## Contents

1 Contents 3
   1.1 Quickstart ................................................................. 3
   1.2 Analog API .................................................................. 4
   1.3 Changelog .................................................................. 13

2 Authors  17
   2.1 License ................................................................. 17

3 Indices and tables  19

Python Module Index  21
Analog is a weblog analysis utility that provides these metrics:

- Number for requests.
- Response request method (HTTP verb) distribution.
- Response status code distribution.
- Requests per path.
- Response time statistics (mean, median).
- Response upstream time statistics (mean, median).
- Response body size in bytes statistics (mean, median).
- Per path request method (HTTP verb) distribution.
- Per path response status code distribution.
- Per path response time statistics (mean, median).
- Per path response upstream time statistics (mean, median).
- Per path response body size in bytes statistics (mean, median).

Code and issues are on github.com/fabianbuechler/analog. Please also post feature requests there.

Analog can be installed from PyPI at pypi.python.org/pypi/analog:

```
$ pip install analog
```
1.1 Quickstart

Use the analog CLI to start the analysis:

```
$ analog nginx /var/log/nginx/mysite.access.log
```

This invokes the analyzer with a predefined Nginx log format and will by default parse the complete logfile for all different request paths and analyze all different request methods (e.g. GET, PUT) and response status codes (e.g. 200, 401, 404, 409, 500). The report would be printed to standard out as a simple list. Use normal piping to save the report output in a file.

For details on the analog command see analog.main.main()

1.1.1 Options

analog has these options:

- **format** Log format identifier. Currently only nginx is predefined. Choose custom to define a custom log entry pattern via --pattern-regex and --time-format.

- **-v / --version** Print analog version and exit.

- **-h / --help** Print manual and exit.

Each format subcommand has the following options:

- **-o / --output** Output format. Defaults to plaintext list output. Choose from table, grid, csv and tsv for tabular formats. For details see the available report renderers

- **-p / --path** Path(s) to monitor. If not provided, all distinct paths will be analyzed. Groups paths by matching the beginning of the log entry values.

- **-v / --verb** HTTP verbs(s) to monitor. If not provided, by default DELETE, GET, PATCH, POST and PUT will be analyzed.
-s/--status  Response status code(s) to monitor. If not provided, by default 1, 2, 3, 4 and 5 are analyzed. Groups paths by matching the beginning of the log entry values.

-a/--max-age  Limit the maximum age of log entries to analyze in minutes. Useful for continuous analysis of the same logfile (e.g. the last ten minutes every ten minutes).

-ps/--path-stats  Include per-path statistics in the analysis report output. By default analog only generates overall statistics.

-t/--timing  Tracks and prints analysis time.

When choosing the custom log format, these options are available additionally:

-pr/--pattern-regex  Regular expression log format pattern. Define named groups for all attributes to match. Required attributes are: timestamp, verb, path, status, body_bytes_sent, request_time, upstream_response_time. See log formats for details.

-tf/--time-format  Log entry timestamp format definition (strftime compatible).

Options from File

To specify the options via a file, call analog like this:

```
$ analog @arguments.txt logfile.log
```

The arguments.txt (can have any name) contains one argument per line. Arguments and their values can also be comma- or whitespace-separated on one line. For example:

```
nginx
-o    table
--verb GET, POST, PUT
--verb PATCH
--status 404, 500
--path /foo/bar
--path /baz
--path-stats
-t
```


## 1.2 Analog API

### 1.2.1 analog Command

The primary way to invoke analog is via the analog command which calls analog.main.main().

```
analog.main.main(argv=\nNone)
analog - Log Analysis Utility.
```

Name the logfile to analyze (positional argument) or leave it out to read from stdin. This can be handy for piping in filtered logfiles (e.g. with grep).

Select the logfile format subcommand that suits your needs or define a custom log format using analog custom --pattern-regex <...> --time-format <...>.

To analyze for the logfile for specified paths, provide them via --path arguments (multiple times). Also, monitoring specific HTTP verbs (request methods) via --verb and specific response status codes via --status argument(s) is possible.
Paths and status codes all match the start of the actual log entry values. Thus, specifying a path `/foo` will group all paths beginning with that value.

Arguments can be listed in a file by specifying @argument_file.txt as parameter.

### 1.2.2 Analyzer

The **Analyzer** is the main logfile parser class. It uses a `analog.formats.LogFormat` instance to parse the log entries and passes them on to a `analog.report.Report` instance for statistical analysis. The report itself can be passed through a `analog.renderers.Renderer` subclass for different report output formats.

```python
class analog.analyzer.Analyzer(log, format, pattern=None, time_format=None, verbs=['DELETE', 'GET', 'PATCH', 'POST', 'PUT'], status_codes=[1, 2, 3, 4, 5], paths=[], max_age=None, path_stats=False):

Log analysis utility.

Scan a logfile for logged requests and analyze calculate statistical metrics in a `analog.report.Report`

```python
__call__()

Analyze defined logfile.

Returns: log analysis report object.

Return type: analog.report.Report

```python
__init__(log, format, pattern=None, time_format=None, verbs=['DELETE', 'GET', 'PATCH', 'POST', 'PUT'], status_codes=[1, 2, 3, 4, 5], paths=[], max_age=None, path_stats=False)

Configure log analyzer.

Parameters:

- `log`: handle on logfile to read and analyze.
- `format`: log format identifier or ‘custom’.
- `pattern`: custom log format pattern expression.
- `time_format`: log entry timestamp format (strftime compatible).
- `verbs`: HTTP verbs to be tracked. Defaults to `analog.analyzer.DEFAULT_VERBS`.
- `status_codes`: status_codes to be tracked. May be prefixes, e.g. `['100', '2', '3', '4', '404']`. Defaults to `analog.analyzer.DEFAULT_STATUS_CODES`.
- `paths`: Paths to explicitly analyze. If not defined, paths are detected automatically. Defaults to `analog.analyzer.DEFAULT_PATHS`.
- `max_age`: Max. age of log entries to analyze in minutes. Unlimited by default.

Raises: analog.exceptions.MissingFormatError if no format is specified.

**analyzer** is a convenience wrapper around `analog.analyzer.Analyzer` and can act as the main and only required entry point when using analog from code.

```python
analog.analyzer.analyze(log, format, pattern=None, time_format=None, verbs=['DELETE', 'GET', 'PATCH', 'POST', 'PUT'], status_codes=[1, 2, 3, 4, 5], paths=[], max_age=None, path_stats=False, timing=False, output_format=None)

Convenience wrapper around analog.analyzer.Analyzer.

Parameters:

- `log`: handle on logfile to read and analyze.
- `format`: log format identifier or ‘custom’.

1.2. Analog API
• **pattern** *(str)* – custom log format pattern expression.

• **time_format** *(str)* – log entry timestamp format (strftime compatible).

• **verbs** *(list)* – HTTP verbs to be tracked. Defaults to `analog.analyzer.DEFAULT_VERBS`.

• **status_codes** *(list)* – status_codes to be tracked. May be prefixes, e.g. ["100", "2", "3", "4", "404"] . Defaults to `analog.analyzer.DEFAULT_STATUS_CODES`.

• **paths** *(list of str)* – Paths to explicitly analyze. If not defined, paths are detected automatically. Defaults to `analog.analyzer.DEFAULT_PATHS`.

• **max_age** *(int)* – Max. age of log entries to analyze in minutes. Unlimited by default.

• **path_stats** *(bool)* – Print per-path analysis report. Default off.

• **timing** *(bool)* – print analysis timing information?

• **output_format** *(str)* – report output format.

Returns log analysis report object.

Return type `analog.report.Report`

```
analog.analyzer.DEFAULT_VERBS = ['DELETE', 'GET', 'PATCH', 'POST', 'PUT']
Default verbs to monitor if unconfigured.
analog.analyzer.DEFAULT_STATUS_CODES = [1, 2, 3, 4, 5]
Default status codes to monitor if unconfigured.
analog.analyzer.DEFAULT_PATHS = []
Default paths (all) to monitor if unconfigured.
```

### 1.2.3 Log Format

A LogFormat defines how log entries are represented in and can be parsed from a log file.

```
class analog.formats.LogFormat (name, pattern, time_format)

Log format definition.

Represents log format recognition patterns by name.

A name:format mapping of all defined log format patterns can be retrieved using `analog.formats.LogFormat.all_formats()`.

Each log format should at least define the following match groups:

• **timestamp**: Local time.

• **verb**: HTTP verb (GET, POST, PUT,...).

• **path**: Request path.

• **status**: Response status code.

• **body_bytes_sent**: Body size in bytes.

• **request_time**: Request time.

• **upstream_response_time**: Upstream response time.

__init__ (name, pattern, time_format)

Describe log format.
The format pattern is a (verbose) regex pattern string specifying the log entry attributes as named groups that is compiled into a re.Pattern object.

All pattern group names are be available as attributes of log entries when using a analog.formats.LogEntry.entry().

Parameters

- **name** (str) – log format name.
- **pattern** (raw str) – regular expression pattern string.
- **time_format** (str) – timestamp parsing pattern.

Raises analog.exceptions.InvalidFormatExpressionError if missing required format pattern groups or the pattern is not a valid regular expression.

**classmethod all_formats()**

Mapping of all defined log format patterns.

Returns dictionary of name:LogFormat instances.

Return type dict

**entry(match)**

Convert regex match object to log entry object.

Parameters match (re.MatchObject) – regex match object from pattern match.

Returns log entry object with all pattern keys as attributes.

Return type collections.namedtuple

Predefined Formats

nginx

analog.formats.NGINX = <analog.formats.LogFormat object>

Nginx combined timed format:

```
'$remote_addr - $remote_user [$time_local] "$request";
'$status $body_bytes_sent "$http_referer";
'"$http_user_agent" "$http_x_forwarded_for";
'$request_time $upstream_response_time $pipe';
```

1.2.4 Reports

A Report collects log entry information and computes the statistical analysis.

**class analog.report.Report(verbs, status_codes)**

Log analysis report object.

Provides these statistical metrics:

- Number for requests.
- Response request method (HTTP verb) distribution.
- Response status code distribution.
- Requests per path.
- Response time statistics (mean, median).
- Response upstream time statistics (mean, median).
- Response body size in bytes statistics (mean, median).
- Per path request method (HTTP verb) distribution.
- Per path response status code distribution.
- Per path response time statistics (mean, median).
- Per path response upstream time statistics (mean, median).
- Per path response body size in bytes statistics (mean, median).

__init__(verbs, status_codes)

Create new log report object.

Use add() method to add log entries to be analyzed.

Parameters

- **verbs** (list) – HTTP verbs to be tracked.
- **status_codes** (list) – status_codes to be tracked. May be prefixes, e.g. ["100", "2", "3", "4", "404"]

Returns Report analysis object

Return type `analog.report.Report`

add(path, verb, status, time, upstream_time, body_bytes)

Add a log entry to the report.

Any request with verb not matching any of self._verbs or status not matching any of self._status is ignored.

Parameters

- **path** (str) – monitored request path.
- **verb** (str) – HTTP method (GET, POST, ...)
- **status** (int) – response status code.
- **time** (float) – response time in seconds.
- **upstream_time** (float) – upstream response time in seconds.
- **body_bytes** (float) – response body size in bytes.

Returns response body size statistics.

Return type `analog.report.ListStats`

finish()

Stop execution timer.

path_body_bytes

Response body size in bytes of all matched requests per path.

Returns path mapping of body size statistics.

Return type `dict of analog.report.ListStats`

path_requests

List paths of all matched requests, ordered by frequency.
Returns tuples of path and occurrence count.

Return type  list of tuple

path_status
List status codes of all matched requests per path.

Status codes are grouped by path and ordered by frequency.

Returns path mapping of tuples of status code and occurrence count.

Return type  dict of list of tuple

path_times
Response time statistics of all matched requests per path.

Returns path mapping of response time statistics.

Return type  dict of analog.report.ListStats

path_upstream_times
Response upstream time statistics of all matched requests per path.

Returns path mapping of response upstream time statistics.

Return type  dict of analog.report.ListStats

path_verbs
List request methods (HTTP verbs) of all matched requests per path.

Verbs are grouped by path and ordered by frequency.

Returns path mapping of tuples of verb and occurrence count.

Return type  dict of list of tuple

render(path_stats, output_format)
Render report data into output_format.

Parameters

• path_stats (bool) – include per path statistics in output.
• output_format (str) – name of report renderer.

Raises analog.exceptions.UnknownRendererError or unknown output_format identifiers.

Returns rendered report data.

Return type  str

status
List status codes of all matched requests, ordered by frequency.

Returns tuples of status code and occurrence count.

Return type  list of tuple

times
Response time statistics of all matched requests.

Returns response time statistics.

Return type  analog.report.ListStats

upstream_times
Response upstream time statistics of all matched requests.
Returns response upstream time statistics.

Return type `analog.report.ListStats`

verbs
List request methods of all matched requests, ordered by frequency.

Returns tuples of HTTP verb and occurrence count.

Return type `list of tuple`

class `analog.report.ListStats(elements)`
Statistic analysis of a list of values.

__init__ `elements`
Calculate some stats from list of values.

Parameters `elements` (list) – list of values.

1.2.5 Renderers

Reports are rendered using one of the available renderers. These all implement the basic `analog.renderers.Renderer` interface.

class `analog.renderers.Renderer`
Base report renderer interface.

classmethod `all_renderers()`
Get a mapping of all defined report renderer names.

Returns dictionary of name to renderer class.

Return type `dict`

classmethod `by_name(name)`
Select specific Renderer subclass by name.

Parameters `name` (str) – name of subclass.

Returns Renderer subclass instance.

Return type `analog.renderers.Renderer`


render `report, path_stats=False`
Render report statistics.

Parameters


- `path_stats` (bool) – include path statistics in output.

Returns output string

Return type `str`
Available Renderers

default

class analog.renderers.PlainTextRenderer
    Default renderer for plain text output in list format.

Tabular Data

class analog.renderers.TabularDataRenderer
    Base renderer for report output in any tabular form.

    _list_stats (list_stats)
        Get list of (key,value) tuples for each attribute of list_stats.

        Parameters list_stats (analog.report.ListStats) – list statistics object.

        Returns (key, value) tuples for each ListStats attribute.

        Return type list of tuple

    _tabular_data (report, path_stats)
        Prepare tabular data for output.

        Generate a list of header fields, a list of total values for each field and a list of the same values per path.

        Parameters


            • path_stats (bool) – include per path statistics in output.

        Returns tuple of table (headers, rows).

        Return type tuple

Visual Tables

class analog.renderers.ASCIITableRenderer
    Base renderer for report output in ascii-table format.

table

class analog.renderers.SimpleTableRenderer
    Renderer for tabular report output in simple reSt table format.

grid

class analog.renderers.GridTableRenderer
    Renderer for tabular report output in grid table format.

Separated Values

class analog.renderers.SeparatedValuesRenderer
    Base renderer for report output in delimiter-separated values format.

csv

class analog.renderers.CSVRenderer
    Renderer for report output in comma separated values format.
tsv

```py
class analog.renderers.TSVRenderer
    Renderer for report output in tab separated values format.
```

## 1.2.6 Utils

```py
class analog.utils.AnalogArgumentParser (prog=None, usage=None, description=None, epilog=None, parents=[], formatter_class=argparse.HelpFormatter, prefix_chars='-', fromfile_prefix_chars=None, argument_default=None, conflict_handler='error', add_help=True, allow_abbrev=True)
```

ArgumentParser that reads multiple values per argument from files.

Arguments read from files may contain comma or whitespace separated values.

To read arguments from files create a parser with `fromfile_prefix_chars` set:

```py
parser = AnalogArgumentParser(fromfile_prefix_chars='@')
```

Then this parser can be called with argument files:

```py
parser.parse_args(['--arg1', '@args_file', 'more-args'])
```

The argument files contain one argument per line. Arguments can be comma or whitespace separated on a line. For example all of this works:

```bash
nginx
-o table
--verb GET, POST, PUT
--verb PATCH
--status 404, 500
--path /foo/bar
--path /baz
--path-stats
-t positional
arg
```

```py
convert_arg_line_to_args (arg_line)
    Comma/whitespace-split arg_line and yield separate attributes.
    Argument names defined at the beginning of a line (-a, --arg) are repeated for each argument value in arg_line.
    Parameters arg_line (str) – one line of argument(s) read from a file
    Returns argument generator
    Return type generator
```

```py
class analog.utils.PrefixMatchingCounter (*args, **kwds)
    Counter-like object that increments a field if it has a common prefix.
    Example: “400”, “401”, “404” all increment a field named “4”.
```
1.2.7 Exceptions

Analog exceptions.

```
excep\n\nexception analog.exceptions.AnalogError
    Exception base class for all Analog errors.
```

```
excep\n\nexception analog.exceptions.InvalidFormatExpressionError
    Error raised for invalid format regex patterns.
```

```
excep\n\nexception analog.exceptions.MissingFormatError
    Error raised when Analyzer is called without format.
```

```
excep\n\nexception analog.exceptions.UnknownRendererError
    Error raised for unknown output format names (to select renderer).
```

1.3 Changelog

1.3.1 1.0.0 - 2015-02-26

- Provide yaml config file for Travis-CI.
- Extend tox environments to cover 2.7, 3.2, 3.3, 3.4, pypy and pypy3.
- Convert repository to git and move to github.
- Set version only in setup.py, use via pkg_resources.get_distribution.

1.3.2 1.0.0b1 - 2014-04-06

- Going beta with Python 3.4 support and good test coverage.

1.3.3 0.3.4 - 2014-04-01

- Test analog.analyzer implementation.
- Test analog.util implementation.

1.3.4 0.3.3 - 2014-03-10

- Test analog.renderers implementation.
- Fix bug in default plaintext renderer.

1.3.5 0.3.2 - 2014-03-02


1.3.6 0.3.1 - 2014-02-09

- Rename --max_age option to --max-age for consistency.
1.3.7 0.3.0 - 2014-02-09

• Ignore __init__.py at PEP257 checks since __all__ is not properly supported.
• Fix custom log format definitions. Format selection in CLI via subcommands.
• Add pypy to tox environments.

1.3.8 0.2.0 - 2014-01-30

• Remove dependency on configparser package for Python 2.x.
• Allow specifying all analog arguments in a file for convenience.

1.3.9 0.1.7 - 2014-01-27

• Giving up on VERSIONS file. Does not work with different distributions.

1.3.10 0.1.6 - 2014-01-27

• Include CHANGELOG in documentation.
• Move VERSION file to analog module to make sure it can be installed.

1.3.11 0.1.5 - 2014-01-27

• Replace numpy with backport of statistics for mean and median calculation.

1.3.12 0.1.4 - 2014-01-27

• Move fallback for verbs, status_codes and paths configuration to analyzer. Also use the fallbacks in analog.analyzer.Analyzer.__init__ and analog.analyzer.analyze.

1.3.13 0.1.3 - 2014-01-27

• Fix API-docs building on readthedocs.

1.3.14 0.1.1 - 2014-01-26

• Add numpy to requirements.txt since installation via setup.py install does not work.
• Strip VERSION when reading it in setup.py.
1.3.15 0.1.0 - 2014-01-26

- Start documentation: quickstart and CLI usage plus API documentation.
- Add renderers for CSV and TSV output. Use `--output [csv|tsv]`. Unified codebase for all tabular renderers.
- Also analyze HTTP verbs distribution for overall report.
- Remove timezone aware datetime handling for the moment.
- Introduce `Report.add` method to not expose `Report` externals to `Analyzer`.
- Install `pytz` on Python <= 3.2 for UTC object. Else use `datetime.timezone`.
- Add tox environment for py2.7 and py3.3 testing.
- Initial implementation of log analyzer and report object.
- Initial package structure, docs, requirements, test scripts.
CHAPTER 2

Authors

Analog has been built by and with the help of:

• Fabian Büchler <fabian.buechler@gmail.com>
• Joris Bayer <jjbayer@gmail.com>

2.1 License

The MIT License (MIT)

Copyright (c) 2014 Fabian Büchler <fabian.buechler@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
CHAPTER 3

Indices and tables

- genindex
- modindex
a

analog.exceptions, 13
Symbols
__call__() (analog.analyzer.Analyzer method), 5
__init__() (analog.analyzer.Analyzer method), 5
__init__() (analog.formats.LogFormat method), 6
__init__() (analog.report.ListStats method), 10
__init__() (analog.report.Report method), 8
_list_stats() (analog.renderers.TabularDataRenderer method), 11
_tabular_data() (analog.renderers.TabularDataRenderer method), 11

A
add() (analog.report.Report method), 8
all_formats() (analog.formats.LogFormat class method), 7
all_renderers() (analog.renderers.Renderer class method), 10
analog.exceptions (module), 13
AnalogArgumentParser (class in analog.utils), 12
AnalogError, 13
analyze() (in module analog.analyzer), 5
Analyzer (class in analog.analyzer), 5
ASCIITableRenderer (class in analog.renderers), 11

B
body_bytes (analog.report.Report attribute), 8
by_name() (analog.renderers.Renderer class method), 10

C
convert_arg_line_to_args() (analog.utils.AnalogArgumentParser method), 12
CSVRenderer (class in analog.renderers), 11

D
DEFAULT_PATHS (in module analog.analyzer), 6
DEFAULT_STATUS_CODES (in module analog.analyzer), 6
DEFAULT_VERBS (in module analog.analyzer), 6

E
e entry() (analog.formats.LogFormat method), 7

F
finish() (analog.report.Report method), 8

G
GridTableRenderer (class in analog.renderers), 11
InvalidFormatExpressionError, 13

I
ListStats (class in analog.report), 10

L
ListStats (class in analog.report), 10
LogFormat (class in analog.formats), 6

M
main() (in module analog.main), 4
MissingFormatError, 13

N
NGINX (in module analog.formats), 7

P
path_body_bytes (analog.report.Report attribute), 8
path_requests (analog.report.Report attribute), 8
path_status (analog.report.Report attribute), 9
path_times (analog.report.Report attribute), 9
path_upstream_times (analog.report.Report attribute), 9
path_verbs (analog.report.Report attribute), 9
PlainTextRenderer (class in analog.renderers), 11
PrefixMatchingCounter (class in analog.utils), 12

R
render() (analog.renderers.Renderer method), 10
render() (analog.report.Report method), 9
Renderer (class in analog.renderers), 10
Report (class in analog.report), 7
SeparatedValuesRenderer (class in analog.renderers), 11
SimpleTableRenderer (class in analog.renderers), 11
status (analog.report.Report attribute), 9
TabularDataRenderer (class in analog.renderers), 11
times (analog.report.Report attribute), 9
TSVRenderer (class in analog.renderers), 12
UnknownRendererError, 13
upstream_times (analog.report.Report attribute), 9
verbs (analog.report.Report attribute), 10