title: Best Practices to Reduce Log Volume with Logagent description: Reduce log volume to save storage or money for volume based licenses.

Centralized Logs can often be very noisy. Developers love verbose log output for potential troubleshooting. Do you want to collect all logs, all the time, and pay for processing and storage? Probably not! A good practice is to reduce the log volume and focus on relevant log messages until a problem arrives. Log shipper settings can be adjusted to fetch all logs for troubleshooting and debugging if needed. Let's have a look at the available Logagent filters for adjusting the log volume for operational needs.

Here are a few examples that show how you can configure Logagent plugins to reduce log volume.

First of all, it's essential to understand the performance impact of filters in various processing stages to know how Logagent works internally.

Input plugins collect data. Those plugins collect data from given data sources. Dropping data sources is the most efficient way to reduce log volume. However, some relevant messages hide in verbose log sources. Sometimes you want to see errors or specific warnings.

Input filters are functions that process logs, line by line, before parsing them. Filtering the input right away saves processing power for parsing logs and is very good to catch logs with specific keywords.

Output filters process log events after they are parsed and can be applied to single and multi-line log messages. Here you can place more advanced logic to skip logs or modify log events. Moreover, some output plugins have features to remove data before being transferred. An example is if you want to remove sensitive and private customer data.

The most efficient way to reduce log volume is to stop collecting logs from noisy data sources. Do you need **all logs from `/var/log/**/*.log`**, **all systemd units** and **all container logs**? Most of the time, you don't. Configure Logagent Input Plugins to skip noisy sources. Most of the input plugins accept a list of input sources or offer settings to filter data sources. Here are the most common examples of file input and Docker container logs.

File Input Plugin

Limit the data sources to dedicated log files or directories, instead of using /var/log/**/*.log (default setting in Logagent)

```
input:
  files:
    - /var/log/system.log
    - /var/log/kernel.log
    - /var/log/audit.log
    - /var/log/nginx/access_log
    - /var/log/myapp/*.log
```

Reduce Container Log volume

Skip some container logs entirely

Use the environment variables

- SKIP_BY_IMAGE
- SKIP_BY_NAME
- MATCH_BY_IMAGE
- MATCH_BY_NAME

The variables above accept regular expressions as values. To exclude a complete Kubernetes namespace, you can use `SKIP_BY_NAME=NOISY_NAMESPACE|OTHER_NOISY_NAMESPACE`.

There is another way to disable logs from noisy containers. You can tag your noisy application containers with the label or environment variable `LOGS_ENABLED="false"`. Logagent does not collect logs from containers having the `LOGS_ENABLED=false` label or environment variable. E.g. `docker run -e LOGS_ENBLED=false my_noisy_app`

Filter mechanisms are very efficient because filters are applied when Logagent or a new container starts. So there is no additional processing load.

Log Routing - put logs in separate buckets with different retention times

Deleting logs is often not an option. However, you can separate logs from each other to keep Logs Apps clean, and noise-free. In Sematext Cloud, you can set different retention times for each Logs App. Setting a short retention time just for the verbose Apps can save you money!

To ship logs of verbose containers to a different Logs App set a separate `LOGS_TOKEN` variable for your application container: `docker run -e LOGS_TOKEN=SeparateSematextLogsToken my_noisy_app`

Now, you don't see the noisy logs in your main Logs App, and you can keep a longer retention time for your relevant logs! You don't lose verbose logs, and you can troubleshoot while you save money.

Filter out health checks and readiness probes

You want to get rid of logs from health checks, frequently triggered by Kubernetes or Docker? To skip health checks and ping requests you can use the environment variable

```
IGNORE_LOGS_PATTERN="/health|/ping"
```

when you set up a Logagent container.

Remove fields before shipping to Sematext Cloud

Logagent supports the environment variable `REMOVE_FIELDS` for the Elasticsearch output plugin. You can specify a list of fields for removal with `REMOVE_FIELDS='logSource,labels"` when you set up Logagent in a container. For more information see the complete Docker setup instructions

Logagent Input Filters

Input Filters process logs line by line before parsing. At this stage of log processing you can't drop multi-line log messages. However, for single-line logs, it's the perfect place to drop events by using regular expressions similar to the `grep` UNIX command. All messages removed in the input stage don't create load in the other processing stages.

```
inputFilter:
  - module: grep
    config:
      matchSource: !!js/regexp /myapp.log/
      include: !!js/regexp /info|error/i
      exclude: !!js/regexp /test|debug|warning/i
```

Logagent Output Filters

Output Filters apply after parsing logs. At this stage you can use any parsed field or custom logic of field combinations. Dropping events is possible before Logagent transmits data to any output destination.

The drop-events allows you to specify `include` and `exclude` criteria as a regular expression for each field name.

```
outputFilter:
  dropEventsFilter:
    module: drop-events
    debug: false
    filters:
      severity:
        # include: !!js/regexp /error|warn/i
        exclude: !!js/regexp /debug/i
      service:
        exclude: !!js/regexp ntp|snmpd|postfix
```

Another interesting plugin is the "remove-fields" plugin. It takes a list of fields for removal:

```
outputFilter:
  remove-fields:
    module: remove-fields
    matchSource: !!js/regexp /myAppLogs.log/i
```

```
fields:
  - logSource
  - labels
```