Advanced log monitoring

Giedrius Stasiulionis

Logs

One of the cornerstones in monitoring field

Logs provide visibility into the day-to-day operations of IT systems and applications, helping ensure smooth functioning

Zabbix & logs

log[]

log.count[]

custom items

Zabbix & logs – custom items

When to use it?

Only when you can’t achieve desired result with log[] or log.count[]

Zabbix & logs – custom items

- monitoring rapidly updated files (600k+ lines per minute)
- something that you would collect only to use for calculated items
- multi-line monitoring
- monitoring logs as “Passive item”

Zabbix & logs – custom items

How?

Mimic the agent in simplified way

You must be able to read the log files in similar way as agent does

Zabbix & logs – custom items

Pareto principle (80/20 rule)

Concept: key points

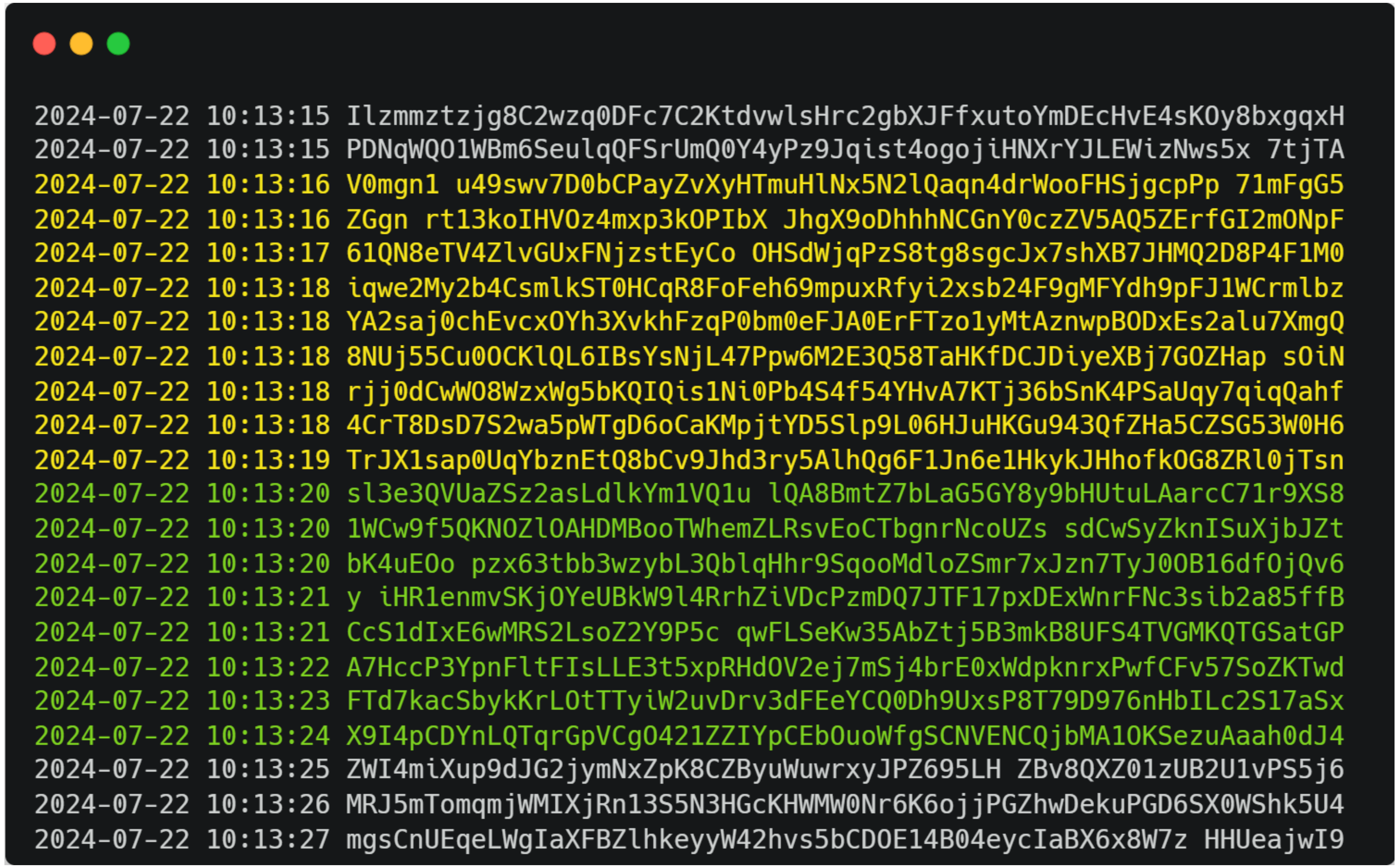
- read log portion-by-portion, just like log[] and log.count[]
- make it fast – operate in bytes instead of lines
- don’t forget log rotation
- choose whether to read or ignore unread data if agent was stopped or item was disabled

Concept: reading

During each run you have to operate with 2 numbers:

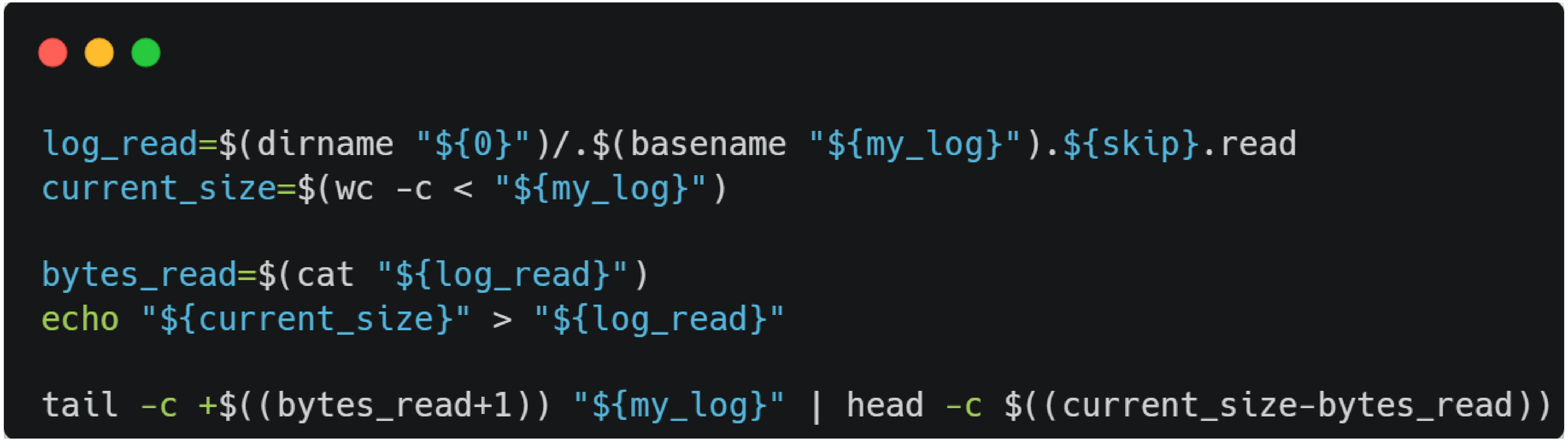
1) how much data was read during last run
2) how much data you have now

Concept: reading



```
2024-07-22 10:13:15 Ilzmmztzjg8C2wzq0DFc7C2KtdvwlsHrc2gbXJFfxutoYmDEcHvE4sKOy8bxgqxH
2024-07-22 10:13:15 PDNqWQO1WBm6SeulqQFSrUmQ0Y4yPz9Jqist4ogojiHNXrYJLEWizNws5x 7tjTA
2024-07-22 10:13:16 V0mgn1 u49swv7D0bCPayZvXyHTmuHlNx5N2lQaqn4drWooFHSjgcpPp 71mFgG5
2024-07-22 10:13:16 ZGgn rt13koIHVOz4mxp3kOPIbX JhgX9oDhhhNCGnY0czZV5AQ5ZErfGI2mONpF
2024-07-22 10:13:17 61QN8eTV4ZlvGUxFNjzstEyCo OHSdWjqPzS8tg8sgcJx7shXB7JHMQ2D8P4F1M0
2024-07-22 10:13:18 iqwe2My2b4CsmlkST0HCqR8FoFeh69mpuxRfyi2xsb24F9gMFYdh9pFJ1WCrmlbz
2024-07-22 10:13:18 YA2saj0chEvcxOYh3XvkhFzqP0bm0eFJA0ErFTzo1yMtAznwpBODxEs2alu7XmgQ
2024-07-22 10:13:18 8NUj55Cu0OCKlQL6IBsYsNjL47Ppw6M2E3Q58TaHKfDCJDiyeXBj7GOZHap sOiN
2024-07-22 10:13:18 rjj0dCwWO8WzxWg5bKQIQis1Ni0Pb4S4f54YHvA7KTj36bSnK4PSaUqy7qiqQahf
2024-07-22 10:13:18 4CrT8DsD7S2wa5pWTgD6oCaKMpjtYD5Slp9L06HJuHKGu943QfZHa5CZSG53W0H6
2024-07-22 10:13:19 TrJX1sap0UqYbznEtQ8bCv9Jhd3ry5AlhQg6F1Jn6e1HkykJHhofkOG8ZRl0jTsn
2024-07-22 10:13:20 sl3e3QVUaZSz2asLdlkYm1VQ1u lQA8BmtZ7bLaG5GY8y9bHUtuLAarcC71r9XS8
2024-07-22 10:13:20 1WCw9f5QKNOZlOAHDMBooTWhemZLRsvEoCTbgnrNcoUZs sdCwSyZknISuXjbJZt
2024-07-22 10:13:20 bK4uEOo pzx63tbb3wzybL3QblqHhr9SqooMdloZSmr7xJzn7TyJ0OB16dfOjQv6
2024-07-22 10:13:21 y iHR1enmvSKjOYeUBkW9l4RrhZiVDcPzmDQ7JTF17pxDExWnrFNc3sib2a85ffB
2024-07-22 10:13:21 CcS1dIxE6wMRS2LsoZ2Y9P5c qwFLSeKw35AbZtj5B3mkB8UFS4TVGMKQTGSatGP
2024-07-22 10:13:22 A7HccP3YpnFltFIsLLE3t5xpRHdOV2ej7mSj4brE0xWdpknrxPwfCFv57SoZKTwd
2024-07-22 10:13:23 FTd7kacSbykKrLOtTTyiW2uvDrv3dFEeYCQ0Dh9UxsP8T79D976nHbILc2S17aSx
2024-07-22 10:13:24 X9I4pCDYnLQTqrGpVCgO421ZZIYpCEbOuoWfgSCNVENCQjbMA1OKSezuAaah0dJ4
2024-07-22 10:13:25 ZWI4miXup9dJG2jymNxZpK8CZByuWuwrxyJPZ695LH ZBv8QXZ01zUB2U1vPS5j6
2024-07-22 10:13:26 MRJ5mTomqmjWMIXjRn13S5N3HGcKHWMW0Nr6K6ojjPGZhwDekuPGD6SX0WShk5U4
2024-07-22 10:13:27 mgsCnUEqeLWgIaXFBZlhkeyyW42hvs5bCDOE14B04eycIaBX6x8W7z HHUeajwI9
```

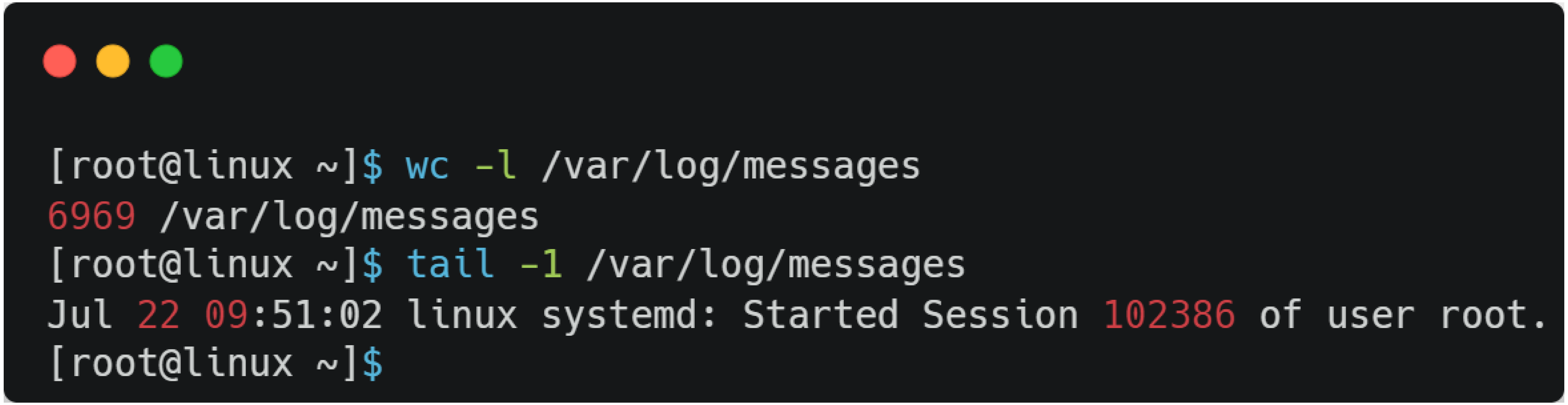
Concept: reading



```
log_read=$(dirname "${0}")/.$(basename "${my_log}").${skip}.read
current_size=$(wc -c < "${my_log}")

bytes_read=$(cat "${log_read}")
echo "${current_size}" > "${log_read}"

tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read))
```

Concept: reading in bytes

Reading in lines is more understandable for human eye



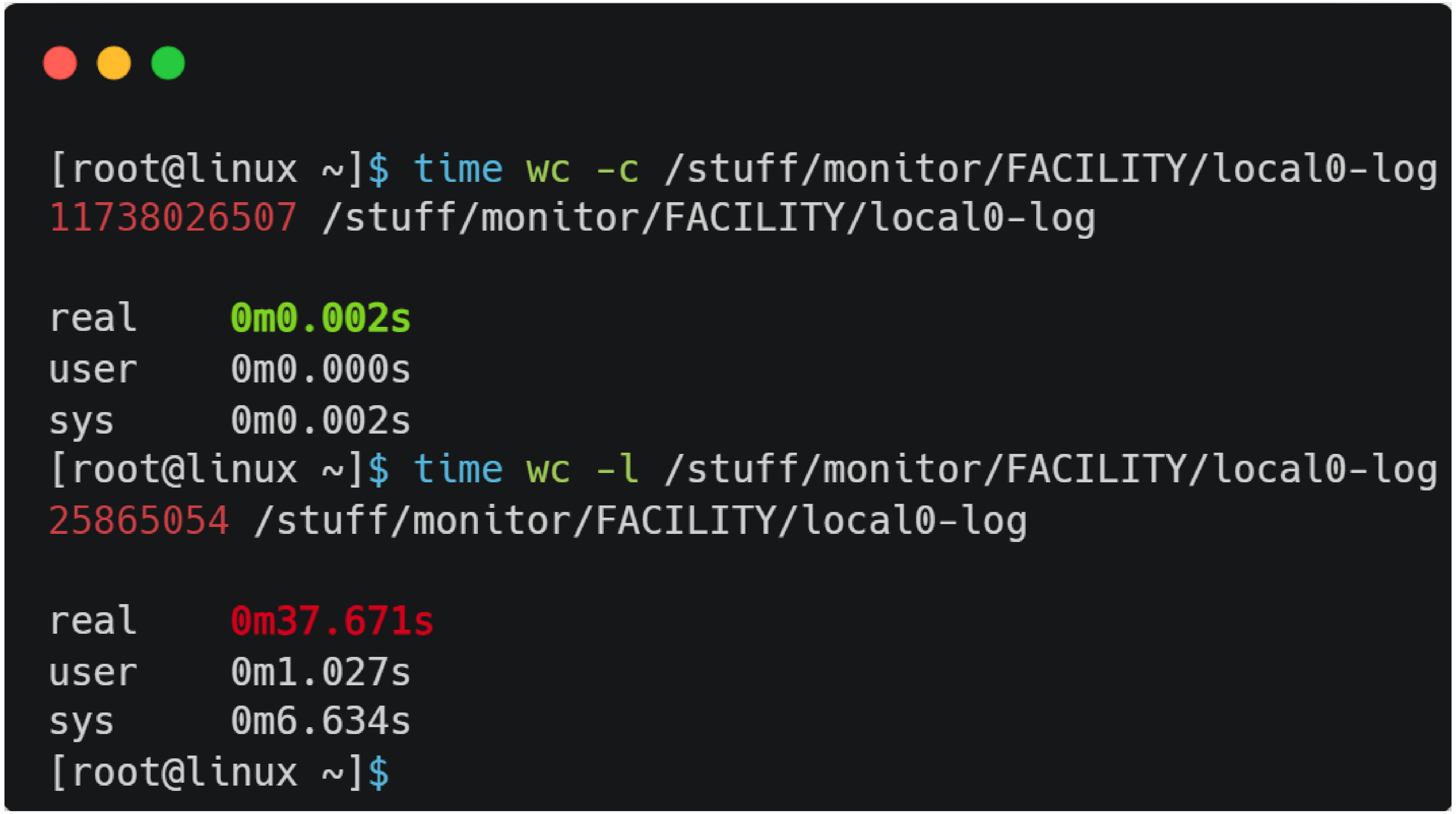
```
[root@linux ~]$ wc -l /var/log/messages
6969 /var/log/messages
[root@linux ~]$ tail -1 /var/log/messages
Jul 22 09:51:02 linux systemd: Started Session 102386 of user root.
[root@linux ~]$
```

Concept: reading in bytes

But reading in lines is very slow when you have big files!

Reading in bytes is crucial for performance

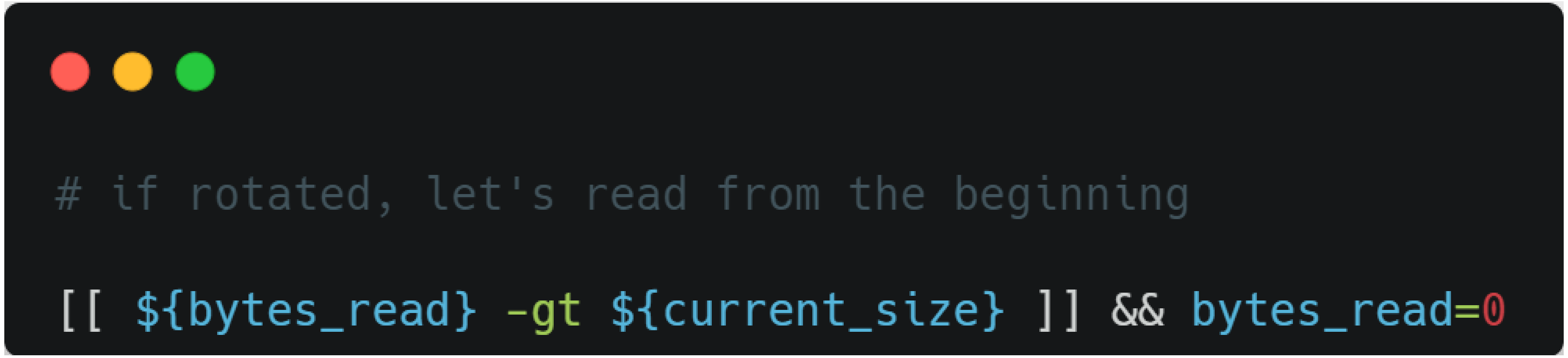
Concept: reading in bytes



```
[root@linux ~]$ time wc -c /stuff/monitor/FACILITY/local0-log
11738026507 /stuff/monitor/FACILITY/local0-log

real	0m0.002s
user	0m0.000s
sys	0m0.002s
[root@linux ~]$ time wc -l /stuff/monitor/FACILITY/local0-log
25865054 /stuff/monitor/FACILITY/local0-log

real	0m37.671s
user	0m1.027s
sys	0m6.634s
[root@linux ~]$
```

Concept: log rotation

Don’t forget log rotation – when log file becomes smaller than it was during last run, it means it’s rotated



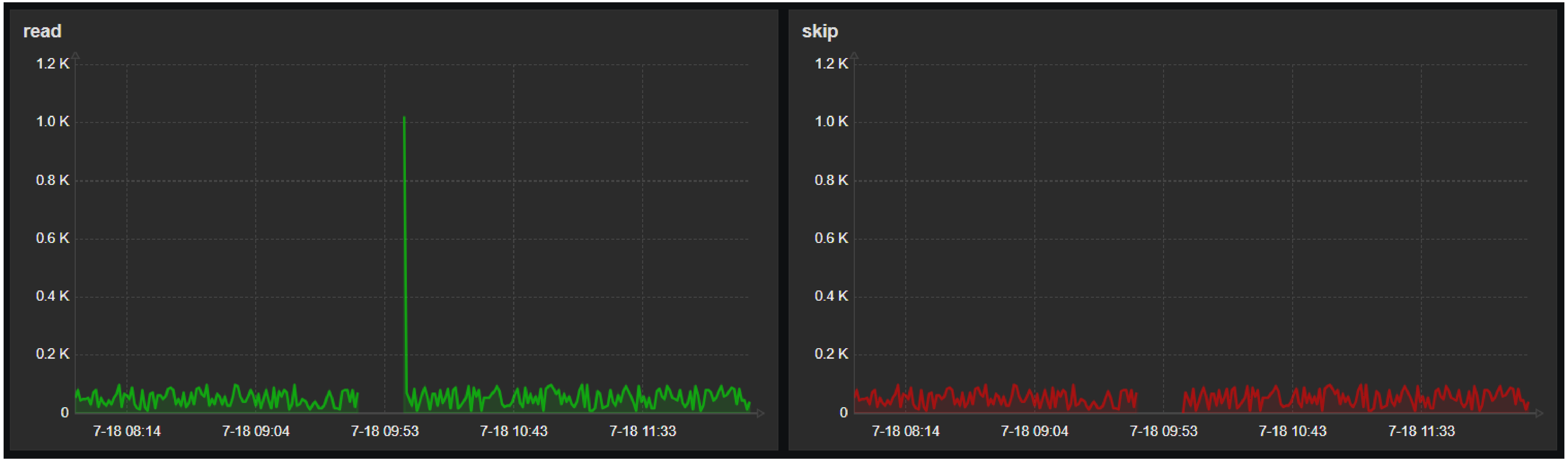
```
# if rotated, let's read from the beginning

[[ ${bytes_read} -gt ${current_size} ]] && bytes_read=0
```

Concept: read vs. skip

If you stop agent for a while, you have to choose what to do when it’s started again: read everything or skip?

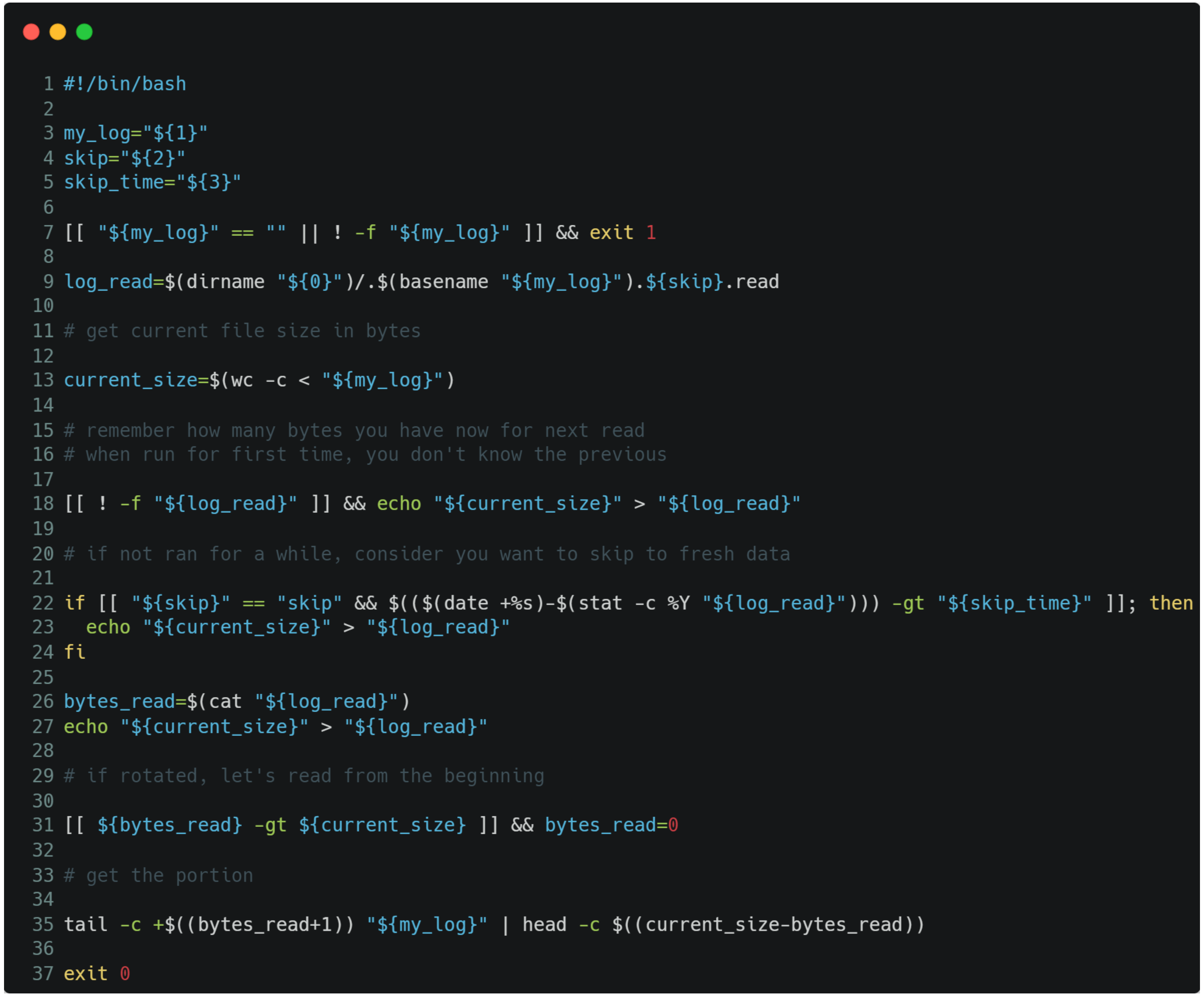
Similar to “maxdelay” setting for log[] / log.count[]

Concept: read vs. skip



Implementation

So, after putting those four ideas into one script, we have a working frame already



```bash
#!/bin/bash

my_log="${1}"
skip="${2}"
skip_time="${3}"

[[ "${my_log}" == "" || ! -f "${my_log}" ]] && exit 1

log_read=$(dirname "${0}")/.$(basename "${my_log}").${skip}.read

# get current file size in bytes

current_size=$(wc -c < "${my_log}")

# remember how many bytes you have now for next read
# when run for first time, you don't know the previous

[[ ! -f "${log_read}" ]] && echo "${current_size}" > "${log_read}"

# if not ran for a while, consider you want to skip to fresh data

if [[ "${skip}" == "skip" && $(($(date +%s)-$(stat -c %Y "${log_read}"))) -gt "${skip_time}" ]]; then
  echo "${current_size}" > "${log_read}"
fi

bytes_read=$(cat "${log_read}")
echo "${current_size}" > "${log_read}"

# if rotated, let's read from the beginning

[[ ${bytes_read} -gt ${current_size} ]] && bytes_read=0

# get the portion

tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read))

exit 0
```

Implementation

It would just read data and output fresh lines without any further processing

What you do next with this ability to collect data depends on your needs and imagination

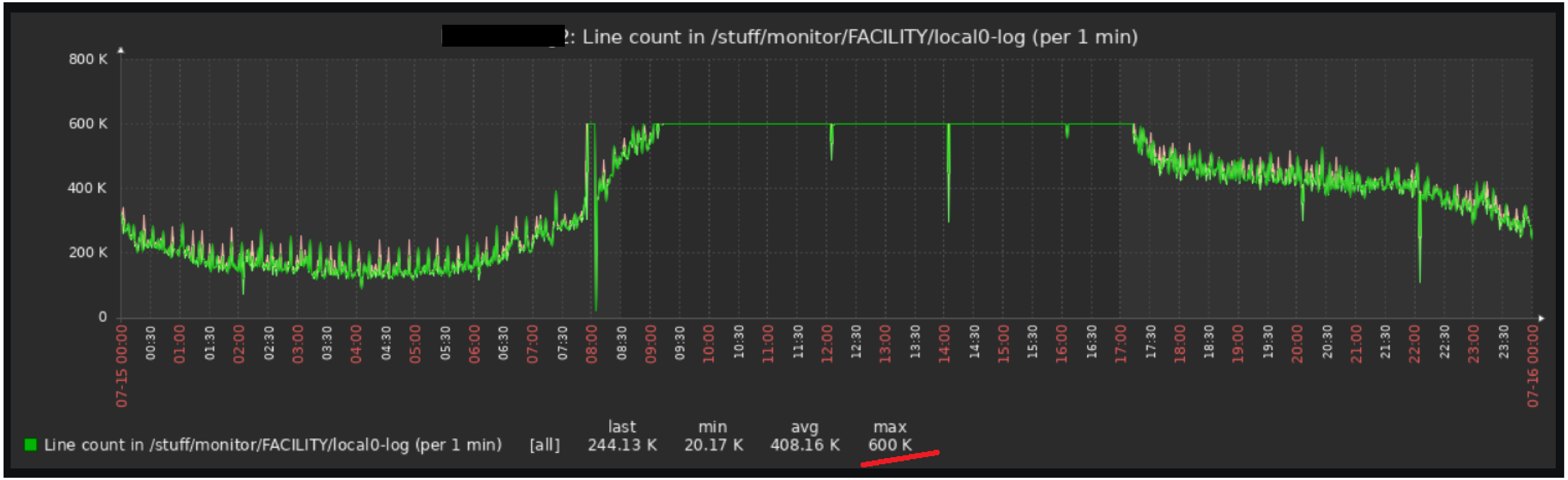
Time to see it in action!

Example no. 1

Limit of 600k lines per minute

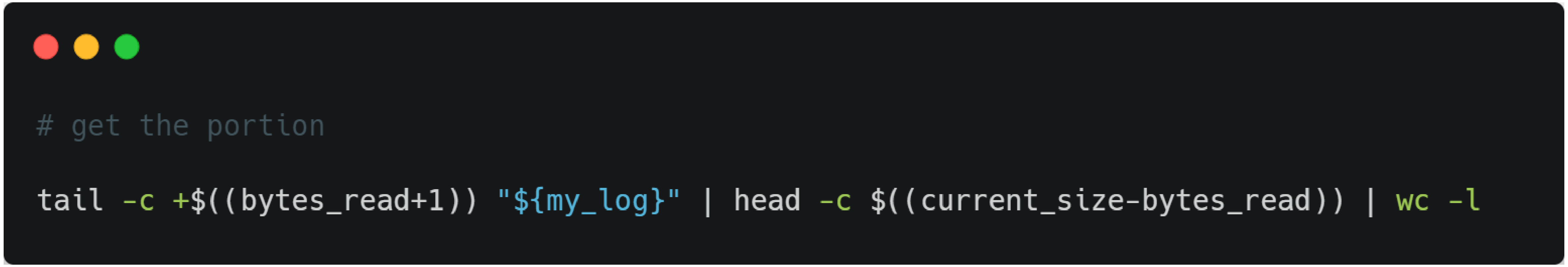
Both log[] and log.count[] share same limitation of 600k lines to be analyzed per minute

Example no. 1



		last	min	avg	max
🟩 Line count in /stuff/monitor/FACILITY/local0-log (per 1 min)	[all]	244.13 K	20.17 K	408.16 K	600 K

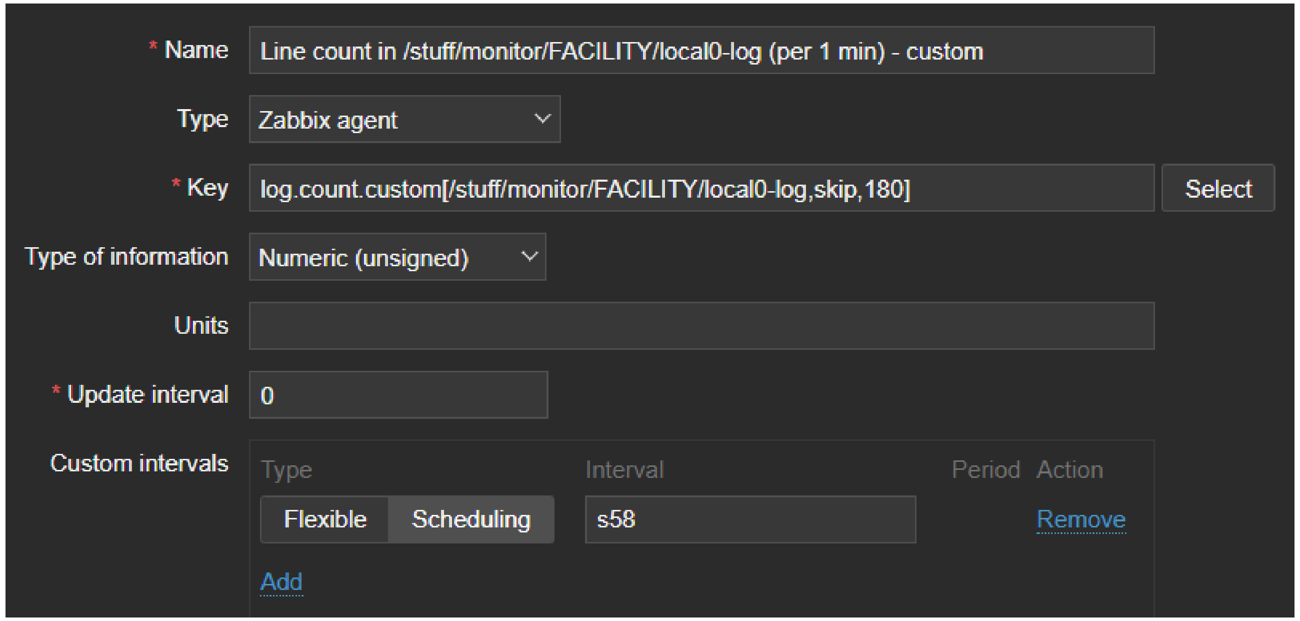
Example no. 1



```bash
# get the portion

tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read)) | wc -l
```

Example no. 1: configuration

```
UserParameter=log.count.custom[*],/etc/zabbix/zabbix_agentd.d/zbx_scripts/log_count.sh "$1" "$2" "$3"
```

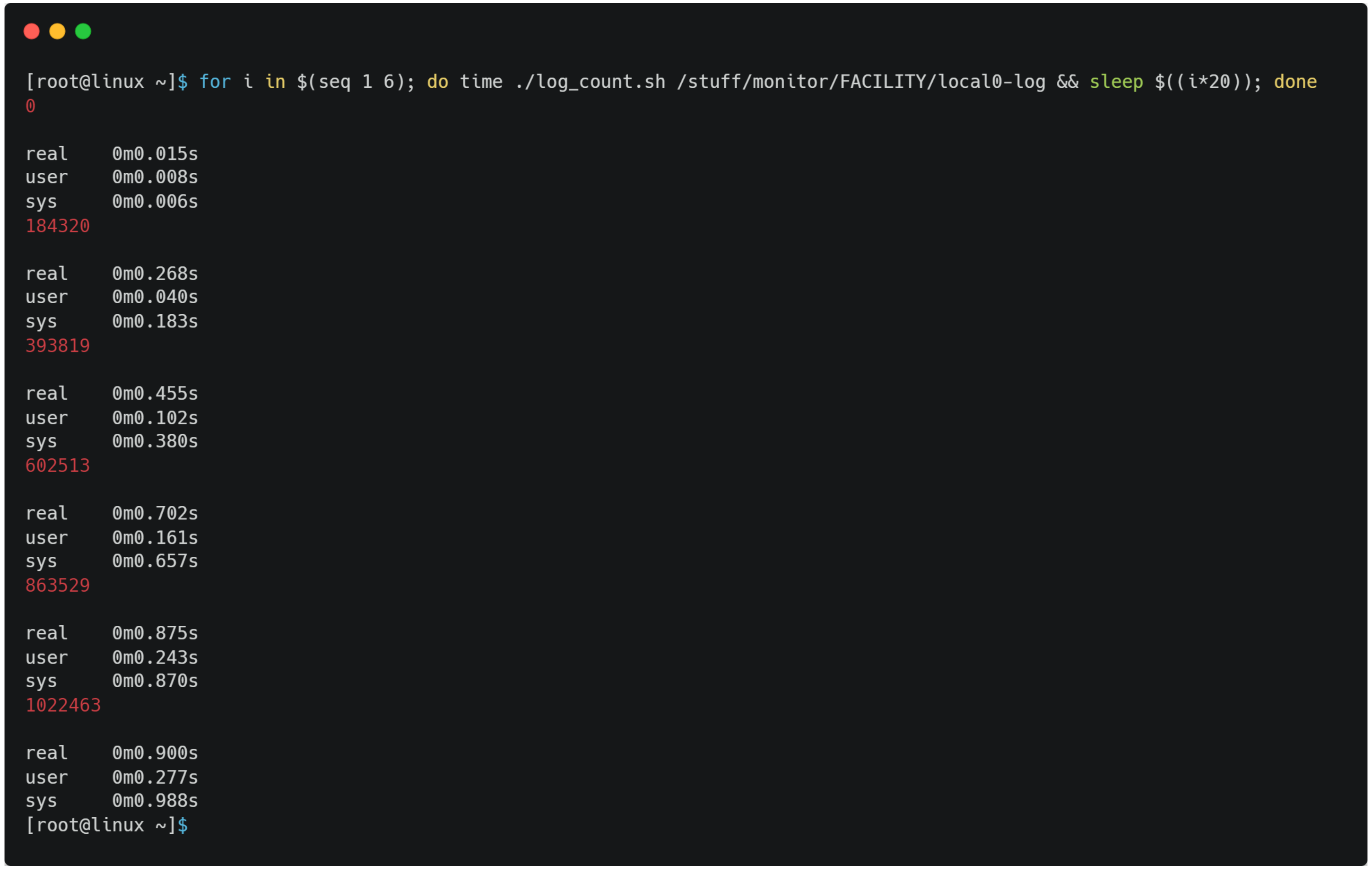


* Name	Line count in /stuff/monitor/FACILITY/local0-log (per 1 min) - custom
Type	Zabbix agent
* Key	log.count.custom[/stuff/monitor/FACILITY/local0-log,skip,180]
Type of information	Numeric (unsigned)
Units	
* Update interval	0

Custom intervals

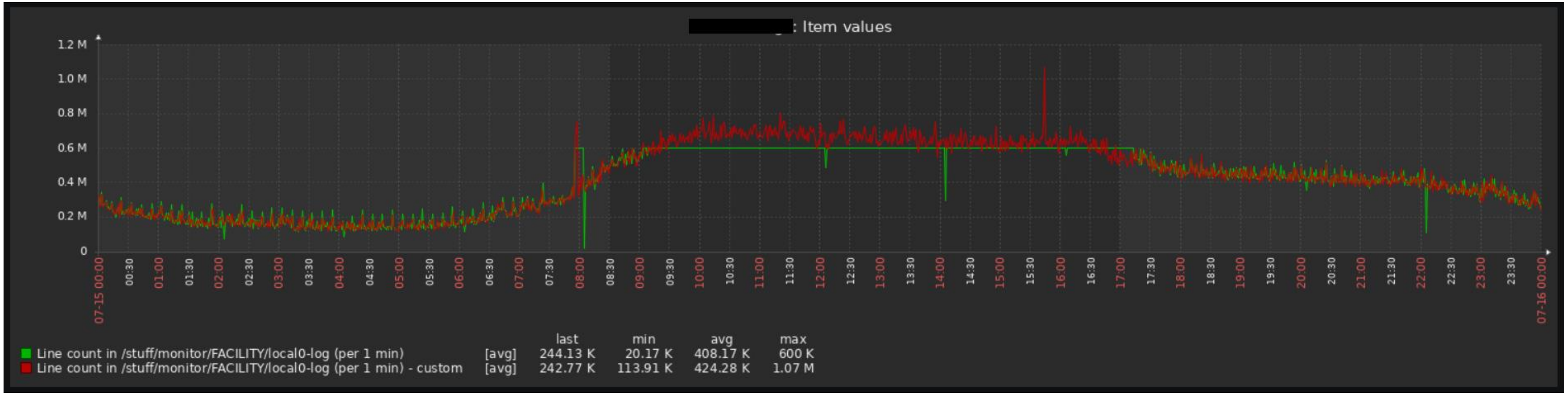
Type	Interval	Period	Action
Flexible [✓ Scheduling]	s58		Remove

Example no. 1: performance



```
[root@linux ~]$ for i in $(seq 1 6); do time ./log_count.sh /stuff/monitor/FACILITY/local0-log && sleep $((i*20)); done
0

real	0m0.015s
user	0m0.008s
sys	0m0.006s
184320

real	0m0.268s
user	0m0.040s
sys	0m0.183s
393819

real	0m0.455s
user	0m0.102s
sys	0m0.380s
602513

real	0m0.702s
user	0m0.161s
sys	0m0.657s
863529

real	0m0.875s
user	0m0.243s
sys	0m0.870s
1022463

real	0m0.900s
user	0m0.277s
sys	0m0.988s
[root@linux ~]$
```

Example no. 1: result



		last	min	avg	max
🟩 Line count in /stuff/monitor/FACILITY/local0-log (per 1 min)	[avg]	244.13 K	20.17 K	408.17 K	600 K
🟥 Line count in /stuff/monitor/FACILITY/local0-log (per 1 min) - custom	[avg]	242.77 K	113.91 K	424.28 K	1.07 M

Example no. 2

Collecting something just for the sake of feeding it to calculated items later on

AND

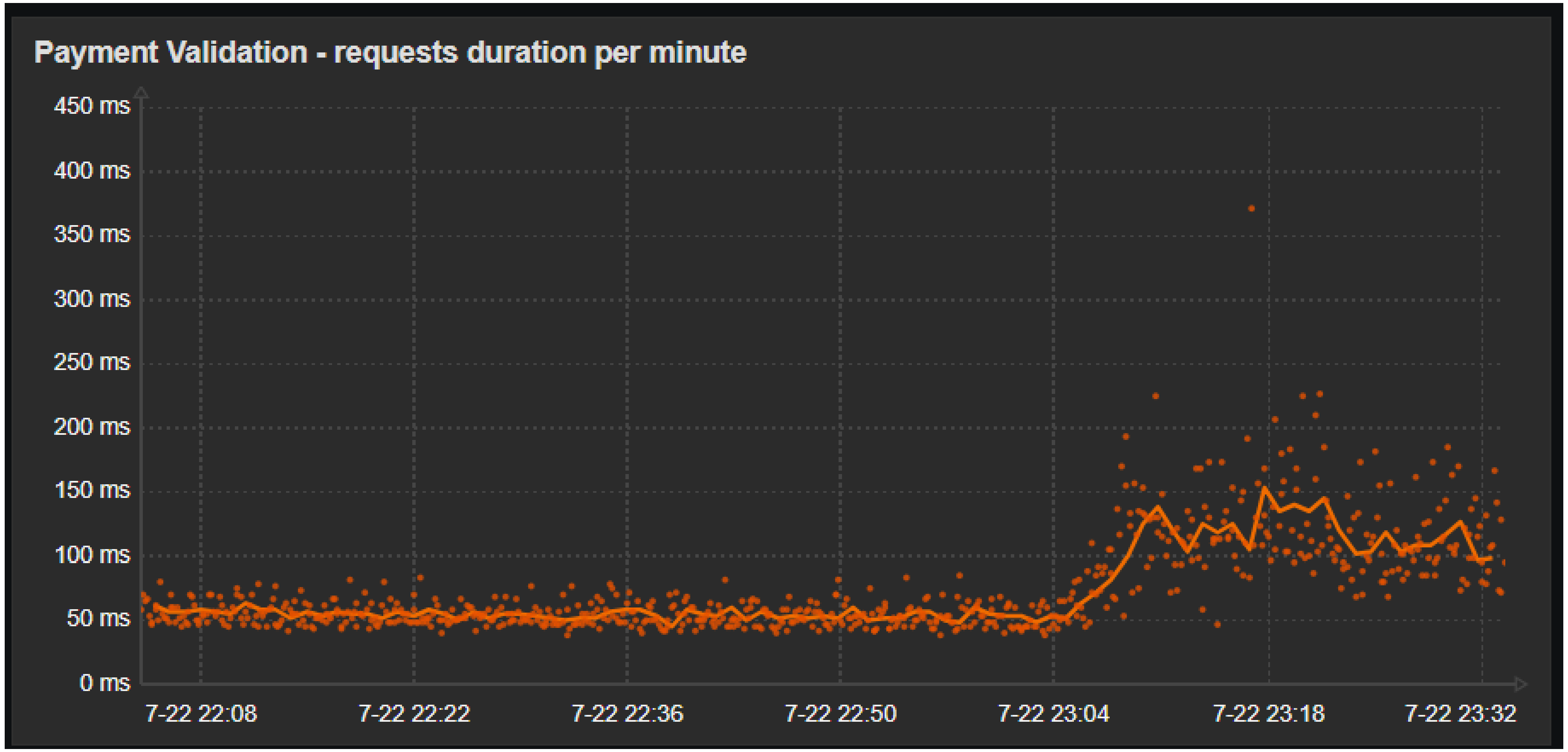
having lots of that “something”

Example no. 2

Each line of log has duration component – say duration of request processing

You want to check, what’s average duration per minute

Example no. 2



Payment Validation - requests duration per minute
450 ms
400 ms
350 ms
300 ms
250 ms
200 ms
150 ms
100 ms
50 ms
0 ms
7-22 22:08
7-22 22:22
7-22 22:36
7-22 22:50
7-22 23:04
7-22 23:18
7-22 23:32

Example no. 2

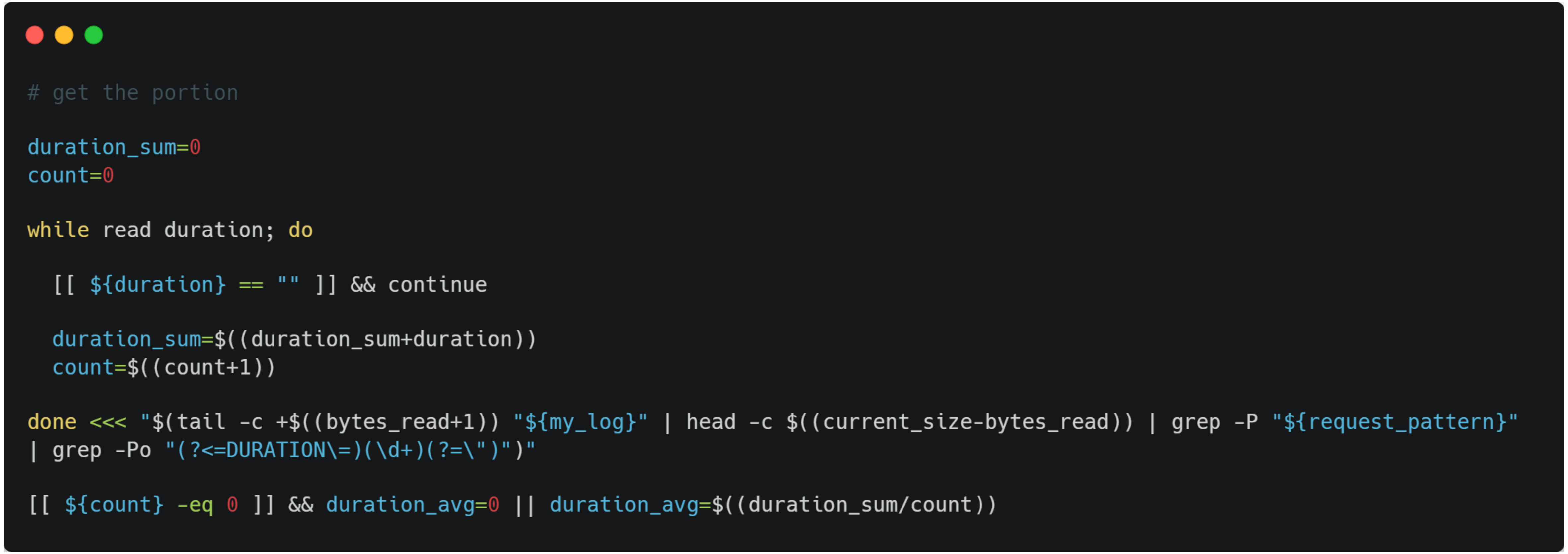
But what to do, when you have 10k+ of such lines per minute?

Storing all of it just for the sake of calculating that one average value is of course inefficient

Displaying 10k+ dots is also not a good idea…

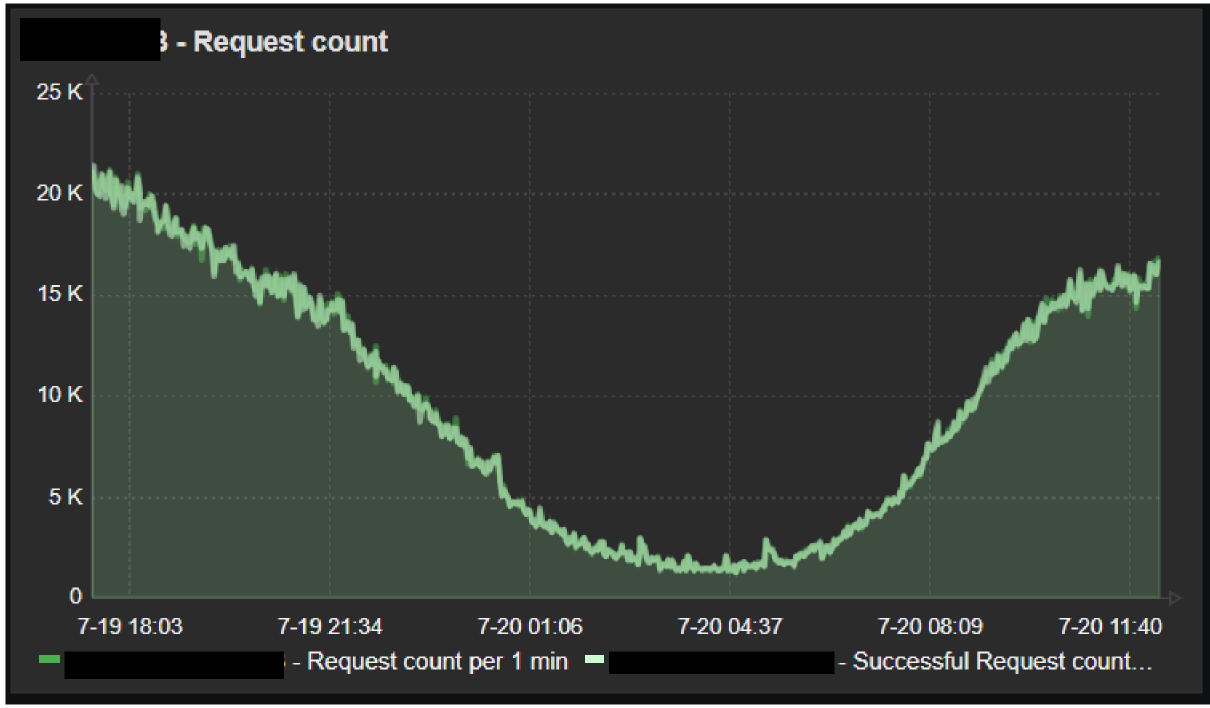
So all you need is just 1 average value per minute

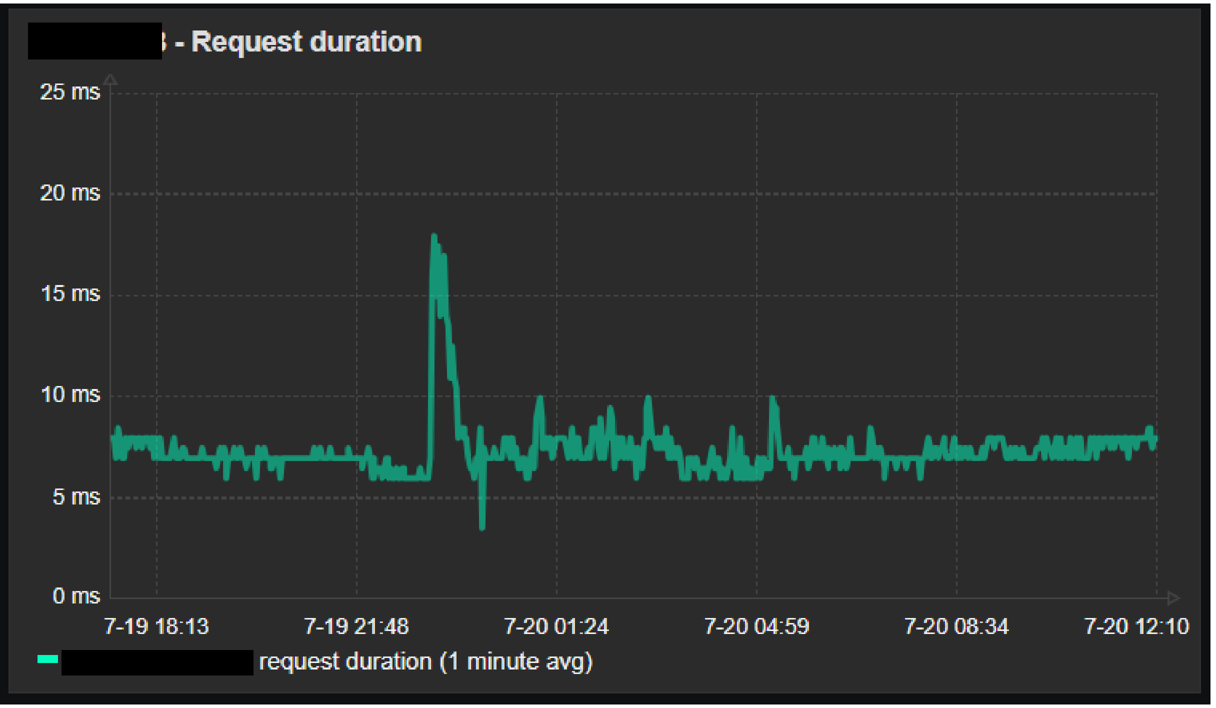
Example no. 2



```
# get the portion

duration_sum=0
count=0

while read duration; do

  [[ ${duration} == "" ]] && continue

  duration_sum=$((duration_sum+duration))
  count=$((count+1))

done <<< "$(tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read)) | grep -P "${request_pattern}"
| grep -Po "(?<=DURATION\=)(\d+)(?=\")")"

[[ ${count} -eq 0 ]] && duration_avg=0 || duration_avg=$((duration_sum/count))
```

Example no. 2: result





Example no. 3

Multi line analysis

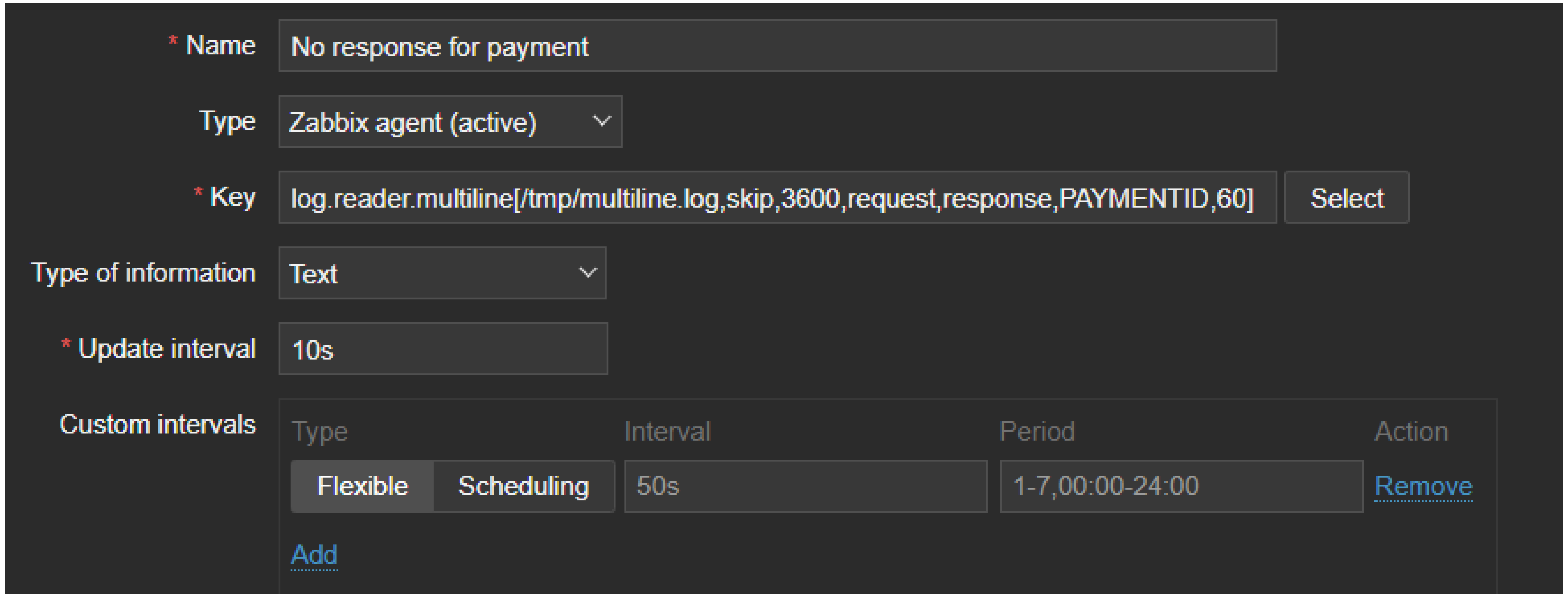
Your interest to find something meaningful might sometimes lay in multiple lines

Example no. 3

Say you have a request with some unique identifier PAYMENTID=… logged as one line and later some answer / response should be logged for the same PAYMENTID

What if response is never logged and you want to know it?

Example no. 3: configuration

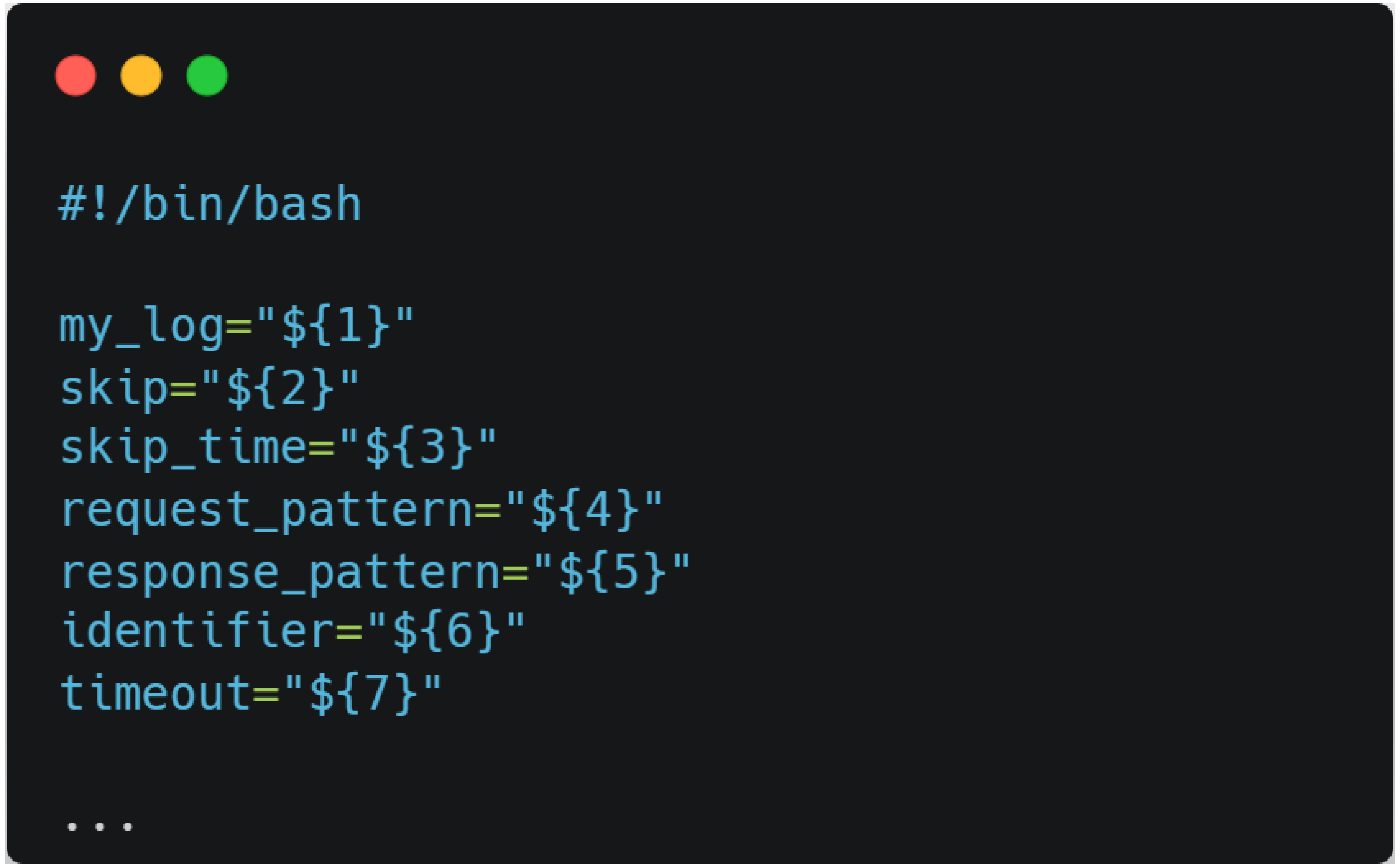


* Name	No response for payment
Type	Zabbix agent (active)
* Key	log.reader.multiline[/tmp/multiline.log,skip,3600,request,response,PAYMENTID,60]
Type of information	Text
* Update interval	10s

Custom intervals

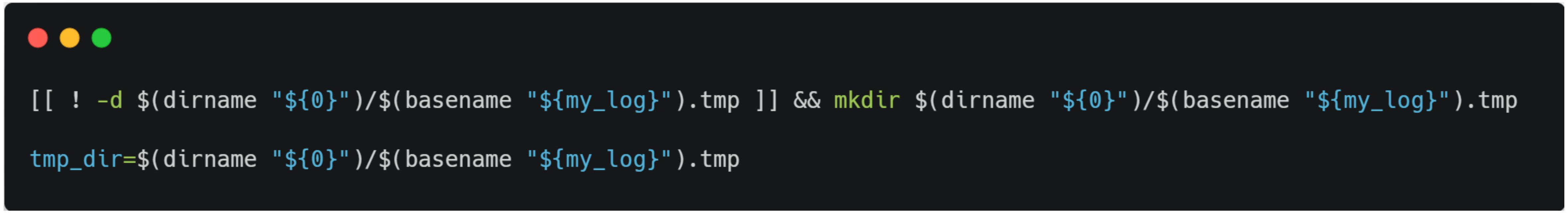
Type	Interval	Period	Action
Flexible [✓] Scheduling	50s	1-7,00:00-24:00	Remove

Example no. 3



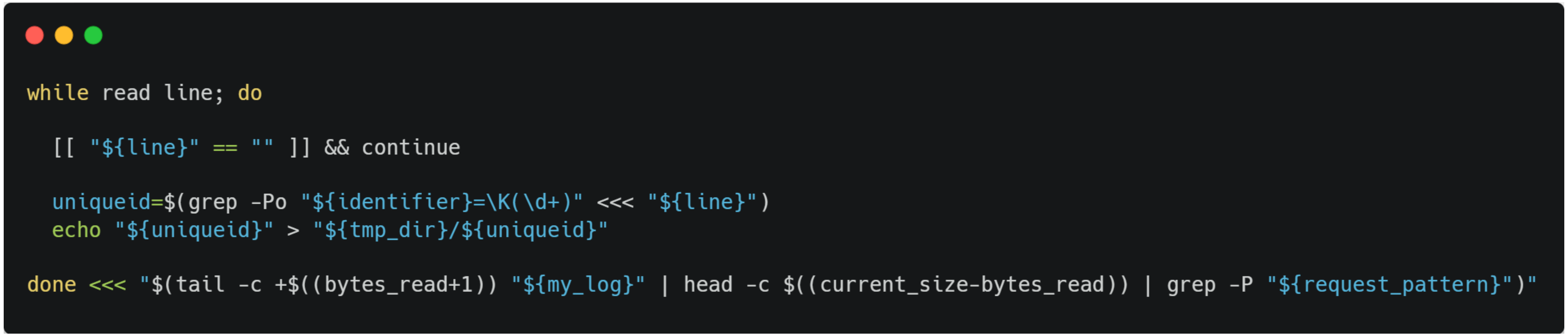
```
#!/bin/bash

my_log="${1}"
skip="${2}"
skip_time="${3}"
request_pattern="${4}"
response_pattern="${5}"
identifier="${6}"
timeout="${7}"

...
```

Example no. 3



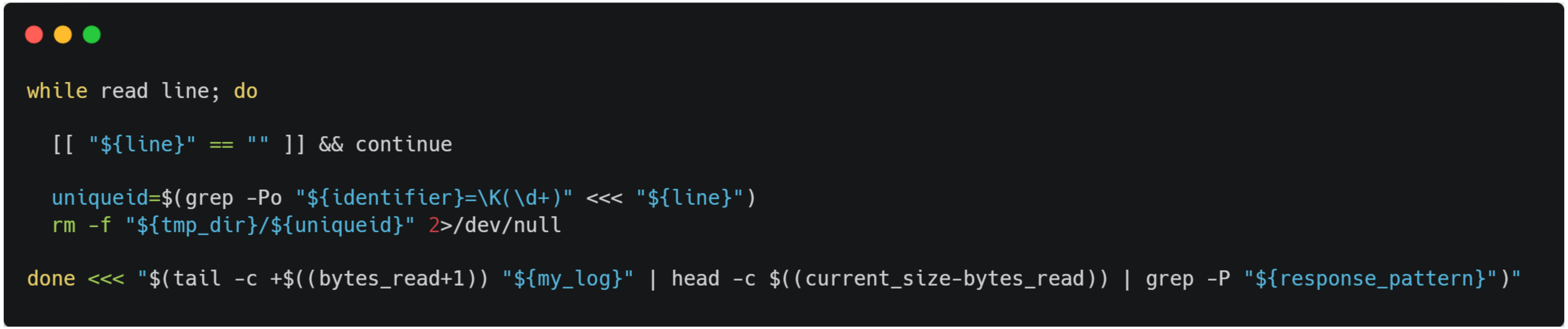
```
[[ ! -d $(dirname "${0}")/$(basename "${my_log}").tmp ]] && mkdir $(dirname "${0}")/$(basename "${my_log}").tmp

tmp_dir=$(dirname "${0}")/$(basename "${my_log}").tmp
```

Example no. 3



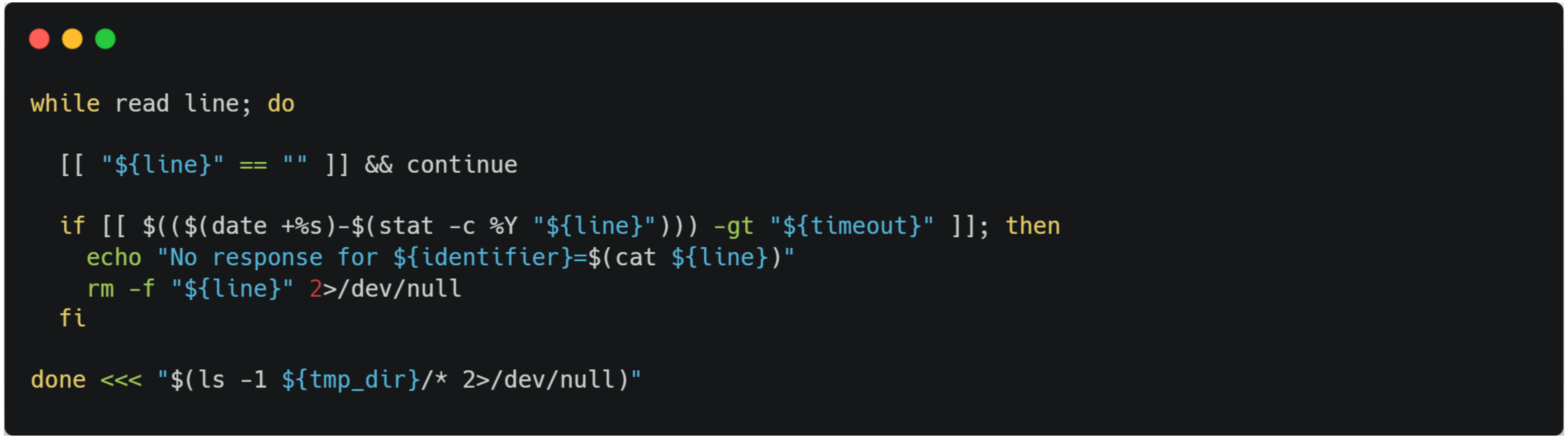
```
while read line; do

  [[ "${line}" == "" ]] && continue

  uniqueid=$(grep -Po "${identifier}=\K(\d+)" <<< "${line}")
  echo "${uniqueid}" > "${tmp_dir}/${uniqueid}"

done <<< "$(tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read)) | grep -P "${request_pattern}")"
```

Example no. 3



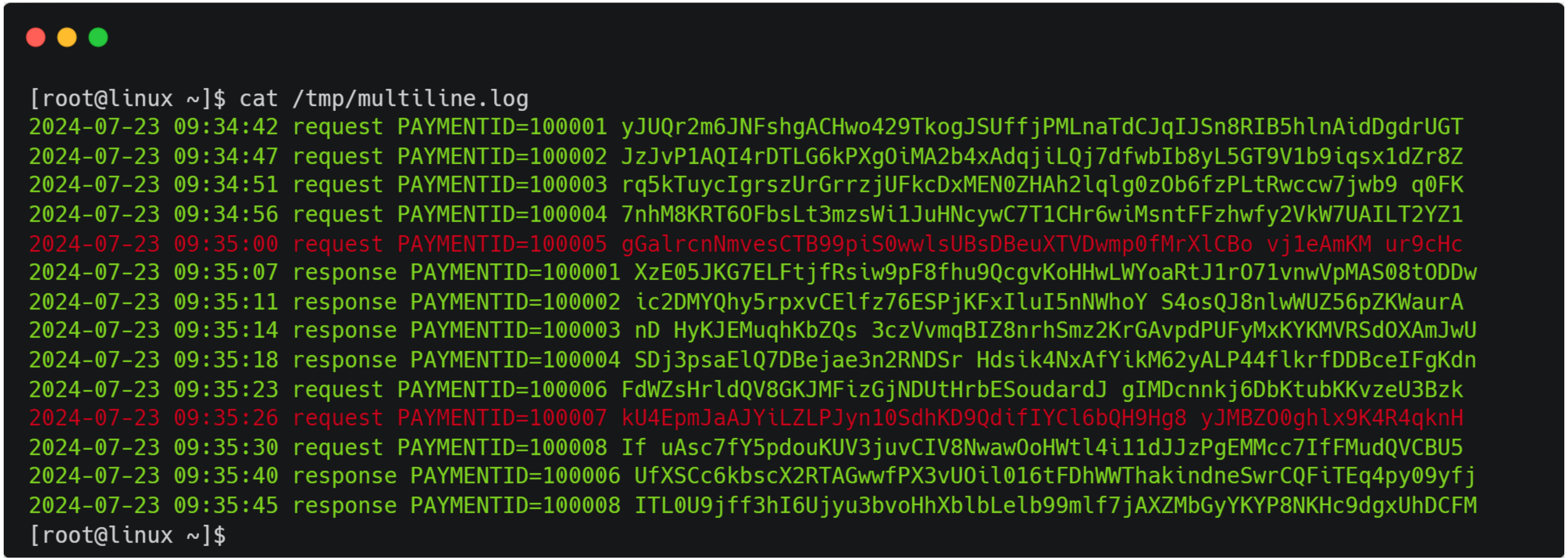
```
while read line; do

  [[ "${line}" == "" ]] && continue

  uniqueid=$(grep -Po "${identifier}=\K(\d+)" <<< "${line}")
  rm -f "${tmp_dir}/${uniqueid}" 2>/dev/null

done <<< "$(tail -c +$((bytes_read+1)) "${my_log}" | head -c $((current_size-bytes_read)) | grep -P "${response_pattern}")"
```

Example no. 3



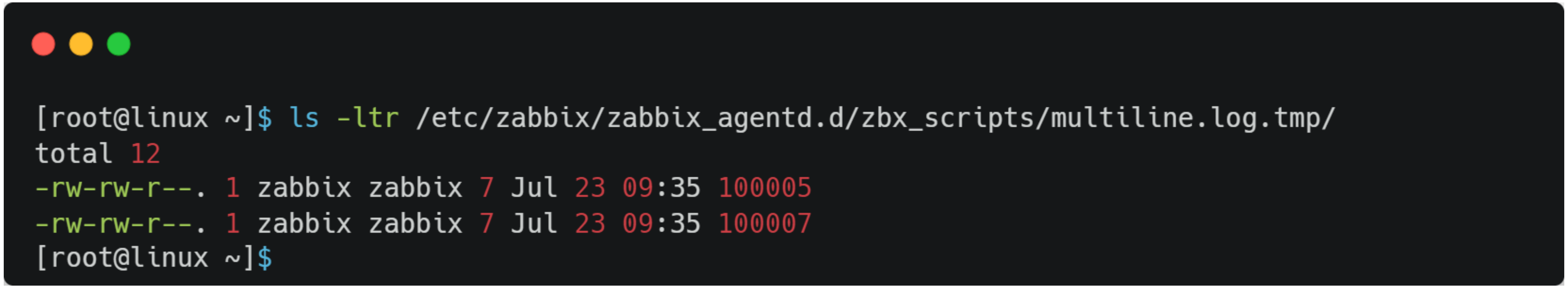
```
while read line; do

  [[ "${line}" == "" ]] && continue

  if [[ $(($(date +%s)-$(stat -c %Y "${line}"))) -gt "${timeout}" ]]; then
    echo "No response for ${identifier}=$(cat ${line})"
    rm -f "${line}" 2>/dev/null
  fi

done <<< "$(ls -1 ${tmp_dir}/* 2>/dev/null)"
```

Example no. 3



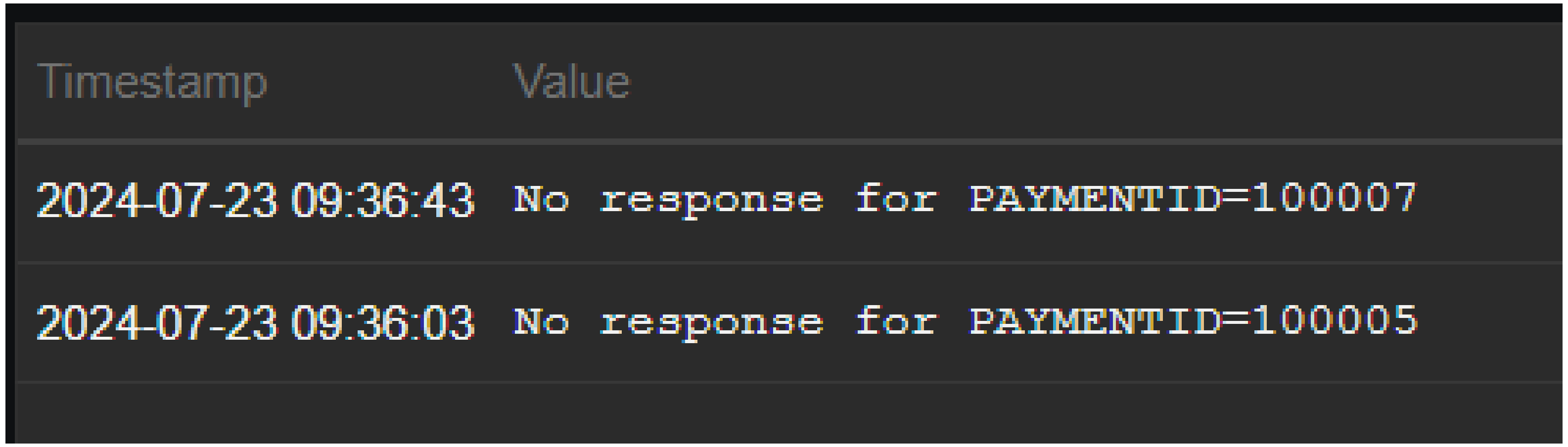
```
[root@linux ~]$ cat /tmp/multiline.log
2024-07-23 09:34:42 request PAYMENTID=100001 yJUQr2m6JNFshgACHwo429TkogJSUffjPMLnaTdCJqIJSn8RIB5hlnAidDgdrUGT
2024-07-23 09:34:47 request PAYMENTID=100002 JzJvP1AQI4rDTLG6kPXgOiMA2b4xAdqjiLQj7dfwbIb8yL5GT9V1b9iqsx1dZr8Z
2024-07-23 09:34:51 request PAYMENTID=100003 rq5kTuycIgrszUrGrrzjUFkcDxMEN0ZHAh2lqlg0zOb6fzPLtRwccw7jwb9 q0FK
2024-07-23 09:34:56 request PAYMENTID=100004 7nhM8KRT6OFbsLt3mzsWi1JuHNcywC7T1CHr6wiMsntFFzhwfy2VkW7UAILT2YZ1
2024-07-23 09:35:00 request PAYMENTID=100005 gGalrcnNmvesCTB99piS0wwlsUBsDBeuXTVDwmp0fMrXlCBo vj1eAmKM ur9cHc
2024-07-23 09:35:07 response PAYMENTID=100001 XzE05JKG7ELFtjfRsiw9pF8fhu9QcgvKoHHwLWYoaRtJ1rO71vnwVpMAS08tODDw
2024-07-23 09:35:11 response PAYMENTID=100002 ic2DMYQhy5rpxvCElfz76ESPjKFxIluI5nNWhoY S4osQJ8nlwWUZ56pZKWaurA
2024-07-23 09:35:14 response PAYMENTID=100003 nD HyKJEMuqhKbZQs 3czVvmqBIZ8nrhSmz2KrGAvpdPUFyMxKYKMVRSdOXAmJwU
2024-07-23 09:35:18 response PAYMENTID=100004 SDj3psaElQ7DBejae3n2RNDSr Hdsik4NxAfYikM62yALP44flkrfDDBceIFgKdn
2024-07-23 09:35:23 request PAYMENTID=100006 FdWZsHrldQV8GKJMFizGjNDUtHrbESoudardJ gIMDcnnkj6DbKtubKKvzeU3Bzk
2024-07-23 09:35:26 request PAYMENTID=100007 kU4EpmJaAJYiLZLPJyn10SdhKD9QdifIYCl6bQH9Hg8 yJMBZO0ghlx9K4R4qknH
2024-07-23 09:35:30 request PAYMENTID=100008 If uAsc7fY5pdouKUV3juvCIV8NwawOoHWtl4i11dJJzPgEMMcc7IfFMudQVCBU5
2024-07-23 09:35:40 response PAYMENTID=100006 UfXSCc6kbscX2RTAGwwfPX3vUOil016tFDhWWThakindneSwrCQFiTEq4py09yfj
2024-07-23 09:35:45 response PAYMENTID=100008 ITL0U9jff3hI6Ujyu3bvoHhXblbLelb99mlf7jAXZMbGyYKYP8NKHc9dgxUhDCFM
[root@linux ~]$
```

Example no. 3



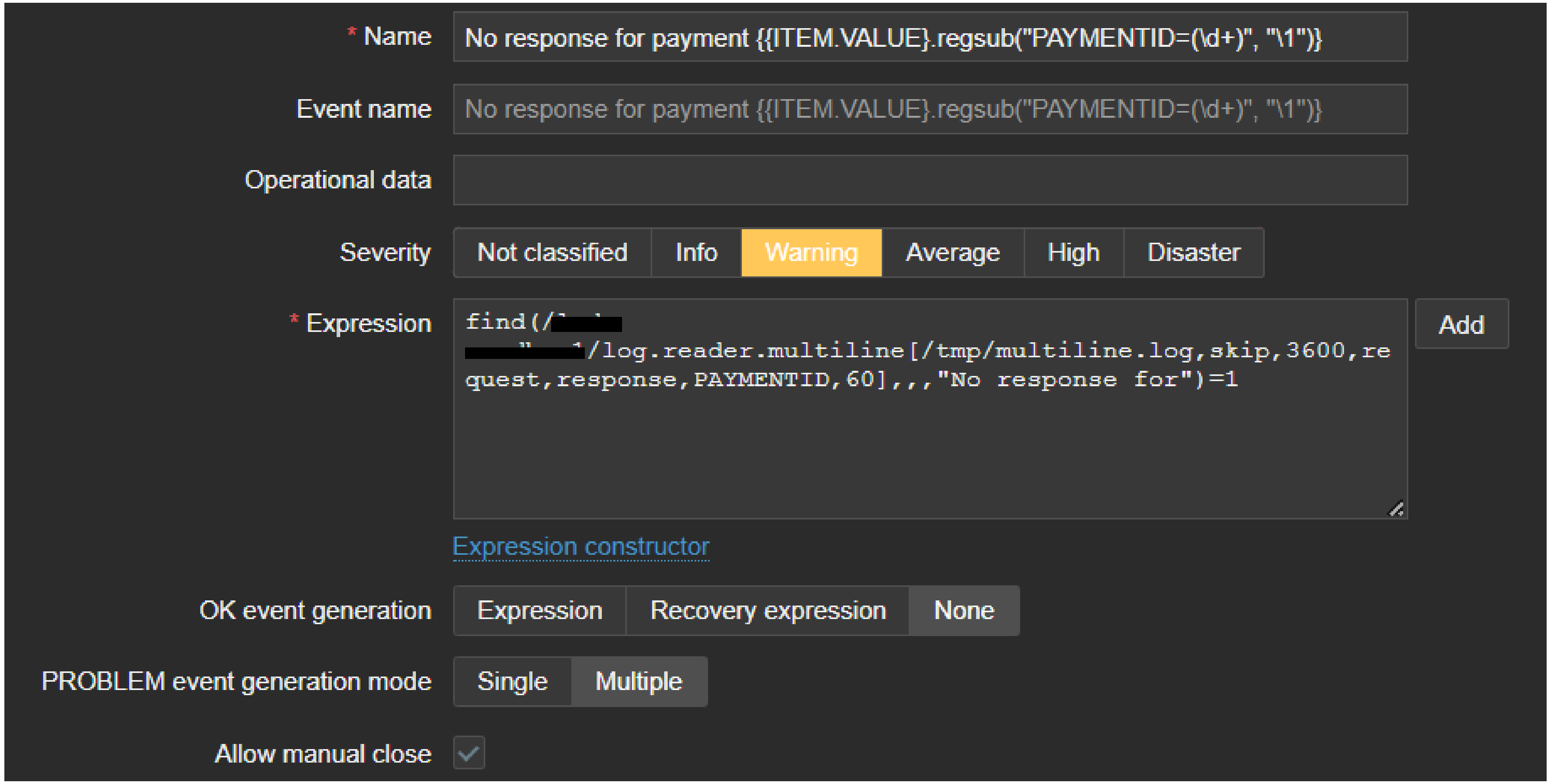
```
[root@linux ~]$ ls -ltr /etc/zabbix/zabbix_agentd.d/zbx_scripts/multiline.log.tmp/
total 12
-rw-rw-r--. 1 zabbix zabbix 7 Jul 23 09:35 100005
-rw-rw-r--. 1 zabbix zabbix 7 Jul 23 09:35 100007
[root@linux ~]$
```

Example no. 3: result



Timestamp	Value
2024-07-23 09:36:43	No response for PAYMENTID=100007
2024-07-23 09:36:03	No response for PAYMENTID=100005

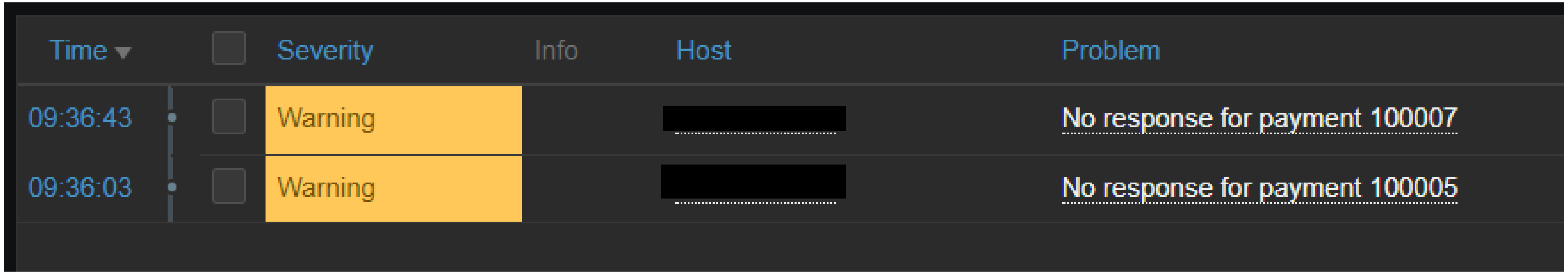
Example no. 3: trigger



* Name	No response for payment {{ITEM.VALUE}.regsub("PAYMENTID=(\d+)", "\1")}
Event name	No response for payment {{ITEM.VALUE}.regsub("PAYMENTID=(\d+)", "\1")}
Operational data	
Severity	Not classified / Info / [✓ Warning] / Average / High / Disaster
* Expression	see below
OK event generation	Expression / Recovery expression / [✓ None]
PROBLEM event generation mode	Single / [✓ Multiple]
Allow manual close	✓

```
find(/[redacted]
[redacted]/log.reader.multiline[/tmp/multiline.log,skip,3600,re
quest,response,PAYMENTID,60],,,"No response for")=1
```

Example no. 3: result



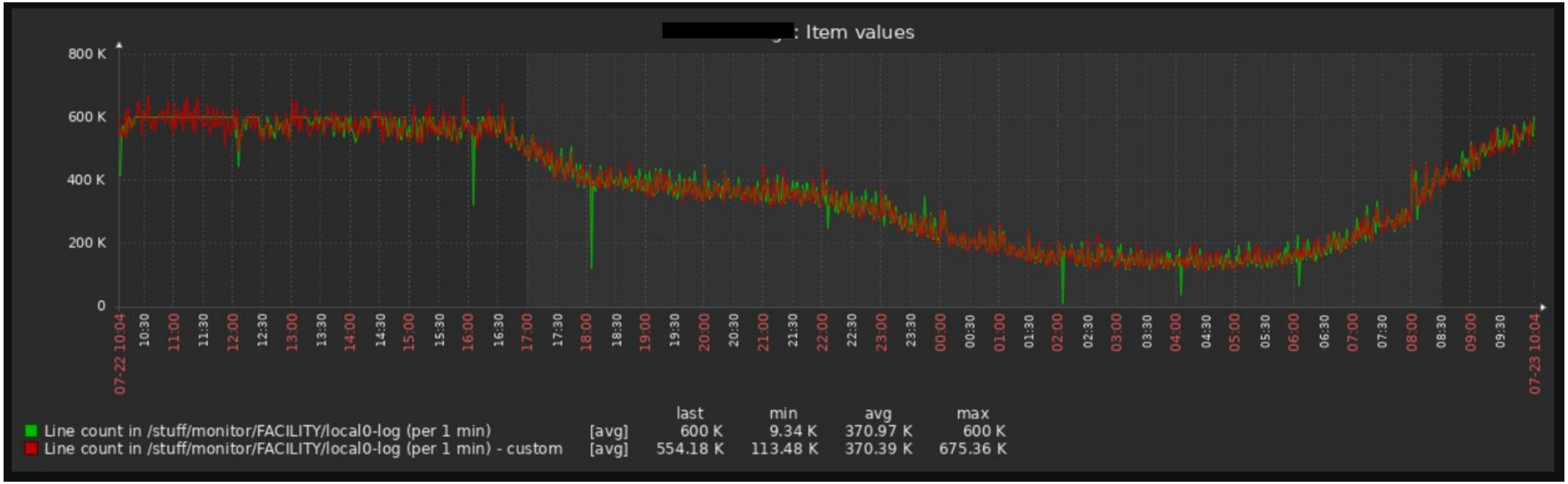
Time ▼		Severity	Info	Host	Problem
09:36:43	☐	Warning		[redacted]	No response for payment 100007
09:36:03	☐	Warning		[redacted]	No response for payment 100005

Run as passive item

With this approach you can analyze logs with passive item type

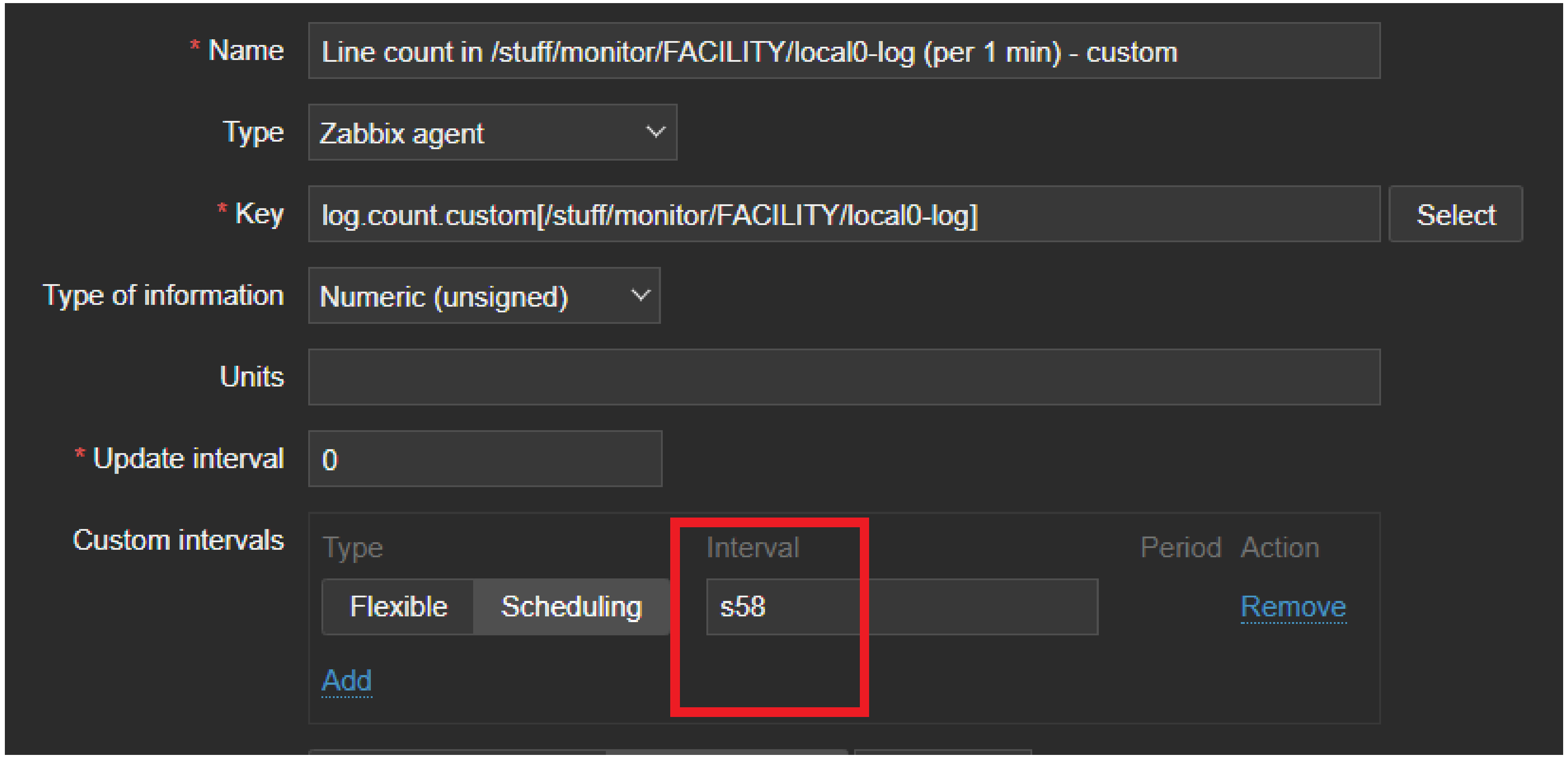
Why is it important?

Run as passive item



		last	min	avg	max
🟩 Line count in /stuff/monitor/FACILITY/local0-log (per 1 min)	[avg]	600 K	9.34 K	370.97 K	600 K
🟥 Line count in /stuff/monitor/FACILITY/local0-log (per 1 min) - custom	[avg]	554.18 K	113.48 K	370.39 K	675.36 K

Run as passive item



* Name	Line count in /stuff/monitor/FACILITY/local0-log (per 1 min) - custom
Type	Zabbix agent
* Key	log.count.custom[/stuff/monitor/FACILITY/local0-log]
Type of information	Numeric (unsigned)
Units	
* Update interval	0

Custom intervals

Type	Interval	Period	Action
Flexible [✓ Scheduling]	s58		Remove

Run as passive item

Another use case – active-passive clusters

Add virtual hostname to point to current active node and run such item as passive

This is how you won’t have unsupported / useless item (trigger) on passive and no manual work needed to control this https://github.com/b1nary1/zabbix

Thanks!