yelp-clog Documentation

Release 2.16.0

Yelp Infra Team

Mar 28, 2018

Contents

clog is a package for handling log data. It can be used for the following:

CHAPTER 1

Python Logging Handler

clog.handlers.ScribeHandler can be used to send standard python `logging` to a scribe stream.

CHAPTER 2

Logging Operational and Mission Critical Data

clog.loggers.ScribeLogger.log_line() can be used to log mission critical, machine readable, and opertional data to scribe. There is also a global `clog.log_line()` which has the same purpose but requires global configuration (see *clog.config*). Use of the global is discouraged.

CHAPTER 3

Reading Scribe Logs

clog.readers provides classes for reading scribe logs locally or from a server.

CHAPTER 4

Contents

4.1 Release Notes

4.1.1 2.16.0

- Remove backend_map logic from ScribeLogger and MonkLogger.
- When Monk is enabled, use ScribeMonkLogger instead of using MonkLogger directly.
- Implement preferred_backend logic in ScribeMonkLogger. See preferred_backend and preferred_backend_map in config.py.
- Upgrade Monk client library to 0.4.1.
- Implement memory buffering for MonkLogger.

4.1.2 2.15.0

- Use scribify function to convert MonkLogger stream names, for backward compatibility with ScribeLogger.

4.1.3 2.14.0

- Add close() method to MonkLogger

4.1.4 2.13.0

- When creating a ScribeTailer instance, fall back to find_tail_host() if no tail service host is specified and the configuration is not set.
- Implement size limit for MonkLogger (5MB)

4.1.5 2.12.1

- Fix a compatiblity issue where gzipped logs weren't decompresser correctly in python 3.

4.1.6 2.12.0

- Change reader configuration file path (from /etc/yelp_clog.json to /nail/srv/configs/yelp_clog.json)

4.1.7 2.11.0

- Prevent exceptions if Monk is enabled but not installed

4.1.8 2.6.2

- Remove references to yelp_lib

4.1.9 2.6.1

- Add type checking to `MockLogger` log_line function

4.1.10 2.6.0

- Make StreamTailer connection message more flexible
- Drop py33 and add py35

4.1.11 2.5.2

- Fix FileLogger when using python3

4.1.12 2.5.1

- Fix use of unicode

4.1.13 2.5.0

- Use six instead of future

4.1.14 2.4.4

- Fix `ScribeHandler` under Python 3.

4.1.15 2.4.3

- Fix tail/nc process leaks during testing.

4.1.16 2.4.2

- Improvements to testing.
- Fix `scribify` function under Python 3.

4.1.17 2.4.1

- Remove the thriftpy hard-pinning. Compatible now with thriftpy 0.1+ and 0.2+.

4.1.18 2.4.0

- Add clog_enable_stdout_logging config option to dump log lines to stdout. Defaults to false.

4.1.19 1.4.0

- Switched to thriftpy, a third-party Thrift implementation.
- Now compatible with Python 3.3+ and PyPy2.

4.1.20 1.3.0

- Add an enforcement on the size of scribe log lines. Log lines are expected to be less than 5MB; they are processed as normal. For log lines with size between 5MB and 50MB, warnings are issued by logging them additionaly to the scribe log "tmp_who_clog_large_line". Any line over 50 MB is treated as an error, and the exception LogLineIsTooLongError is raised.

4.1.21 1.2.0

- The logging_timeout argument of loggers.ScribeLogger is added. This allows scribe logging to timeout instead of being blocked if the scribe server is non-responding promptly; this benefits the applications prioritizing user experience over logging.

4.1.22 1.1.4

- The _recv_compressed method of StreamTailer and the python-snappy dependency have been removed. Now it's no more possible to automatically decompress a snappy-compressed stream.

4.1.23 1.0.0

This is a major release as it breaks backwards compatibility.

- Many imports were removed from the top level *clog* namespace. They are still available from the full module name (ex: *clog.loggers*, *clog.config*, etc)
- Configuration defaults have changed. The default is now to log to a local file in */tmp*, instead of raising a ValueError if scribe is not configured. Note this only applies to `clog.log_line()`

4.2 Configuration

Configuration for *clog.global_state*. The following configuration settings are supported:

scribe_host (string) hostname of a scribe service used by `clog.log_line()` to write logs to scribe

scribe_port (int) port of the abovementioned scribe service

scribe_retry_interval number of seconds to wait before retrying a connection to scribe. Used by `clog.log_line()` (default 10)

log_dir directory used to store files when clog_enable_file_logging is enabled. Defaults to the value of $TMPDIR, or */tmp* if unset

scribe_disable disable writing any logs to scribe (default True)

scribe_errors_to_syslog flag to enable sending errors to syslog, otherwise to stderr (default False)

scribe_logging_timeout number of milliseconds to wait on socket connection to scribe server. This prevents from being blocked forever on, e.g., writing to scribe server. If a write times out, the delivery of the log line is not guaranteed and the status of the delivery is unknown; it can be either succeeded or failed. For backward compatibility, the old default blocking behavior is on when this parameter is left unset or set to 0; both the values None and 0 are treated as "infinite".

clog_enable_file_logging flag to enable logging to local files. (Default False)

clog_enable_stdout_logging flag to enable logging to stdout. Each log line is prefixed with the stream name. (Default False)

localS3 If True, will fetch s3 files directly rather than talking to a service.

`clog.config.`**`configure`**(*scribe_host*, *scribe_port*, ***kwargs*)
Configure the *clog* package from arguments.

Parameters

- **`scribe_host`** – the scribe service hostname
- **`scribe_port`** – the scribe service port
- **`kwargs`** – other configuration parameters

`clog.config.`**`configure_from_dict`**(*config_dict*)
Configure the *clog* package from a dictionary.

Parameters **`config_dict`** – a dict of config data

`clog.config.`**`configure_from_object`**(*config_obj*)
Configure the *clog* package from an object (or module).

Parameters **`config_obj`** – an object or module with config attributes

4.3 Global Logger

Log lines to scribe using the default global logger.

`exception` `clog.global_state.`**`LoggingNotConfiguredError`**

`clog.global_state.`**`check_create_default_loggers`**`()`
Set up global loggers, if necessary.

clog.global_state.**create_preferred_backend_map**()
PyStaticConfig doesn't support having a map in the configuration, so we represent a map as a list, and we use this function to generate an actual python dictionary from it.

clog.global_state.**log_line**(*stream*, *line*)
Log a single line to the global logger(s). If the line contains any newline characters each line will be logged as a separate message. If this is a problem for your log you should encode your log messages.

Parameters

- **stream** – name of the scribe stream to send this log
- **line** – contents of the log message

clog.global_state.**reset_default_loggers**()
Destroy the global *clog* loggers. This must be done when forking to ensure that children do not share a desynchronized connection to Scribe

Any writes *after* this call will cause the loggers to be rebuilt, so this must be the last thing done before the fork or, better yet, the first thing after the fork.

4.4 Handlers

logging.Handler objects which can be used to send standard python logging to a scribe stream.

class clog.handlers.**ScribeHandler**(*host*, *port*, *stream*, *retry_interval=0*)
Handler for sending python standard logging messages to a scribe stream.

```
import clog.handlers, logging
log = logging.getLogger(name)
log.addHandler(clog.handlers.ScribeHandler('localhost', 3600, 'stream', retry_
↪interval=3))
```

Parameters

- **host** – hostname of scribe server
- **port** – port number of scribe server
- **stream** – name of the scribe stream logs will be sent to
- **retry_interval** – default 0, number of seconds to wait between retries

emit(*record*)
Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

class clog.handlers.**CLogHandler**(*stream*, *logger=None*)
Deprecated since version 0.1.6.

> **Warning:** Use ScribeHandler if you want to log to scribe, or a logging.handlers.FileHandler to log to a local file.

Handler for the standard logging library that logs to clog.

emit(*record*)
Do whatever it takes to actually log the specified logging record.

This version is intended to be implemented by subclasses and so raises a NotImplementedError.

clog.handlers.**add_logger_to_scribe**(*logger*, *log_level=20*, *fmt='%(process)s\t%(asctime)s\t%(name)-12s %(levelname)-8s: %(message)s'*, *clogger_object=None*)
Sets up a logger to log to scribe.

By default, messages at the INFO level and higher will go to scribe.

Deprecated since version 0.1.6.

> **Warning:** This function is deprecated in favor of using `clog.log_line()` or *ScribeHandler* directly.

Parameters

- **logger** – A logging.Logger instance
- **log_level** – The level to log at
- **clogger_object** – for use in testing

clog.handlers.**get_scribed_logger**(*log_name*, **args*, ***kwargs*)
Get/create a logger and adds it to scribe.

Deprecated since version 0.1.6.

> **Warning:** This function is deprecated in favor of using `clog.log_line()` directly.

Parameters

- **log_name** – name of log to write to using logging.getLogger
- **kwargs** (*args,*) – passed to add_logger_to_scribe

Returns a `logging.Logger`

4.5 Loggers

Loggers which implement *log_line(stream, data)* which are used to send log data to a stream.

class clog.loggers.**FileLogger**
Implementation that logs to local files under a directory

class clog.loggers.**GZipFileLogger**(*day=None*)
Implementation of a logger that logs to local gzipped files.

exception clog.loggers.**LogLineIsTooLongError**

class clog.loggers.**MockLogger**
Mock implementation for testing

class clog.loggers.**MonkLogger**(*client_id*, *host=None*, *port=None*)
Wrapper around MonkProducer

exception clog.loggers.**ScribeIsNotForkSafeError**

class clog.loggers.**ScribeLogger**(*host*, *port*, *retry_interval*, *report_status=None*, *logging_timeout=None*)

Implementation that logs to a scribe server. If errors are encountered, drop lines and retry occasionally.

Parameters

- **host** – hostname of the scribe server
- **port** – port number of the scribe server
- **retry_interval** – number of seconds to wait between retries
- **report_status** – a function *report_status(is_error, msg)* which is called to print out errors and status messages. The first argument indicates whether what is being printed is an error or not, and the second argument is the actual message.
- **logging_timeout** – milliseconds to time out scribe logging; "0" means blocking (no timeout)

log_line(*stream*, *line*)

Log a single line. It should not include any newline characters. If the line size is over 50 MB, an exception raises and the line will be dropped. If the line size is over 5 MB, a message consisting origin stream information will be recorded at WHO_CLOG_LARGE_LINE_STREAM (in json format).

class clog.loggers.**ScribeMonkLogger**(*config*, *scribe_logger*, *monk_logger*, *preferred_backend_map={}*)

The ScribeMonkLogger is a wrapper around both the ScribeLogger and the MonkLogger. The actuall logger being used will depend on the preferred_backend and preferred_backend_map.

class clog.loggers.**StdoutLogger**

Implementation that logs to stdout with stream name as a prefix.

clog.loggers.**get_default_reporter**(*use_syslog=None*)

Returns the default reporter based on the value of the argument

Parameters **report_to_syslog** – Whether to use syslog or stderr. Defaults to the value of *config.scribe_errors_to_syslog*

clog.loggers.**report_to_syslog**(*is_error*, *msg*)

Report errors into Syslog.

Use Syslog with UDP in order not to be forced to write to a unix socket, useful if yelp-clog is run inside docker containers with Syslog running on the actual host.

Syslogs prepends 'ident: ' to messages with an identifier. This function uses *clog* as identifier.

4.6 Scribe Readers

Classes which read log data from scribe.

class clog.readers.**CLogStreamIterator**(*stream_reader*, *line_num=-1*, *current_chunk=None*, *chunk_line_num=-1*)

Iterator used by `ClogStreamReader` for iterating over lines of chunks of a stream.

class clog.readers.**CLogStreamReader**(*stream_name*, *stream_dir*, *date*, *fail_on_missing=False*)

Make a stream reader for a day of *clog* entries

clog entries are stored by stream name and date and broken into separate chunks which may or may not be compressed with gzip or bzip or be plaintext.

For instance, the entries for a stream called 'foo' on New Years Day 2009 will be laid out in the file system like

STREAM_DIR/foo/foo-2009-01-01_00000.gz
STREAM_DIR/foo/foo-2009-01-01_00001.gz
...

Example usage:

```
reader = CLogStreamReader('stream_name', '/path/to/logs', date.today())
for line in reader:
    print line
```

Parameters

- **stream_dir** – the stream directory like */storage/coraid5/scribe_logs*
- **stream_name** – the stream name like *biz_views*
- **date** – the date of the logs
- **fail_on_missing** – Fail if there are no log files for the specified stream and date

chunk_filenames()
Get an iterator for all the chunk filenames

class clog.readers.**NetCLogStreamReader**(*bufsize=1024*, *host=None*, *port=None*, *automagic_recovery=False*, *localS3=False*)

Read logs from a scribe server

Note: This reader will stream logs from the source, it is not recommended for large logs. Use a mrjob instead.

Example usage:

```
stream_reader = NetCLogStreamReader()
with stream_reader.read_date_range(
        'ranger',
        date(2010, 1, 1),
        date(2010,12,31)
) as reader:
    for line in reader:
        print line
```

Configuration:

This class can be configured either by passing a *host* and *port* to the constructor, or by using `staticconf` to with the following settings

scribe_net_reader.host hostname of the scribe server used to stream scribe logs

scribe_net_reader.port port of the scribe server used to stream scribe logs

Parameters

- **bufsize** – How many bytes to buffer internally
- **host** – The host to connect to (defaults to scribe_net_reader.host)

- **port** – The port to connect to (defaults to scribe_net_reader.port)
- **automagic_recovery** – Whether to tail the stream, continuously retrying the connection (defaults to False)

list_streams()
Get a context manager to use for reading list names

read_date_range(*stream_name*, *start_date*, *end_date*)
Get a context manager to use for reading a stream for a date range

exception clog.readers.**NoLogDataError**

class clog.readers.**StreamTailer**(*stream*, *host=None*, *port=None*, *bufsize=4096*, *automagic_recovery=True*, *add_newlines=True*, *raise_on_start=True*, *timeout=None*, *reconnect_callback=None*, *use_kafka=False*, *lines=None*, *protocol_opts=None*)
Tail a Scribe stream from a tailing server

Example Usage:

```
tailer = StreamTailer('stream_name', host, port)
for line in tailer:
    print line
```

Configuration:

This class can be configured by passing a host and port to the constructor or by using `staticconf` with the following setting:

scribe_tail_services (list of dicts {'host': host, 'port': port}) list of host and port addresses of scribe endpoints for tailing logs in real time.

Parameters

- **stream** (*string*) – the name of the string like 'ranger'
- **host** (*string*) – the host name
- **port** – the port to connect to
- **bufsize** – the number of bytes to buffer
- **automagic_recovery** (*bool*) – continue to retry connection forever
- **add_newline** – add newlines to the items yielded in the iter
- **raise_on_start** (*bool*) – raise an error if you get a disconnect immediately after starting (otherwise, returns silently), Default True
- **timeout** (*int*) – connection timeout
- **reconnect_callback** (*function*) – callback called when reconnecting
- **protocol_opts** (*dict*) – optional protocol parameters

list_streams()
Get a context manager to use for reading list names

exception clog.readers.**StreamTailerSetupError**(*host*, *port*, *message*)

clog.readers.**construct_conn_msg**(*stream*, *lines=None*, *protocol_opts=None*)
Return a connnection message

Parameters

- **stream** – stream name
- **lines** – number of messages to consume
- **protocol_opts** – optional arguments

clog.readers.**get_s3_info**(*hostname*, *stream_name=None*)

Returns (s3_host, s3_bucket(s), s3_prefix)

If no stream name is provided (i.e. None), both normal and tmp buckets are returned as a dict.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index