6 Log file monitoring

Overview

Zabbix can be used for centralized monitoring and analysis of log files with/without log rotation support.

Notifications can be used to warn users when a log file contains certain strings or string patterns.

To monitor a log file you must have:

- Zabbix agent running on the host
- log monitoring item set up

The size limit of a monitored log file depends on large file support.

Configuration

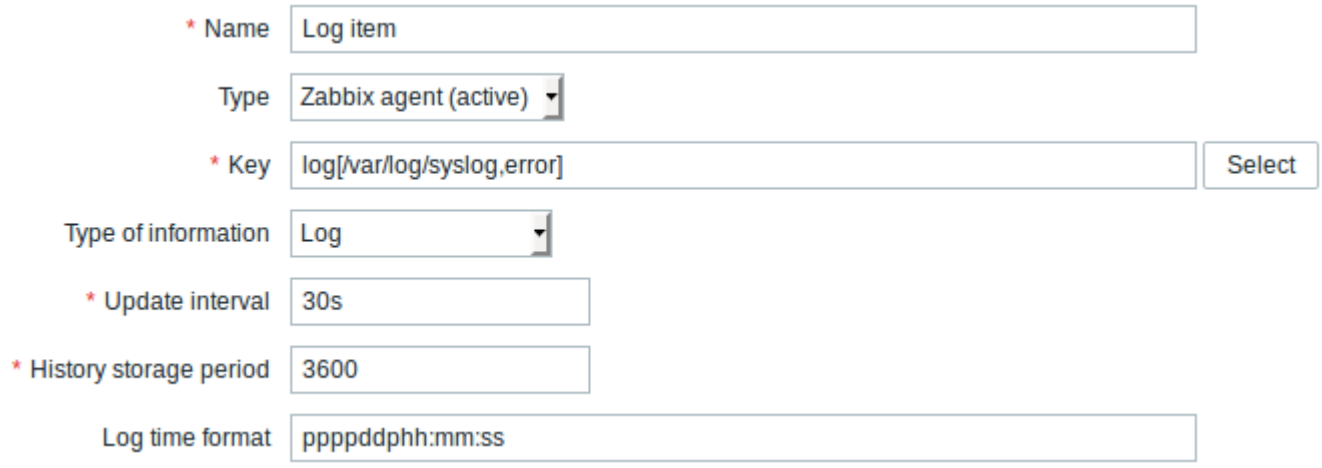
Verify agent parameters

Make sure that in the agent configuration file:

- 'Hostname' parameter matches the host name in the frontend
- Servers in the 'ServerActive' parameter are specified for the processing of active checks

Item configuration

Configure a log monitoring item.



All mandatory input fields are marked with a red asterisk.

Specifically for log monitoring items you enter:

Type	Select **Zabbix agent (active)** here.

Key	Use one of the following item keys: **log[]** or **logrt[]**: These two item keys allow to monitor logs and filter log entries by the content regexp, if present. For example: `log[/var/log/syslog,error]`. Make sure that the file has read permissions for the 'zabbix' user otherwise the item status will be set to 'unsupported'. **log.count[]** or **logrt.count[]**: These two item keys allow to return the number of matching lines only. See supported Zabbix agent item key section for details on using these item keys and their parameters.
Type of information	Select: For log[] or logrt[] items - `Log`; For log.count[] or logrt.count[] items - `Numeric (unsigned)`. If optionally using the `output` parameter, you may select the appropriate type of information other than `Log`. Note that choosing a non-Log type of information will lead to the loss of local timestamp.
Update interval (in sec)	The parameter defines how often Zabbix agent will check for any changes in the log file. Setting it to 1 second will make sure that you get new records as soon as possible.
Log time format	In this field you may optionally specify the pattern for parsing the log line timestamp. If left blank the timestamp will not be parsed. Supported placeholders: * **y**: *Year (0001-9999)* * **M**: *Month (01-12)* * **d**: *Day (01-31)* * **h**: *Hour (00-23)* * **m**: *Minute (00-59)* * **s**: *Second (00-59)* For example, consider the following line from the Zabbix agent log file: " 23480:20100328:154718.045 Zabbix agent started. Zabbix 1.8.2 (revision 11211)." It begins with six character positions for PID, followed by date, time, and the rest of the line. Log time format for this line would be "ppppp:yyyyMMdd:hhmmss". Note that "p" and ":" chars are just placeholders and can be anything but "yMdhms".

Important notes

- The server and agent keep the trace of a monitored log's size and last modification time (for logrt) in two counters. Additionally:
  - The agent also internally uses inode numbers (on UNIX/GNU/Linux), file indexes (on Microsoft Windows) and MD5 sums of the first 512 log file bytes for improving decisions when logfiles get truncated and rotated.
  - On UNIX/GNU/Linux systems it is assumed that the file systems where log files are stored report inode numbers, which can be used to track files.
  - On Microsoft Windows Zabbix agent determines the file system type the log files reside on and uses:
    - On NTFS file systems 64-bit file indexes.

        - On ReFS file systems (only from Microsft Windows Server 2012) 128-bit file IDs.
        - On file systems where file indexes change (e.g. FAT32, exFAT) a fall-back algorithm is used to take a sensible approach in uncertain conditions when log file rotation results in multiple log files with the same last modification time.
    - The inode numbers, file indexes and MD5 sums are internally collected by Zabbix agent. They are not transmitted to Zabbix server and are lost when Zabbix agent is stopped.
    - Do not modify the last modification time of log files with 'touch' utility, do not copy a log file with later restoration of the original name (this will change the file inode number). In both cases the file will be counted as different and will be analyzed from the start, which may result in duplicated alerts.
    - If there are several matching log files for `logrt[]` item and Zabbix agent is following the most recent of them and this most recent log file is deleted, a warning message "`there are no files matching "<regexp mask>" in "<directory>"` is logged. Zabbix agent ignores log files with modification time less than the most recent modification time seen by the agent for the `logrt[]` item being checked.
- The agent starts reading the log file from the point it stopped the previous time.
- The number of bytes already analyzed (the size counter) and last modification time (the time counter) are stored in the Zabbix database and are sent to the agent to make sure the agent starts reading the log file from this point in cases when the agent is just started or has received items which were previously disabled or not supported. However, if the agent has receives a non-zero size counter from server, but the logrt[] or logrt.count[] item has not found and does not find matching files, the size counter is reset to 0 to analyze from the start if the files appear later.
- Whenever the log file becomes smaller than the log size counter known by the agent, the counter is reset to zero and the agent starts reading the log file from the beginning taking the time counter into account.
- If there are several matching files with the same last modification time in the directory, then the agent tries to correctly analyze all log files with the same modification time and avoid skipping data or analyzing the same data twice, although it cannot be guaranteed in all situations. The agent does not assume any particular log file rotation scheme nor determines one. When presented multiple log files with the same last modification time, the agent will process them in a lexicographically descending order. Thus, for some rotation schemes the log files will be analyzed and reported in their original order. For other rotation schemes the original log file order will not be honored, which can lead to reporting matched log file records in altered order (the problem does not happen if log files have different last modification times).
- Zabbix agent processes new records of a log file once per *Update interval* seconds.
- Zabbix agent does not send more than **maxlines** of a log file per second. The limit prevents overloading of network and CPU resources and overrides the default value provided by **MaxLinesPerSecond** parameter in the agent configuration file.
- To find the required string Zabbix will process 10 times more new lines than set in MaxLinesPerSecond. Thus, for example, if a `log[]` or `logrt[]` item has *Update interval* of 1 second, by default the agent will analyse no more than 200 log file records and will send no more than 20 matching records to Zabbix server in one check. By increasing **MaxLinesPerSecond** in the agent configuration file or setting **maxlines** parameter in the item key, the limit can be increased up to 10000 analysed log file records and 1000 matching records sent to Zabbix server in one check. If the *Update interval* is set to 2 seconds the limits for one check would be set 2 times higher than with *Update interval* of 1 second.
- Additionally, log and log.count values are always limited to 50% of the agent send buffer size, even if there are no non-log values in it. So for the **maxlines** values to be sent in one connection (and not in several connections), the agent BufferSize parameter must be at least maxlines x 2.

- In the absence of log items all agent buffer size is used for non-log values. When log values come in they replace the older non-log values as needed, up to the designated 50%.
- For log file records longer than 256kB, only the first 256kB are matched against the regular expression and the rest of the record is ignored. However, if Zabbix agent is stopped while it is dealing with a long record the agent internal state is lost and the long record may be analysed again and differently after the agent is started again.
- Special note for "\" path separators: if file_format is "file\.log", then there should not be a "file" directory, since it is not possible to unambiguously define whether "." is escaped or is the first symbol of the file name.
- Regular expressions for `logrt` are supported in filename only, directory regular expression matching is not supported.
- On UNIX platforms a `logrt[]` item becomes NOTSUPPORTED if a directory where the log files are expected to be found does not exist.
- On Microsoft Windows, if a directory does not exist the item will not become NOTSUPPORTED (for example, if directory is misspelled in item key).
- An absence of log files for `logrt[]` item does not make it NOTSUPPORTED. Errors of reading log files for `logrt[]` item are logged as warnings into Zabbix agent log file but do not make the item NOTSUPPORTED.
- Zabbix agent log file can be helpful to find out why a `log[]` or `logrt[]` item became NOTSUPPORTED. Zabbix can monitor its agent log file except when at DebugLevel=4.

Extracting matching part of regular expression

Sometimes we may want to extract only the interesting value from a target file instead of returning the whole line when a regular expression match is found.

Since Zabbix 2.2.0, log items have the ability to extract desired values from matched lines. This is accomplished by the additional **output** parameter in `log` and `logrt` items.

Using the 'output' parameter allows to indicate the subgroup of the match that we may be interested in.

So, for example

```
log[/path/to/the/file,"large result buffer allocation.*Entries:
([0-9]+)",,,,\1]
```

should allow returning the entry count as found in the content of:

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result
buffer allocation - /Length: 437136/Entries: 5948/Client Ver: >=10/RPC
ID: 41726453/User: AUser/Form: CFG:ServiceLevelAgreement
```

The reason why Zabbix will return only the number is because 'output' here is defined by **\1** referring to the first and only subgroup of interest: **([0-9]+)**

And, with the ability to extract and return a number, the value can be used to define triggers.

Using maxdelay parameter

The 'maxdelay' parameter in log items allows ignoring some older lines from log files in order to get the most recent lines analyzed within the 'maxdelay' seconds.

Specifying 'maxdelay' > 0 may lead to **ignoring important log file records and missed alerts**. Use it carefully at your own risk only when necessary.

By default items for log monitoring follow all new lines appearing in the log files. However, there are applications which in some situations start writing an enormous number of messages in their log files. For example, if a database or a DNS server is unavailable, such applications flood log files with thousands of nearly identical error messages until normal operation is restored. By default, all those messages will be dutifully analyzed and matching lines sent to server as configured in `log` and `logrt` items.

Built-in protection against overload consists of a configurable 'maxlines' parameter (protects server from too many incoming matching log lines) and a 4*'maxlines' limit (protects host CPU and I/O from overloading by agent in one check). Still, there are 2 problems with the built-in protection. First, a large number of potentially not-so-informative messages are reported to server and consume space in the database. Second, due to the limited number of lines analyzed per second the agent may lag behind the newest log records for hours. Quite likely, you might prefer to be sooner informed about the current situation in the log files instead of crawling through old records for hours.

The solution to both problems is using the 'maxdelay' parameter. If 'maxdelay' > 0 is specified, during each check the number of processed bytes, the number of remaining bytes and processing time is measured. From these numbers the agent calculates an estimated delay - how many seconds it would take to analyze all remaining records in a log file.

If the delay does not exceed 'maxdelay' then the agent proceeds with analyzing the log file as usual.

If the delay is greater than 'maxdelay' then the agent **ignores a chunk of a log file by "jumping" over it** to a new estimated position so that the remaining lines could be analyzed within 'maxdelay' seconds.

Note that agent does not even read ignored lines into buffer, but calculates an approximate position to jump to in a file.

The fact of skipping log file lines is logged in the agent log file like this:

```
14287:20160602:174344.206
item:"logrt["/home/zabbix32/test[0-9].log",ERROR,,1000,,,120.0]"
logfile:"/home/zabbix32/test1.log" skipping 679858 bytes
(from byte 75653115 to byte 76332973) to meet maxdelay
```

The "to byte" number is approximate because after the "jump" the agent adjusts the position in the file to the beginning of a log line which may be further in the file or earlier.

Depending on how the speed of growing compares with the speed of analyzing the log file you may see no "jumps", rare or often "jumps", large or small "jumps", or even a small "jump" in every check. Fluctuations in the system load and network latency also affect the calculation of delay and hence, "jumping" ahead to keep up with the "maxdelay" parameter.

Setting 'maxdelay' < 'update interval' is not recommended (it may result in frequent small “jumps”).

Notes on handling 'copytruncate' log file rotation

`logrt` with the `copytruncate` option assumes that different log files have different records (at least their timestamps are different), therefore MD5 sums of initial blocks (up to the first 512 bytes) will be different. Two files with the same MD5 sums of initial blocks means that one of them is the original, another - a copy.

`logrt` with the `copytruncate` option makes effort to correctly process log file copies without reporting duplicates. However, things like producing multiple log file copies with the same timestamp, log file rotation more often than logrt[] item update interval, frequent restarting of agent are not recommended. The agent tries to handle all these situations reasonably well, but good results cannot be guaranteed in all circumstances.

Actions if communication fails between agent and server

Each matching line from `log[]` and `logrt[]` item and a result of each `log.count[]` and `logrt.count[]` item check requires a free slot in the designated 50% area in the agent send buffer. The buffer elements are regularly sent to server (or proxy) and the buffer slots are free again.

While there are free slots in the designated log area in the agent send buffer and communication fails between agent and server (or proxy) the log monitoring results are accumulated in the send buffer. This helps to mitigate short communication failures.

During longer communication failures all log slots get occupied and the following actions are taken:

- `log[]` and `logrt[]` item checks are stopped. When communication is restored and free slots in the buffer are available the checks are resumed from the previous position. No matching lines are lost, they are just reported later.
- `log.count[]` and `logrt.count[]` checks are stopped if `maxdelay = 0` (default). Behaviour is similar to `log[]` and `logrt[]` items as described above. Note that this can affect `log.count[]` and `logrt.count[]` results: for example, one check counts 100 matching lines in a log file, but as there are no free slots in the buffer the check is stopped. When communication is restored the agent counts the same 100 matching lines and also 70 new matching lines. The agent now sends count = 170 as if they were found in one check.
- `log.count[]` and `logrt.count[]` checks with `maxdelay > 0`: if there was no “jump” during the check, then behaviour is similar to described above. If a “jump” over log file lines took place then the position after “jump” is kept and the counted result is discarded. So, the agent tries to keep up with a growing log file even in case of communication failure.