Configuration guide

Please note Monitoring & Logging for Docker Enterprise including detailed feature descriptions and configuration examples for Sematext Docker Agent.

Configuration Parameters

Parameter / Environment variable	Description
Required Parameters	
SPM_TOKEN	SPM Application Token enables metric and event collection
LOGSENE_TOKEN	Logsene Application Token enables logging to Logsene, see logging specific parameters for filter options and Log Routing section to route logs from different containers to separate Logsene applications

Parameter / Environment variable	Description
`-v /var/run/docker.sock`	Path to the docker socket (optional, if dockerd provides TCP on 2375, see also DOCKER_PORT and DOCKER_HOST parameter)
TCP and TLS connection	If the Unix socket is not available Sematext Agent assumes the Container Gateway Address (autodetect) and port 2375 as default (no TLS) - this needs no configuration. In case the Docker Daemon TCP settings are different, you have to configure the TCP settings. The TCP settings can be modified with the following parameters

Parameter / Environment variable	Description
DOCKER_HOST	e.g. tcp://ip-reachable-from-container:2375/ - default value 'unix:///var/run/docker.sock'. When the Unix socket is not available the agent tries to connect to tcp://gateway:2375. In case a TCP socket is used there is no need to mount the Docker Unix socket as volume
DOCKER_PORT	Sematext Agent will use its gateway address (auto detect) with the given DOCKER_PORT
DOCKER_TLS_VERIFY	0 or 1
DOCKER_CERT_PATH	Path to your certificate files, mount the path to the container with "-v $DOCKER_{C}ERT_{P}ATH$:DOCKER_CERT_PATH"

Parameter / Environment variable	Description
Configuration via docker swarm secrets:	
CONFIG_FILE	Path to the configuration file, containing environment variables `key=value`. Default value: `/run/secrets/sematext-agent`. Create a secret with `docker secret create sematext-agent ./sematext-agent.cfg`. Start Sematext Docker agent with ‘docker service create –mode global –secret sematext-agent –mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock sematext/sematext-agent-docker
Optional Parameters:	

Parameter / Environment variable	Description
–privileged	The parameter might be helpful when Sematext Agent could not start because of limited permission to connect and write to the Docker socket /var/run/docker.sock. The privileged mode is a potential security risk, we recommend to enable the appropriate security. Please read about Docker security: https://docs.docker.com/engine/security/security/
HOSTNAME_LOOKUP_URL	On Amazon ECS, a meta data query must be used to get the instance hostname (e.g. "169.254.169.254/latest/meta-data/local-hostname")

Parameter / Environment variable	Description
HTTPS_PROXY	URL for a proxy server (behind firewalls)
LOGSENE_RECEIVER_URL	URL for bulk inserts into Logsene. Required for Sematext Enterprise (local IP:PORT) or Sematext Cloud Europe: https://logsene-receiver.eu.sematext.com
SPM_RECEIVER_URL	URL for bulk inserts into SPM. Required for Sematext Enterprise (local IP:PORT) or Sematext Cloud Europe: https://spm-receiver.eu.sematext.com/receiver/v1.

Parameter / Environment variable	Description
EVENTS_RECEIVER_URL	URL for SPM events receiver. Required for Sematext Enterprise (local IP:PORT) or Sematext Cloud Europe: https://event-receiver.eu.sematext.com
Docker Logs Parameters	
TAGGING_LABELS	A list of docker label names or environment variable names to tag container logs. Supporting wildcards e.g. TAGGING_LABELS=‘com.docker.swarm*,com.myapp.*’
Whitelist containers for logging	
MATCH_BY_NAME	Regular expression to white list container names

Parameter / Environment variable	Description
MATCH_BY_IMAGE	Regular expression to white list image names
Blacklist containers	
SKIP_BY_NAME	Regular expression to black list container names
SKIP_BY_IMAGE	Regular expression to black list image names for logging
PATTERNS_URL	Load pattern.yml via HTTP e.g. `-e PATTERNS_URL=https://raw.githubusercontent.com/sematext/lo`
LOGAGENT_PATTERNS	Pass patterns.yml via env. variable e.g. `-e LOGAGENT_PATTERNS="$(cat ./patters.yml)"`

Parameter / Environment variable	Description
PATTERN_MATCHING_ENABLED	Activate logagent-js parser, default value is `true`. To disable the log parser set the value to `false`. This could increase the throughput of log processing for nodes with a very high log volume.
-v /yourpatterns/patterns.yml:/etc/logagent/patterns.yml	to provide custom patterns for log parsing, see logagent-js
-v /tmp:/logsene-log-buffer	Directory to store logs, in a case of a network or service outage. Docker Agent deletes these files after successful transmission.
GEOIP_ENABLED	`true` enables GeoIP lookups in the log parser, default value: `false`

Parameter / Environment variable	Description
MAXMIND_DB_DIR	Directory for the Geo-IP lite database, must end with `/`. Storing the DB in a volume could save downloads for updates after restarts. Using `/tmp/` (ramdisk) could speed up Geo-IP lookups (requires add. ~30 MB main memory).

Parameter / Environment variable	Description
ENABLE_LOGSENE_STATS	Enables logging of transmission stats to Logsene. Default value 'false'. Provides a number of logs received, a number of logs shipped, number of failed/successful HTTP transmissions (bulk requests to Logsene) and retransmissions of failed requests.

Access to the Docker Socket / Docker API

Note that Docker Daemon can be configured to use Unix sockets (default), TCP sockets (default port 2375) and TLS sockets (authentication with certificates). Depending on your Docker setup, Sematext Agent needs to be configured to access the Docker Socket (API access).

Docker Unix Socket

Make sure that you have the permissions to access /var/run/docker.sock (or the actual location of the docker unix socket). E.g. use 'sudo' to run the "docker run" command.

Check your permissions first:

```
ls -la /var/run/docker.sock
srw-rw---- 1 root docker 0 Dec  3 07:52 /var/run/docker.sock
```

If you prefer to create a docker group to access docker without super user permissions, see https://docs.docker.com/engine/installation/linux/docker-ee/ubuntu/

How to activate the Unix socket in parallel to a TCP socket?
Check the configuration of the Docker Daemon in /etc/defaults/docker - it is possible to activate TCP and the Unix socket in parallel - simply add "-H unix:///var/run/docker.sock" and restart dockerd.

```
## /etc/defaults/docker
DOCKER_OPTS="-H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock"
```

Run Sematext Agent with access to the Unix socket:

```
docker run --name sematext-agent --restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-e SPM_TOKEN=YOUR_SPM_TOKEN -e LOGSENE_TOKEN=YOUR_LOGSENE_TOKEN \
sematext/sematextagent-docker
```

Docker TCP Socket

When Sematext Agent can't find the Unix socket it tries to connect to Docker Daemon via TCP on port 2375. The parameter DOCKER_PORT specifies the TCP port of the local Docker Daemon (set in /etc/default/docker in DOCKER_OPTS). This setup is typically used in Docker Swarm Nodes (TCP port 2375).

Run Sematext Agent with Access to Docker TCP socket:

```
docker run --name sematext-agent -e DOCKER_PORT=2375 -e SPM_TOKEN=YOUR_SPM_TOKEN -e LOGSENE_
```

Relevant Parameters:

-e DOCKER_PORT - Sematext Agent will use the container gateway address (autodetect) with the given DOCKER_PORT

-e DOCKER_HOST - e.g. tcp://ip-of-docker-host-reachable-from-container-network:2375/

Docker TLS Socket

To access the Docker TLS socket (on port 2376 or 3376 for Docker Swarm Master), Sematext Agent needs access to the certificates. Please use the following parameters to configure TLS access:

- -e DOCKER_HOST - e.g. tcp://ip-reachable-from-container:2375/
- -e DOCKER_TLS_VERIFY - 0 or 1
- -e DOCKER_CERT_PATH - path to your certificate files, mount the path to the container with "-v *DOCKER_CERT_PATH* :DOCKER_CERT_PATH"

Run Sematext Agent with access to Docker TLS socket:

```
# Example with docker-machine
docker-machine env --swarm swarm-master
```

```
# export DOCKER_TLS_VERIFY="1"
# export DOCKER_HOST="tcp://192.168.99.101:3376"
# export DOCKER_CERT_PATH="/Users/stefan/.docker/machine/machines/swarm-master"
# export DOCKER_MACHINE_NAME="swarm-master"
eval "$(docker-machine env swarm-master)"
docker run -d --name sematext-agent --restart=always
 -e SPM_TOKEN=YOUR_SPM_TOKEN  -e LOGSENE_TOKEN=YOUR_LOGSENE_TOKEN \
 -e DOCKER_TLS_VERIFY -e DOCKER_CERT_PATH -e DOCKER_HOST -v $DOCKER_CERT_PATH:$DOCKER_CERT_P
sematext/sematext-agent-docker
```

Blacklisting and Whitelisting Logs

Not all logs might be of interest, so sooner or later you will have the need to blacklist some log types. This is one of the reasons why Sematext Docker Agent automatically adds the following tags to all logs:

- Container ID
- Container Name
- Image Name
- Docker Compose Project Name
- Docker Compose Service Name
- Docker Compose Container Number

Using this "log metadata" you can whitelist or blacklist log outputs by image or container names. The relevant environment variables are:

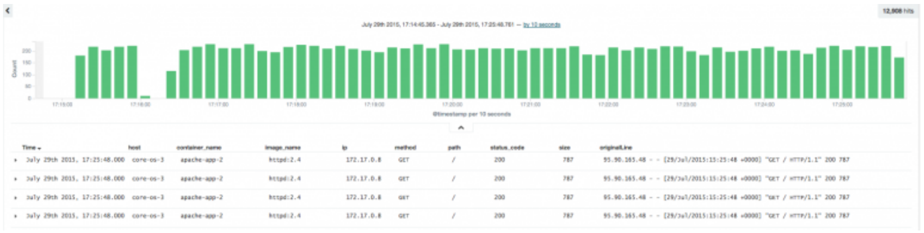
- MATCH_BY_NAME — a regular expression to whitelist container names
- MATCH_BY_IMAGE — a regular expression to whitelist image names
- SKIP_BY_NAME — a regular expression to blacklist container names
- SKIP_BY_IMAGE — a regular expression to blacklist image names

Automatic Parser for Container Logs

In Docker, logs are console output streams from containers. They might be a mix of plain text messages from start scripts and structured logs from applications. The problem is obvious – you can't just take a stream of log events all mixed up and treat them like a blob. You need to be able to tell which log event belongs to what container, what app, parse it correctly in order to structure it so you can later derive more insight and operational intelligence from logs, etc.

Sematext Docker Agent analyzes the event format, parses out data, and turns logs into structured JSON. This is important because the value of logs increases when you structure them — you can then slice and dice them and gain a lot more insight about how your containers, servers, and applications operate.

Traditionally it was necessary to use log shippers like Logstash, Fluentd or Rsyslog to parse log messages. The problem is that such setups are typically deployed in a very static fashion and configured for each input source. That does not work well in the hyper-dynamic world of containers! We have seen people struggling with the Syslog drivers, parsers configurations, log routing, and more! With its integrated automatic format detection, Sematext Docker Agent eliminates this struggle — and the waste of resources — both computing and human time that goes into dealing with such things! This integration has a low footprint, doesn't need retransmissions of logs to external services, and it detects log types for the most popular applications and generic JSON and line-oriented log formats out of the box!


12,908 hits
July 29th 2015, 17:14:45.365 - July 29th 2015, 17:25:48.761 — by 10 seconds
Count
@timestamp per 10 seconds

Time	host	container_name	image_name	ip	method	path	status_code	size	originalLine
July 29th 2015, 17:25:48.000	core-os-3	apache-app-2	httpd:2.4	172.17.0.8	GET	/	200	787	95.90.165.48 - - [29/Jul/2015:15:25:48 +0000] "GET / HTTP/1.1" 200 787
July 29th 2015, 17:25:48.000	core-os-3	apache-app-2	httpd:2.4	172.17.0.8	GET	/	200	787	95.90.165.48 - - [29/Jul/2015:15:25:48 +0000] "GET / HTTP/1.1" 200 787
July 29th 2015, 17:25:48.000	core-os-3	apache-app-2	httpd:2.4	172.17.0.8	GET	/	200	787	95.90.165.48 - - [29/Jul/2015:15:25:48 +0000] "GET / HTTP/1.1" 200 787
July 29th 2015, 17:25:48.000	core-os-3	apache-app-2	httpd:2.4	172.17.0.8	GET	/	200	787	95.90.165.48 - - [29/Jul/2015:15:25:48 +0000] "GET / HTTP/1.1" 200 787

Figure 1: Example: Apache Access Log fields generated by Sematext Docker Agent

For example, Sematext Docker Agent can parse logs from official images like:

- Nginx, Apache, Redis, MongoDB, MySQL
- Elasticsearch, Solr, Kafka, Zookeeper
- Hadoop, HBase, Cassandra
- Any JSON output with special support for Logstash or Bunyan format
- Plain text messages with or without timestamps in various formats
- Various Linux and Mac OSX system logs

Add patterns for log parsing

In addition, you can define your own patterns for any log format you need to be able to parse and structure. There are three options to pass individual log parser patterns:

- Configuration file in a mounted volume: `-v PATH_TO_YOUR_FILE:/etc/logagent/patterns.yml`
- Kubernetes ConfigMap example: Template for patterns.yml as ConfigMap
- Content of the configuration file in an environment variable: `-e LOGAGENT_PATTERNS="$(cat patterns.yml)"`
- Set patterns URL as environment variable: `-e PATTERNS_URL=http://yourserver/patterns.yml`

The file format for the patterns.yml file is based on JS-YAML, in short:

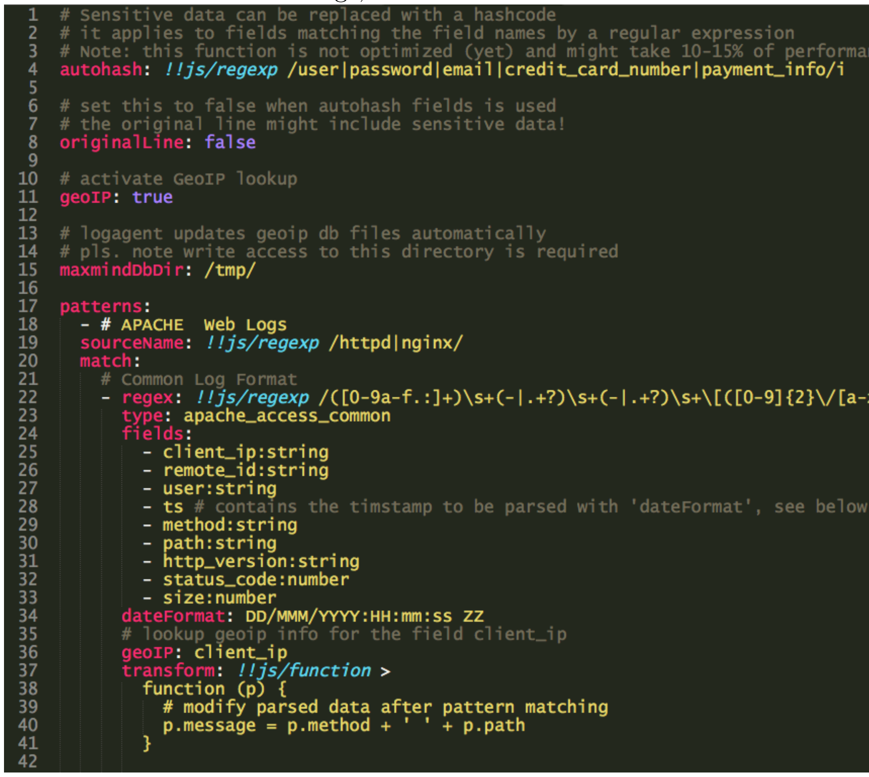
- `-` indicates an array element

- `!js/regexp` – indicates a JavaScript regular expression
- `!!js/function >` – indicates a JavaScript function

The file has the following properties:

- patterns: list of patterns, each pattern starts with "-"
  - match: a list of pattern definition for a specific log source (image/container)
    * sourceName: a regular expression matching the name of the log source. The source name is a combination of image name and container name.
  - regex: JS regular expression
  - fields: field list of extracted match groups from the regex
  - type: type used in Logsene (Elasticsearch Mapping)
  - dateFormat: format of the special fields 'ts', if the date format matches, a new field @timestamp is generated
  - transform: A JavaScript function to manipulate the result of regex and date parsing

The following example shows pattern definitions for web server logs, which is one of the patterns available by default:



```
# Sensitive data can be replaced with a hashcode
# it applies to fields matching the field names by a regular expression
# Note: this function is not optimized (yet) and might take 10-15% of performa
autohash: !!js/regexp /user|password|email|credit_card_number|payment_info/i

# set this to false when autohash fields is used
# the original line might include sensitive data!
originalLine: false

# activate GeoIP lookup
geoIP: true

# logagent updates geoip db files automatically
# pls. note write access to this directory is required
maxmindDbDir: /tmp/

patterns:
  - # APACHE  Web Logs
  sourceName: !!js/regexp /httpd|nginx/
  match:
    # Common Log Format
    - regex: !!js/regexp /([0-9a-f.:]+)\s+(-|.+?)\s+(-|.+?)\s+\[([0-9]{2}\/[a-
      type: apache_access_common
      fields:
        - client_ip:string
        - remote_id:string
        - user:string
        - ts # contains the timstamp to be parsed with 'dateFormat', see below
        - method:string
        - path:string
        - http_version:string
        - status_code:number
        - size:number
      dateFormat: DD/MMM/YYYY:HH:mm:ss ZZ
      # lookup geoip info for the field client_ip
      geoIP: client_ip
      transform: !!js/function >
        function (p) {
          # modify parsed data after pattern matching
          p.message = p.method + ' ' + p.path
        }

```

This example shows a few very interesting features:

- Masking sensitive data with "autohash" property, listing fields to be re-

placed with a hash code
- Automatic Geo-IP lookups including automatic updates for Maxmind Geo-IP lite database
- Post-processing of parsed logs with JavaScript functions

The component for detecting and parsing log messages — logagent-js — is open source and contributions for even more log formats are welcome.

Log Routing

Routing logs from different containers to separate Logsene Apps can be configured via docker labels (or environment variables e.g. on Kubernetes). Simply tag a container with the label (or environment variable) `LOGSENE_TOKEN=YOUR_LOGSENE_TOKEN`. Sematext Agent inspects the containers for this label and ships the logs to the specified Logsene App.

To disable logging to Logsene/Elasticsearch simply label the container with `LOGSENE_ENABLED=false`. `LOGSENE_ENABLED=true` enables logging for the container again.

Example: The following command will start Nginx webserver and logs for this container will be shipped to the related Logsene App.

```
docker run --label LOGSENE_TOKEN=REPLACE_WITH_YOUR_LOGSENE_TOKEN -p 80:80 nginx
# or use environment variable on Kubernetes (no support for Docker labels)
# docker run -e LOGSENE_TOKEN=REPLACE_WITH_YOUR_LOGSENE_TOKEN -p 80:80 nginx
```

All other container logs will be shipped to the Logsene App specified in the docker run command for `sematext/sematext-agent-docker` with the environment variable `LOGSENE_TOKEN`.

Please refer to Docker Log Management & Enrichment for further details.

Known Issues

Conflict with Docker logging-drivers. Sematext Docker Agent is running with a valid Logsene Token, but Logsene does not show container logs.

Please note that Sematext Docker Agent collects logs via Docker Remote API. If you use a Docker logging-driver other than the default json-file driver, logs will not be available via the Docker Remote API. Please make sure that your container or docker daemon uses json-file logging driver. This ensures that logs are exposed via Docker Remote API. To check, run the “docker logs” command. If “docker logs CID” shows container logs then Sematext Docker Agent should be able to collect the logs as well.