title: Logagent Configuration File description: YAML config files for Logagent, light-weight log shipper with out of the box and extensible log parsing, on-disk buffering, secure transport, bulk indexing to Elasticsearch and Sematext logs management platform. Define log parser pattern loading and definitions and ship your logs to Elasticsearch or Sematext Cloud for real-time log view.

Logagent is flexible. You can choose to run it as a system service and use the default configuration file, or pass the CLI tool a `--config custom.yml` flag with a custom configuration.

Default Configuration File Location

When Logagent is installed as a system service, by running the `logagent-setup` command, the default config file is located in:

`/etc/sematext/logagent.conf`

Default Configuration File Structure

```
# /etc/sematext/logagent.conf

# Global options
options:
  # print stats every 60 seconds
  printStats: 60
  # don't write parsed logs to stdout
  suppress: true
  # Enable/disable GeoIP lookups
  # Startup of logagent might be slower, when downloading the GeoIP database
  geoipEnabled: false
  # Directory to store Logagent status and temporary files
  # this is equals to LOGS_TMP_DIR env variable
  diskBufferDir: /tmp/sematext-logagent

input:
  # a list of glob patterns to watch files to tail
  files:
    - '/var/log/**/*.log'

output:
  # index logs in Elasticsearch or Sematext Logs
  elasticsearch:
    module: elasticsearch
    url: https://logsene-receiver.sematext.com
    # default Elasticsearch index or Sematext Logs token to use:
    index: <LOGS_TOKEN or ES_INDEX>
```

Custom Configuration with `--config` Flag

Logagent can also run with a custom config file without running `logagent-setup`. Instead, you run Logagent as a command line tool. To use a custom config file run Logagent with the `--config` flag.

```
logagent --config custom.yml
```

Generate a Complete Configuration File with `--writeConfig` Flag

You can generate a complete config file with this command:

```
logagent --writeConfig <filename>
```

Here's what the generated config file looks like.

Configuration File Sections

There are 4 sections of the configuration file:

- Options
- Input
- Parser
- Output

Options

This section defines the global configuration of Logagent.

```
# Global options
options:
  # print stats every 60 seconds
  printStats: 60
  # don't write parsed logs to stdout
  suppress: true
  # Enable/disable GeoIP lookups
  # Startup of logagent might be slower, when downloading the GeoIP database
  geoipEnabled: false
  # Directory to store Logagent status and temporary files
  diskBufferDir: ./tmp
```

Input

The input section defines how you will ingest files into Logagent.

```
input:
  # a list of glob patterns to watch files to tail
  files:
    - '/var/log/**/*.log'
```

```
    - '/opt/myapp/logs/*.log'
  # listen to udp syslog protocol
  syslog:
    port: 514
  # listen to http to receive data from Heroku log drains
  heroku:
    port: 9999
  # listen to http to receive data from Cloud Foundry drains
  cloudFoundry:
    port: 8888
  # listen to stdin
  stdin: true
```

Parser

This section defines loading of custom pattern files or inline pattern definitions for the log parser. Check it out in detail here.

```
# optional, if not specified default patterns are used
parser:
  patternFiles:
    # load a list of pattern files to parse logs
    # later files overwrite settings from previous files
    # a 'hot reload' is done as soon one of the listed fields changes on disk
    - patterns1.yml
    - patterns2.yml
  # inline pattern definitions, to put on top of patterns list
  # loaded from files or default library. For inline patterns hot reload is not available.
  patterns:
    - # timestamped messages from /var/log/*.log on Mac OS X
      sourceName: !!js/regexp /\system\.log/ # catch all system.log files
      match:
        -
          type: system_log
          regex: !!js/regexp /([\w|\s]+\s+\d{2}\s[\d|\:]+)\s(.+?)\s(.+?)\s<(.+)>(.*)/
          fields: [ts,host,service,severity,message]
          dateFormat: MMM DD HH:mm:ss
```

Output

Logs can be shipped to various destinations including:

- Elasticsearch
- AWS Elasticsearch Service
- Sematext Cloud
- MQTT
- GELF

- Apache Kafka
- ZeroMQ
- InfluxDB
- ClickHouse DB
- …

The Elasticsearch output supports HTTPS and username/password in the URL. It is possible to use **multiple indices** to route logs from **different files** to **different indices** in Elasticsearch. All logs that don't match any rules in the indices section are **routed to the default** Elasticsearch **index**.

"'yaml hl_lines="4 7 16 22 23 28" output: # index logs in Elasticsearch or Sematext Cloud sematext: module: elasticsearch # URL to Elasticearch server, defaults to Sematext # Cloud logs receiver if not set url: https://logsene-receiver.sematext.com

```
# Proxy settings behind firewalls
# httpProxy:  http://localProxy:port
# httpsProxy: https://localHttpsProxy:port

# default index to use for all logs that don't match
# the indices specified in the indices section
# for Sematext Cloud use the Logs App Token here
index: 0a835c75-9847-4f74-xxxx

# specific index to use per logSource field of parsed logs
# logSource is by default the file name of the log file
# but it can be modified by JS transforms
# functions in the patterns.yml file
indices:
  4f70a0c7-9458-43e2-bbc5-xxxx:
  # list of RegEx matching logSource / filename
  # all logs matching logSource name will be indexed to above index
    - .*wifi.*
    - .*bluetooth.*
  999532c9-18f1-4c4b-8753-xxxx:
    - system\.log
    - access\.log
    - auth\.log
```

print parsed logs in YAML format to stdout # (only if options.suppress is set to false)
stdout: yaml # use 'pretty' for pretty json and 'ldjson' # for line delimited json (default)

"'

A collection of example config files is here.