title: Installing Logagent on Docker description: Logagent, Sematext log shipper and Logstash alternative, is available as Docker image. It has automatic Docker installation and seamless logging system service integration with our log management and analysis platform

Certified Logagent Images

Please note the following instructions use the latest Logagent image from Docker Hub. If you want to use certified images please pull the image from Red Hat and Docker registries.

- Red Hat Certified Image: `docker pull registry.connect.redhat.com/sematext/logagent`
- Docker Certified Image: `docker pull store/sematext/logagent`

Installation for Docker

The Logagent Docker image is pre-configured for the log collection on container platforms. It runs as a tiny container on every Docker host and collects logs for all cluster nodes and their containers. All container logs are enriched with Kubernetes and Docker EE/Swarm metadata.

See: sematext/logagent on Docker Hub.

Getting started

To ship logs to Sematext you will need a Logs App Token. If you don't have Logs Apps yet, you can create Apps now.

The Logagent container can be configured through the following environment variables:

- **REGION**: Sematext Cloud region **US** or **EU** (default: US). The receiver URL will be set to EU/US default values. When using REGION, you don't need to set `LOGS_RECEIVER_URL` (see below).

- **LOGS_RECEIVER_URL**: The URL of your Elasticsearch Endpoint *(defaults to Sematext Cloud US `https://logsene-receiver.sematext.com`)*.

  - For Sematext Europe use `https://logsene-receiver.eu.sematext.com`.
  - For Elasticsearch `https://elasticserch-server-name:9200`.

- **LOGS_TOKEN**: The index to ship logs to. For Sematext use the Logs App Token.

- **LOGAGENT_ARGS**: Additional command line arguments for Logagent

  to specify a log source name or

  to act as syslog server. Please refer to Logagent command line arguments in the Logagent Documentation

- **LOG_GLOB**: Semicolon-separated list of file globs

  ```
  LOG_GLOB=/mylogs/**/*.log;/var/log/**/*.log
  ```

  Mount your server log files into the container using a Docker volume e.g.

  ```
  -v /var/log:/mylogs
  ```

  This feature is developed with the Glob module. Here's a guide on how to use Glob patterns.

- **-v /var/run/docker.sock:/var/run/docker.sock** - Collect container logs by mounting the docker socket (mandatory)

Docker Run

The most basic way to start is via docker run command:

```
docker pull sematext/logagent
docker run -d --restart=always --name logagent \
-e LOGS_TOKEN=YOUR_LOGS_TOKEN \
-e LOGS_RECEIVER_URL="https://logsene-receiver.sematext.com" \
-v /var/run/docker.sock:/var/run/docker.sock sematext/logagent
```

Docker Compose

To use Docker Compose create docker-compose.yml as follows and insert real tokens:

```
# docker-compose.yml
logagent:
  image: 'sematext/logagent:latest'
  environment:
    - LOGS_TOKEN=YOUR_LOGS_TOKEN
    - LOGS_RECEIVER_URL="https://logsene-receiver.sematext.com"
  cap_add:
    - SYS_ADMIN
  restart: always
  volumes:
    - '/var/run/docker.sock:/var/run/docker.sock'
```

Then start Logagent with the docker-compose file:

```
docker-compose up -d
```

Docker Swarm and Docker Enterprise

Connect your Docker client to Swarm or UCP remote API endpoint and deploy Logagent with following docker command with your Logs App Token:

```
docker service create --mode global --name st-logagent \
--restart-condition any \
--mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock \
-e LOGS_TOKEN="REPLACE THIS WITH YOUR LOGS TOKEN" \
-e REGION=US \
sematext/logagent:latest
```

Kubernetes and OpenShift

Run Logagent as Kubernetes DaemonSet.

First, create logagent-daemonset.yml

```
curl -o logagent-daemonset.yml https://raw.githubusercontent.com/sematext/logagent-js/maste
```

Then run the DaemonSet:

```
kubectl create -f logagent-daemonset.yml
```

On Red Hat OpenShift use the "oc" command instead of kubectl.

```
oc apply -f logagent-daemonset.yml
```

Kubernetes with containerd and IBM Cloud

Kubernetes can use cointainerd as container engine. In this case Logagent can't use the Docker remote API to retrieve logs and metadata. Instead, logs are collected from containerd log files, which requires access to the relevant directories. The Logagent input-filter for containerd supports:

- Tailing log files from `/var/log/containers/`, `/var/log/pods` and `/var/data/kubeletlogs`
- Enrichment of logs with podName, namespace, containerName, containerId
- Parsing containerd log headers (timestamp, stream, flags)
- Parsing message content with logagent parser library

Run Logagent as Kubernetes DaemonSet.

First, create ibm-cloud-logagent-ds.yml

```
curl -o ibm-cloud-logagent-ds.yml  https://raw.githubusercontent.com/sematext/logagent-js/ma
```

Set your Logs App Token in the spec.env section in the `ibm-cloud-logagent-ds.yml` file.

Then run the DaemonSet:

```
kubectl create -f ibm-cloud-logagent-ds.yml
```

Mesos / Marathon

The following configuration will activate Logagent on every node in the Mesos cluster. Please note that you have to specify the number of Mesos nodes (instances) and Logs App Token. An example call to the Marathon API:

```
curl -XPOST -H "Content-type: application/json" http://your_marathon_server:8080/v2/apps  -d
{
  "container": {
    "type": "DOCKER",
    "docker": {
      "image": "sematext/logagent"
    },
    "volumes": [
      {
        "containerPath": "/var/run/docker.sock",
        "hostPath": "/var/run/docker.sock",
        "mode": "RW"
      }
    ],
    "network": "BRIDGE"
  },
  "env": {
        "LOGS_TOKEN": "YOUR_LOGS_TOKEN",
        "LOGS_RECEIVER_URL": "https://logsene-receiver.sematext.com"

  },
  "id": "sematext-logagent",
  "instances": 1,
  "cpus": 0.1,
  "mem": 100,
  "constraints": [
    [
      "hostname",
      "UNIQUE"
    ]
  ]
}
```

Configuration Parameters

Mandatory Parameters

Parameter / Environment variable

Description

LOGS_TOKEN

Logs Token enables logging to Sematext Cloud, see logging specific parameters for filter options and Log Routing section to route logs from different containers to separate Logs Apps (indices)

-v /var/run/docker.sock

Path to the docker socket

Optional Parameters

Parameter / Environment variable

Description

REGION

Sematext Cloud region US or EU (default: US). The receiver URL will be set to EU/US default values. When using REGION, you don't need to set LOGS_RECEIVER_URL (see below).

LOG_GLOB

Semicolon-separated list of file globs (e.g. /var/log//*.log;/mylogs//*.log) to collect log files from the host, assuming the log files are mounted to /mylogs using Docker -v /var/logs:/mylogs

LOGAGENT_ARGS

Additional command line arguments for Logagent (e.g. LOGAGENT_ARGS="-n httpd" to specify a log source name or LOGAGENT_ARGS="-u 514" to act as syslog server)

HTTPS_PROXY

URL for a proxy server (behind firewalls)

LOGS_RECEIVER_URL

URL for bulk inserts into Sematext Cloud. Required for Sematext Enterprise (local IP:PORT) or Sematext Cloud Europe: https://logsene-receiver.eu.sematext.com

LOGS_RECEIVER_URLS

Specify multiple receiver URLs for bulk inserts into Sematext Cloud. Required for Sematext Enterprise (local IP:PORT) or Sematext Cloud Europe: https://logsene-receiver.eu.sematext.com

JOURNALD_UPLOAD_PORT

Port number for the collection of journald logs, forwarded by systemd-journal-upload.service. Equals to Logagent argument –journald PORT.

SYSTEMD_UNIT_FILTER

A regular expression to filter journald logs by systemd unit name, e.g. “ssh.service|docker.service”. The default value is ”.*”.

CONFIG_FILE

Path to the configuration file, containing environment variables key=value. Default value: /run/secrets/logagent. Create a secret with docker secret create logagent ./logagent.cfg. Start Logagent with ‘docker service create –mode global –secret logagent –mount type=bind,src=/var/run/docker.sock,dst=/var/run/docker.sock sematext/logagent

LA_CONFIG

Point to the location of a logagent config file. E.g.

LA_CONFIG_OVERRIDE

Ignores env vars for LOGS_TOKEN, REGION, and LOGS_RECEIVER_URL. E.g.

–privileged

The parameter might be helpful when Logagent 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/

Docker Logs Parameters

Parameter / Environment variable

Description

TAGGING_LABELS

A list of docker label names or environment variable names to tag container logs. Supporting wildcards. Default value:

IGNORE_LOGS_PATTERN

Filter logs with a regular expression.

This will match log lines that contain “healthcheck” or “ping” in the message, and drop them. Only add the regular expression without forward slashes.

LOGSENE_ENABLED_DEFAULT

Enables log collection for containers having no explicit label/environment variable LOGSENE_ENABLED set. Default value: true. See section Log Routing.

Whitelist Containers for Logging

Parameter / Environment variable

Description

MATCH_BY_NAME

Regular expression to whitelist container names.

This will match any container name that contains “nginx”. Only add the regular expression without forward slashes.

MATCH_BY_IMAGE

Regular expression to whitelist image names.

This will match any image that contains “nginx”. Only add the regular expression without forward slashes.

Blacklist Containers

Parameter / Environment variable

Description

SKIP_BY_NAME

Regular expression to blacklist container names for logging.

This will match any container name that contains “nginx”. Only add the regular expression without forward slashes.

SKIP_BY_IMAGE

Regular expression to blacklist image names for logging.

This will match any image that contains “nginx”. Only add the regular expression without forward slashes.

Set Log Patterns

Logagent supports various log formats defined in patterns.yml file. The Log Parser Patterns can be customized by proving your YAML file.

Parameter / Environment variable

Description

PATTERNS_URL

Load pattern.yml via HTTP e.g.

LOGAGENT_PATTERNS

Pass patterns.yml via env. variable e.g.

LOGAGENT_PATTERNS_BASE64

Set to "true" if the LOGAGENT_PATTERNS patterns file you are passing in via env. variable is base64 encoded e.g

. Useful if your patterns file is not getting set properly due to shell interpretation or otherwise.

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 /patterns.yml:/etc/logagent/patterns.yml

Mount a patterns file to customize patterns for log parsing

Other Options

Parameter / Environment variable

Description

-v /tmp:/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

Enables GeoIP lookups in the log parser, default value: "false"

GEOIP_FIELDS

A list of fields to perform geo IP lookups e.g.

MAXMIND_LICENSE_KEY

Your MaxMind license key

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).

REMOVE_FIELDS

Removes fields from parsed/enriched logs. E.g.

Configuration Manual

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 Logagent 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

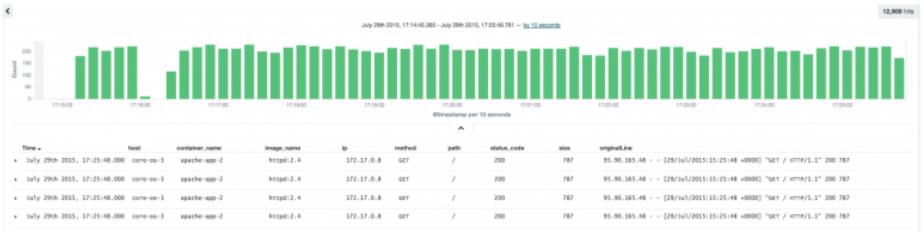
Some log messages are useless or noisy, like access to health check URLs in Kubernetes. You can filter such messages via regular expressions by setting the following environment variable:

Container Log Parsing

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.

Logagent 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, Logagent 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 Logagent

For example, Logagent 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

Adding log parsing patterns

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:

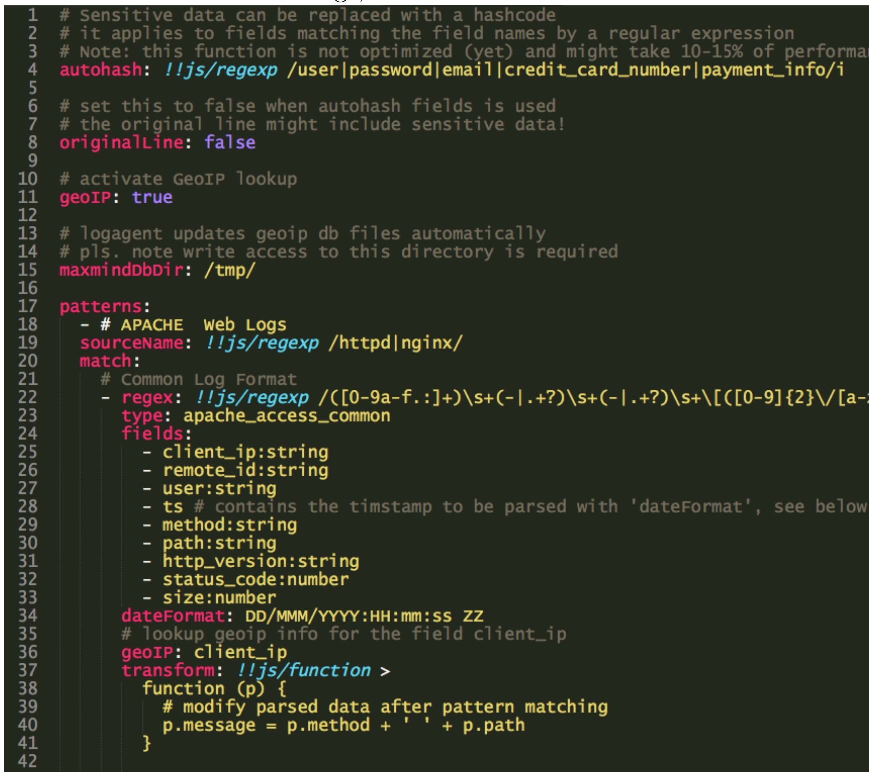
- `-` indicates an array element
- `!js/regexp` – indicates a JavaScript regular expression
- `!!js/function >` – indicates a JavaScript function

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 Sematext Cloud (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 replaced 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 Sematext Cloud Logs Apps can be configured via Docker labels, or environment variables, e.g. on Kubernetes.

Log Routing with Env Vars

Tag a container with the label, or environment variable `LOGS_TOKEN=YOUR_LOGS_TOKEN`. Logagent inspects the containers for this label and ships the logs to the specified Logs App.

The following container environment variables and labels are supported:

- `LOGS_ENABLED=<true|false>` - switch log collection for the container on or off. Note, the default value is configurable in Logagent configuration via the setting `LOGSENE_ENABLED_DEFAULT`.

- `LOGS_TOKEN=<YOUR_LOGS_TOKEN>` - logs token for the container
- `LOGS_RECEIVER_URL=<URL>` - set a single log destination. The URL should not include the token or index of an Elasticsearch API endpoint. E.g. `https://logsene-receiver.sematext.com`. **Requires you to set the `LOGS_TOKEN` env var**.
- `LOGS_RECEIVER_URLS=<URL, URL, URL>` - set multiple log destinations. The URL should include the token or index of an Elasticsearch API endpoint. E.g. `https://logsene-receiver.sematext.com/<YOUR_LOGS_TOKEN>`. **Does not require setting the `LOGS_TOKEN` env var**.

The following Kubernetes Pod annotations are equivalent:

- `sematext.com/logs-enabled=<true|false>`
- `sematext.com/logs-token=<YOUR_LOGS_TOKEN>`
- `sematext.com/logs-receiver-url=<URL>`
- `sematext.com/logs-receiver-urls=<URL, URL, URL>`

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

```
docker run --label LOGS_TOKEN=REPLACE_WITH_YOUR_LOGS_TOKEN -p 80:80 nginx
```

Or use environment variables for Kubernetes:

```
docker run -e LOGS_TOKEN=REPLACE_WITH_YOUR_LOG_TOKEN -p 80:80 nginx
```

All other container logs will be shipped to the Logs App specified in the docker run command for `sematext/logagent` with the environment variable `LOGS_TOKEN`.

By default, all logs from all containers are collected and sent to Sematext Cloud. You can change this default by setting the `LOGSENE_ENABLED_DEFAULT=false` label for the Logagent container. This default can be overridden, on each container, through the `LOGS_ENABLED` label.

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

Log Routing with Logagent Config File

Add the `LA_CONFIG_OVERRIDE=true` env var to your Logagent instance, alongside the `LA_CONFIG=/etc/sematext/logagent.conf`. Next, add a `logagent.conf` file as a config resource in the `configs` section of your Docker Compose/Stack file.

```
version: "3.7"
services:
  st-logagent:
    image: 'sematext/logagent:latest'
    environment:
      - LA_CONFIG=/etc/sematext/logagent.conf
      - LA_CONFIG_OVERRIDE=true

    # if you're using Docker Stack use configs,
    # otherwise use volumes or secrets
    configs:
      - source: logagent_config
        target: /etc/sematext/logagent.conf
    deploy:
      mode: global
      restart_policy:
        condition: on-failure
    volumes:
      - '/var/run/docker.sock:/var/run/docker.sock'

configs:
  logagent_config:
    file: ./logagent.yml
```

The `logagent.yml` looks like this:

```
options:
  printStats: 60
  suppress: true
  geoipEnabled: true
  diskBufferDir: /tmp/sematext-logagent

input:
  docker:
    module: docker-logs
    socket: /var/run/docker.sock
    labelFilter: com.docker.*,io.kubernetes.*,annotation.*

parser:
  patternFiles:
    - /etc/logagent/patterns.yml

outputFilter:
  dockerEnrichment:
    module: docker-enrichment
    autodetectSeverity: true

  backward_compatible_field_name:
    module: !!js/function >
      function (context, config, eventEmitter, data, callback)  {
        // debug: dump data to error stream
        // console.error(data.labels)
        if (data.labels && data.labels['com_docker_swarm_service_name']) {
          data.swarm_service_name = data.labels ['com_docker_swarm_service_name']
          // data.swarm_service_id = data.labels ['com_docker_swarm_service_id']
          // data.swarm_task_name = data.labels ['com_docker_swarm_task_name']
        }
        callback(null, data)
      }

output:
  sematext-logs1:
    module: elasticsearch
    url: https://logsene-receiver.sematext.com
    index: LOGS_TOKEN_1
  sematext-logs2:
    module: elasticsearch
    url: https://logsene-receiver.sematext.com
    index: LOGS_TOKEN_2
  elasticsearch:
    module: elasticsearch
    url: http://local-es:9200
    index: app
```

In the `output` section you can configure as many outputs as you want.

Known Issues

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

Please note that Logagent 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 Logagent should be able to collect the logs as well.

Please check the parsed timestamps! Logs with timestamps in the future (or several months or years in the past) might not be displayed in Sematext Cloud.