaver will store the current line for a given filename/file id.:

```
# /etc/beaver/conf
[beaver]
sincedb_path: /etc/beaver/since.db

[/var/log/syslog]
type: syslog
tags: sys,main
sincedb_write_interval: 3 ; time in seconds

# From the commandline
beaver -c /etc/beaver/conf
```

3.2.6 Multi-line Parsing

Simple multi-line event: if line is indented it is the continuation of an event:

```
# /etc/beaver/conf
[/tmp/somefile]
multiline_regex_before = ^\s+
```

Multi-line event for Python traceback:

```
# /etc/beaver/conf
[/tmp/python.log]
multiline_regex_after = (^\s+File.*, line \d+, in)
multiline_regex_before = (^Traceback \(most recent call last\):)|(^\s+File.*, line␣
↪\d+, in)|(^\w+Error: )

# /tmp/python.log
DEBUG:root:Calling faulty_function
WARNING:root:An error occured
```

(continues on next page)

(continued from previous page)

```
Traceback (most recent call last):
  File "doerr.py", line 12, in <module>
    faulty_function()
  File "doerr.py", line 7, in faulty_function
    0 / 0
ZeroDivisionError: integer division or modulo by zero
```

3.2.7 SSH Tunneling Support

Use SSH options for redis transport through SSH Tunnel:

```
# /etc/beaver/conf
[beaver]
transport: redis
redis_url: redis://localhost:6379/0
redis_namespace: logstash:beaver
ssh_options: StrictHostKeyChecking=no, Compression=yes, CompressionLevel=9
ssh_key_file: /etc/beaver/remote_key
ssh_tunnel: remote-logger@logs.example.net
ssh_tunnel_port: 6379
ssh_remote_host: 127.0.0.1
ssh_remote_port: 6379
```

3.2.8 Environment Import Support

Using add_field_env allows you to add additional fields based upon OS environment data. For example if you want the instance ID from an AWS host (and you've imported that data into the environment before launch), you could add the following:

add_field_env: instanceID,INSTANCE_ID

If you're using Asgard to manage your auto scaling groups, you can extract the information that it sets as well.

add_field_env: asgEnvironment,CLOUD_DEV_PHASE,launchConfig,CLOUD_LAUNCH_CONFIG,asgName,CLOUD_CLUSTER

Assuming the following items in the environment:

```
# printenv | egrep '(CLOUD_LAUNCH_CONFIG|CLOUD_CLUSTER|INSTANCE_ID)'
CLOUD_CLUSTER=always-cms-services-ext-d0prod
CLOUD_LAUNCH_CONFIG=always-cms-services-ext-d0prod-20131030104814
INSTANCE_ID=i-3cf70c0b
```

And the following beaver.conf file:

```
[beaver]
tcp_host: 10.21.52.249
tcp_port: 9999
format: json

[/mnt/logs/jetty/access.log]
type: cms-serv-ext
tags: beaver-src
add_field_env: launchConfig, CLOUD_LAUNCH_CONFIG, cloudCluster, CLOUD_CLUSTER,␣
↪instanceID, INSTANCE_ID
```

You would get the following event in your logstash input (using tcp for an input with an oldlogstashjson codec):

```
{
         "message" => "10.21.56.244 - - [07/11/2013:22:44:15 +0000] \"GET / HTTP/1.1\
↪" 200 14108 \"-\" \"NING/1.0\"",
     "source_host" => "ip-10-21-56-68",
     "source_path" => "/mnt/logs/jetty/access.log",
          "source" => "file://ip-10-21-56-68/mnt/logs/jetty/access.log",
            "tags" => [
        [0] "beaver-src"
    ],
            "type" => "cms-serv-ext",
      "@timestamp" => "2013-11-07T22:44:15.068Z",
      "instanceID" => "i-3cf70c0b",
    "cloudCluster" => "always-cms-services-ext-d0prod",
    "launchConfig" => "always-cms-services-ext-d0prod-20131030104814",
        "@version" => "1",
            "host" => "10.21.56.68:36952"
}
```

This is functionally equivalent to the logstash environment filter. The information format with add_field_env is slightly different than add_field. The add_field keyword will add the values to an array within logstash, whereas add_field_env passes it as a string. You end up with a key => value pair, just as you would in the source system's environment.

Sample of data from add_field::

myKey => [[0] "myValue"

],

Sample of data from add_field_env:: myKey => "myValue"

CHAPTER 4

Caveats

File tailing and shipping has a number of important issues that you should be aware of. These issues may affect just Beaver, or may affect other tools as well.

4.1 Copytruncate

When using `copytruncate` style log rotation, two race conditions can occur:

1. Any log data written prior to truncation which Beaver has not yet read and processed is lost. Nothing we can do about that.
2. Should the file be truncated, rewritten, and end up being larger than the original file during the sleep interval, Beaver won't detect this. After some experimentation, this behavior also exists in GNU tail, so I'm going to call this a "don't do that then" bug :)

   Additionally, the files Beaver will most likely be called upon to watch which may be truncated are generally going to be large enough and slow-filling enough that this won't crop up in the wild.

4.2 FreeBSD

When you get an error similar to `ImportError: No module named _sqlite3` your python seems to miss the sqlite3-module. This can be the case on FreeBSD and probably other systems. If so, use the local package manager or port system to build that module. On FreeBSD:

```
cd /usr/ports/databases/py-sqlite3
sudo make install clean
```

4.3 Binary Log Data

Binary data in your logs will be converted to escape sequences or ?'s depending on the encoding settings to prevent decoding exceptions from crashing Beaver.

> http://docs.python.org/2/library/codecs.html#codecs.replace_errors
>
> malformed data is replaced with a suitable replacement character such as '?' in bytestrings and 'ufffd' in Unicode strings.

CHAPTER 5

Credits

Based on work from Giampaolo and Lusis:

```
Real time log files watcher supporting log rotation.

Original Author: Giampaolo Rodola' <g.rodola [AT] gmail [DOT] com>
http://code.activestate.com/recipes/577968-log-watcher-tail-f-log/

License: MIT

Other hacks (ZMQ, JSON, optparse, ...): lusis
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search