



aws SUMMIT

INDIA | MAY 25, 2023

ETECH003

Building a log strategy so you can sleep better at night

SPONSORED BY DATADOG

Mohamed ElNokali
Enterprise Solutions Architect
Datadog



© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.



**This is a streaming
service company**

CRON jobs run every night...



But one day, they stop working



So you sit at your computer during the night using tail -f.

```
ismail@ubu2:~$ tail -n 25 /var/log/auth.log
Feb 27 16:17:01 ubu2 CRON[28748]: pam_unix(cron:session): session closed for user root
Feb 27 16:25:01 ubu2 CRON[28849]: pam_unix(cron:session): session opened for user root by (uid=0)
Feb 27 16:25:01 ubu2 CRON[28849]: pam_unix(cron:session): session closed for user root
Mar  1 06:23:22 ubu2 systemd-logind[1221]: New seat seat0.
Mar  1 06:23:22 ubu2 systemd-logind[1221]: Watching system buttons on /dev/input/event0 (Power Button)
Mar  1 06:23:23 ubu2 sshd[1639]: Server listening on 0.0.0.0 port 22.
Mar  1 06:23:23 ubu2 sshd[1639]: Server listening on :: port 22.
Mar  1 06:25:01 ubu2 CRON[2477]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:25:01 ubu2 CRON[2478]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:25:02 ubu2 CRON[2478]: pam_unix(cron:session): session closed for user root
Mar  1 06:25:22 ubu2 CRON[2477]: pam_unix(cron:session): session closed for user root
Mar  1 06:35:01 ubu2 CRON[2709]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:35:01 ubu2 CRON[2709]: pam_unix(cron:session): session closed for user root
Mar  1 06:45:01 ubu2 CRON[2845]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:45:01 ubu2 CRON[2845]: pam_unix(cron:session): session closed for user root
Mar  1 06:52:01 ubu2 CRON[2943]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:52:01 ubu2 CRON[2943]: pam_unix(cron:session): session closed for user root
Mar  1 06:52:53 ubu2 sshd[2956]: Accepted publickey for ismail from 192.168.122.1 port 40788 ssh2: RSA
P44A41qdamkB+2+/sVtVHj8TpWRYByt5gzF4
Mar  1 06:52:53 ubu2 sshd[2956]: pam_unix(sshd:session): session opened for user ismail by (uid=0)
Mar  1 06:52:53 ubu2 systemd-logind[1221]: New session 6 of user ismail.
Mar  1 06:52:53 ubu2 systemd: pam_unix(systemd-user:session): session opened for user ismail by (uid=0)
Mar  1 06:55:01 ubu2 CRON[3142]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 06:55:01 ubu2 CRON[3142]: pam_unix(cron:session): session closed for user root
Mar  1 07:05:01 ubu2 CRON[3293]: pam_unix(cron:session): session opened for user root by (uid=0)
Mar  1 07:05:01 ubu2 CRON[3293]: pam_unix(cron:session): session closed for user root
```

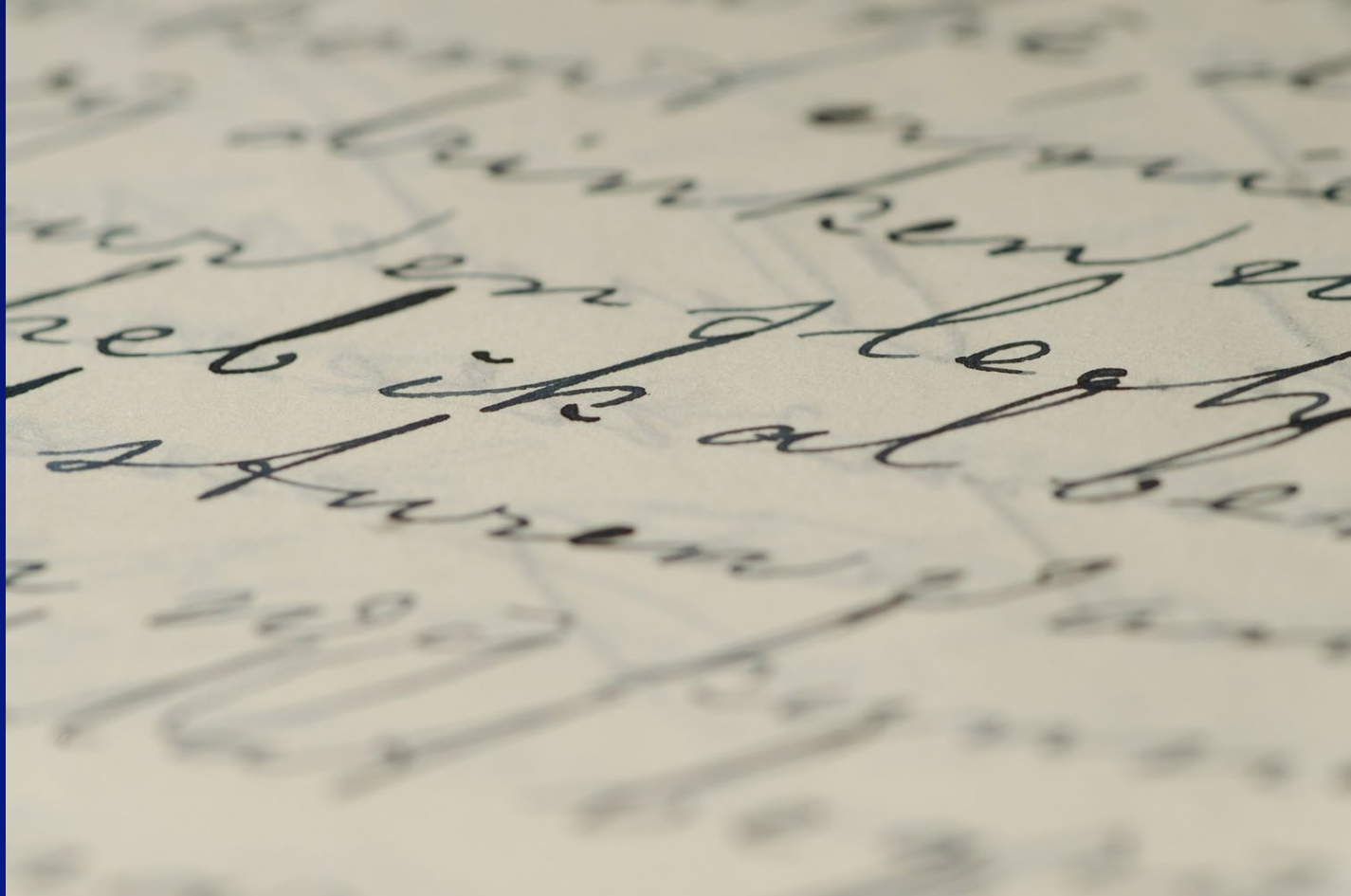
**And you fall asleep
before you can find a solution.**



Does the situation sound familiar?

What is a log?

At its most basic level, a log is just text.



A log is a record of an event

September/October			ROUTE	NOTES
DAY	DISTANCE	TIME / PACE		
Monday 25	6	53:20 8:53/m	Bike path -riverfront	Warm day, sunny. Had a good run
Tuesday 26	4	36:12 8:03/m	West St. Hill - Commons	Hot again! Went with Steve & Mary
Wednesday 27	-		off	
Thursday 28	8	64:32 8:04/m	Bridge road to the hill, woods trail for 3 miles and then back	With the running club. Had a strong run tonite...
Friday 29	3	27m 9 mi/mile		
Saturday 30	6	66m 8:15/m		
Sunday 1	-			
Week # 18	27	Goal = 30		2016
Previous YTD	862			
Year to Date	889	Total		

Different types of logs

Developers



*Application Logs
(NodeJS, Python)*



SRE



Systems Logs (MacOS, Windows, ...)
Service Logs (MariaDB, Nginx, ...)



A log can be just a print statement...

```
# A basic example of log
try:
    print("\nBackup succeed!")
except:
    print("====WARNING====")
    print("Backup failed!")
```

Backup succeed!

**...but we can agree
that it isn't a very good log**

Logs should be informative

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM
information_schema.tables WHERE table_schema = %(table_schema)s AND table_name =
%(table_name)
```

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine [generated in 0.00019s]
{'table_schema': 'inventory_db', 'table_name': 'product'}
```

Logs should be informative

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM  
information_schema.tables WHERE table_schema = %(table_schema)s AND table_name =  
%(table_name)
```

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine [generated in 0.00019s]  
{'table_schema': 'inventory_db', 'table_name': 'product'}
```


Logs should be informative

2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = %(table_schema)s AND table_name = %(table_name)

2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine [generated in 0.00019s] {'table_schema': 'inventory_db', 'table_name': 'product'}

Logs should be informative

2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM information_schema.tables WHERE table_schema = %(table_schema)s AND table_name = %(table_name)

2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine [generated in 0.00019s] {'table_schema': 'inventory_db', 'table_name': 'product'}

Logs should be informative

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM  
information_schema.tables WHERE table_schema = %(table_schema)s AND table_name =  
%(table_name)
```

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine [generated in 0.00019s]  
{'table_schema': 'inventory_db', 'table_name': 'product'}
```

Logs format can be different

Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```


Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

Here are the Nginx logs...

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

And here are the MariaDB logs.

2023-01-24 14:48:21 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)

2023-01-24 14:49:29 3 [Warning] Aborted connection 3 to db: 'inventory_db' user: 'dev' host: '172.20.0.3' (Got an error reading communication packets)

And here are the MariaDB logs.

2023-01-24 14:48:21 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)

2023-01-24 14:49:29 3 [Warning] Aborted connection 3 to db: 'inventory_db' user: 'dev' host: '172.20.0.3' (Got an error reading communication packets)

And here are the MariaDB logs.

2023-01-24 14:48:21 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)

2023-01-24 14:49:29 3 [Warning] Aborted connection 3 to db: 'inventory_db' user: 'dev' host: '172.20.0.3' (Got an error reading communication packets)

And here are the MariaDB logs.

2023-01-24 14:48:21 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)

2023-01-24 14:49:29 3 [Warning] Aborted connection 3 to db: 'inventory_db' user: 'dev' host: '172.20.0.3' (Got an error reading communication packets)

And here are the MariaDB logs.

2023-01-24 14:48:21 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)

2023-01-24 14:49:29 3 [Warning] Aborted connection 3 to db: 'inventory_db' user: 'dev' host: '172.20.0.3' (Got an error reading communication packets)



Recap

- Logs format are different from one service to another
- Date format is almost always different
- Logs can contain info such status code, exit codes, log levels

What is an effective log strategy?

It depends!



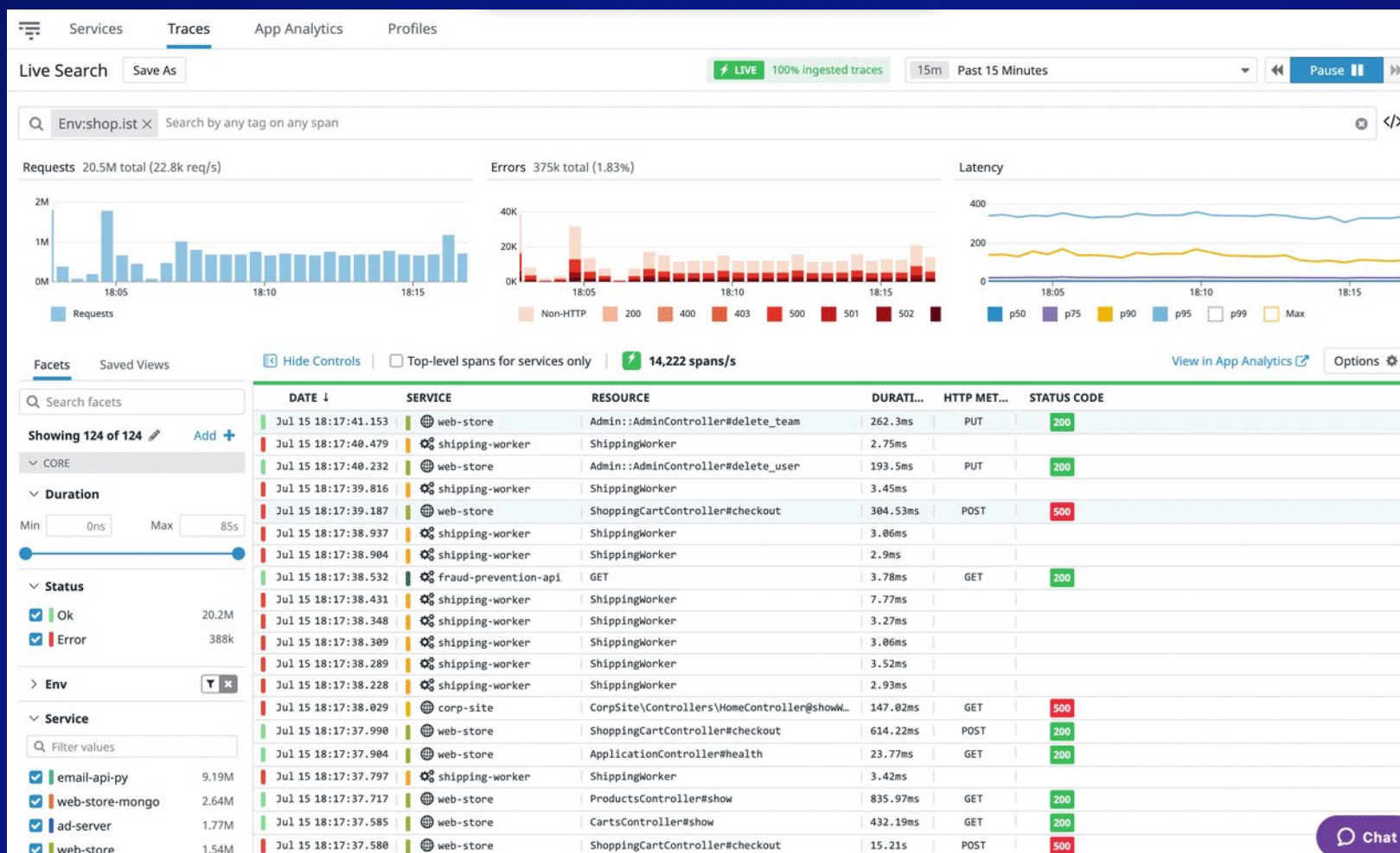
First Step: Pick up your log management tool

Having a tool will give you a vision and guidelines

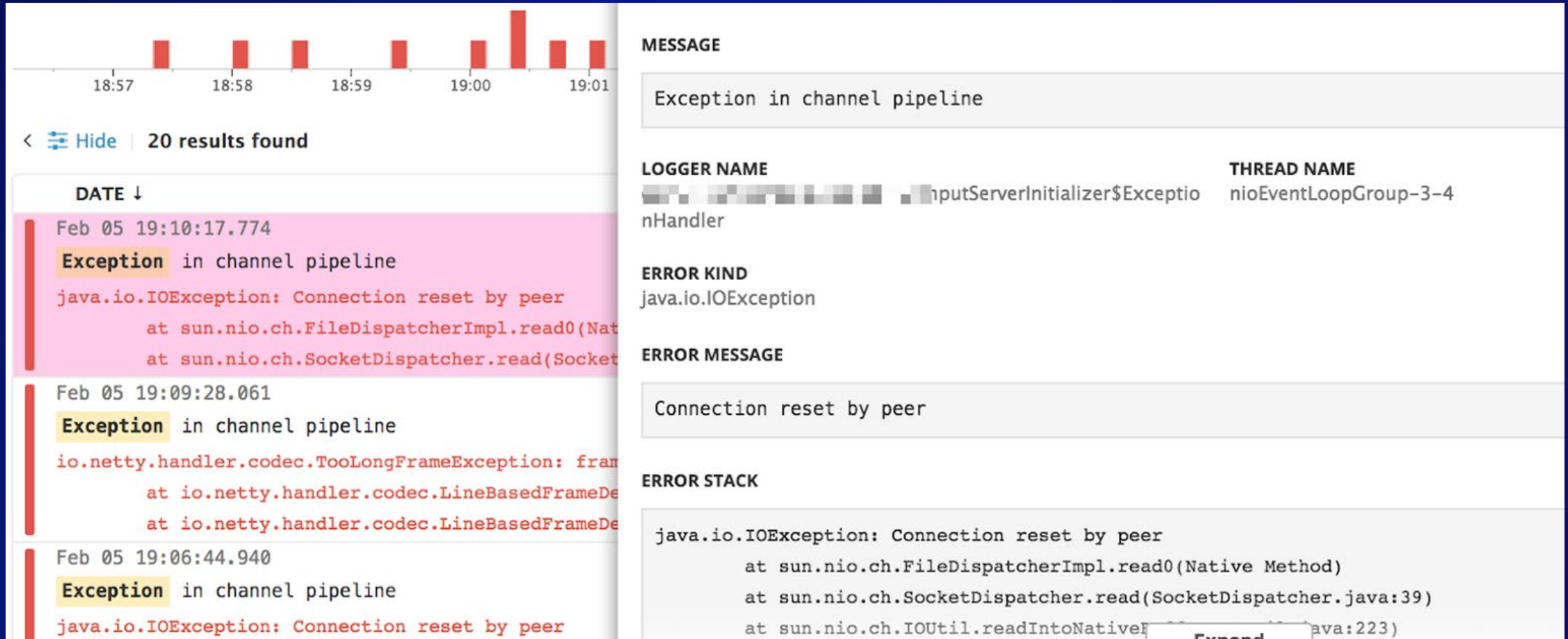


Your new log tool should allow you...

To search your data across your logs



To be aware of bugs or security threats



The screenshot displays AWS CloudWatch logs for a Java application. At the top, a bar chart shows log activity over time from 18:57 to 19:01. Below the chart, a list of log entries is shown, with the first three entries highlighted in pink. Each entry includes a date and time, a message, and a stack trace. The messages indicate 'Exception in channel pipeline' and 'Connection reset by peer'. The stack traces show the exception originates from the Netty framework and the Java NIO library.

MESSAGE

Exception in channel pipeline

LOGGER NAME

InputServerInitializer\$ExceptionHandler

THREAD NAME

nioEventLoopGroup-3-4

ERROR KIND

java.io.IOException

ERROR MESSAGE

Connection reset by peer

ERROR STACK

```
java.io.IOException: Connection reset by peer
    at sun.nio.ch.FileDispatcherImpl.read0(Native Method)
    at sun.nio.ch.SocketDispatcher.read(SocketDispatcher.java:39)
    at sun.nio.ch.IOUtil.readIntoNativeBuffer(SocketDispatcher.java:223)
```

Determine how long to retain logs

New Index

Filter

Grant access of this index's content to

Grant editing Exclusion Filters of this index to

2 Set retention period

15 days

3 days

7 days

15 days

30 days

45 days

60 days

Volume exceeds million/day

Cancel Save

In short, they will give you guidance

Logs Guides

Logging Without Limits™


Programmatically access log data using the Logs Search API	→
Logging Without Limits™ Guide	→
Correlate Logs with Metrics	→
Monitor your log usage	→

LANGUAGE

English ▾

SITE ?

US ▾

 EDIT

ON THIS PAGE

Log Collection

Send AWS services logs with the Datadog Kinesis Firehose Destination	→
Sending Events and Logs to Datadog with Amazon EventBridge API Destinations	→
Send AWS services logs with the Datadog Lambda function	→
Collect Heroku Logs	→
Log Collection Troubleshooting Guide	→

Second step: Structure and parse your logs

Structured logging



JSON and structured logging

// Raw Format

```
2023-01-24 14:57:43,241 INFO sqlalchemy.engine.Engine SELECT COUNT(*) FROM information_schema.tables
WHERE table_schema = %(table_schema)s AND table_name = %(table_name)
```

// JSON Format

```
{
  "asctime": "2023-01-24 14:57:43,241",
  "levelname": "INFO",
  "name": "sqlalchemy.engine.Engine",
  "message": "SELECT COUNT(*) FROM information_schema.tables WHERE
table_schema = %(table_schema)s AND table_name = %(table_name)"
}
```


Using Grok Parsing with Nginx

```
172.17.0.1 - - [22/Mar/2023:13:47:26 +0000] "GET / HTTP/1.1" 200 7231 "-" "Mozilla/5.0  
(Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.74 Safari/537.36" "-"
```

```
172.17.0.1 - - [22/Mar/2023:13:51:02 +0000] "GET /asd HTTP/1.1" 200 7234 "-"  
"Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:98.0) Gecko/20100101 Firefox/98.0" "-"
```

Creating logs pipelines

▼

4

Nginx (updat... source:nginx

Feb 10 2022

- 1 Grok Parser: Parsing Nginx logs
- 2 Remapper: Remap client to client ip
- 3 Grok Parser: Parsing Nginx Error log requests
- 4 Url Parser
- 5 User-Agent Parser
- 6 Date Remapper: Define Date_access as the official timestamp of the log
- 7 Category Processor: Categorise status code
- 8 Status Remapper: Set the log status based on the status code value
- 9 GeoIP Parser: geoip
- 10 Arithmetic Processor: duration = http.response_time_s * 1000000000
- 11 Grok Parser: Grok Parser
- 12 Grok Parser: hour extraction
- 13 Category Processor: business_hours

Parsing with Grok Parser

2

Define parsing rules ?

```
access.common %{_client_ip} %{_ident} %{_auth} \[%{_date_access}\] "(?>%{_method} |)%{_url}(?> %  
{_version}|)" %{_status_code} (?>%{_bytes_written})|-
```

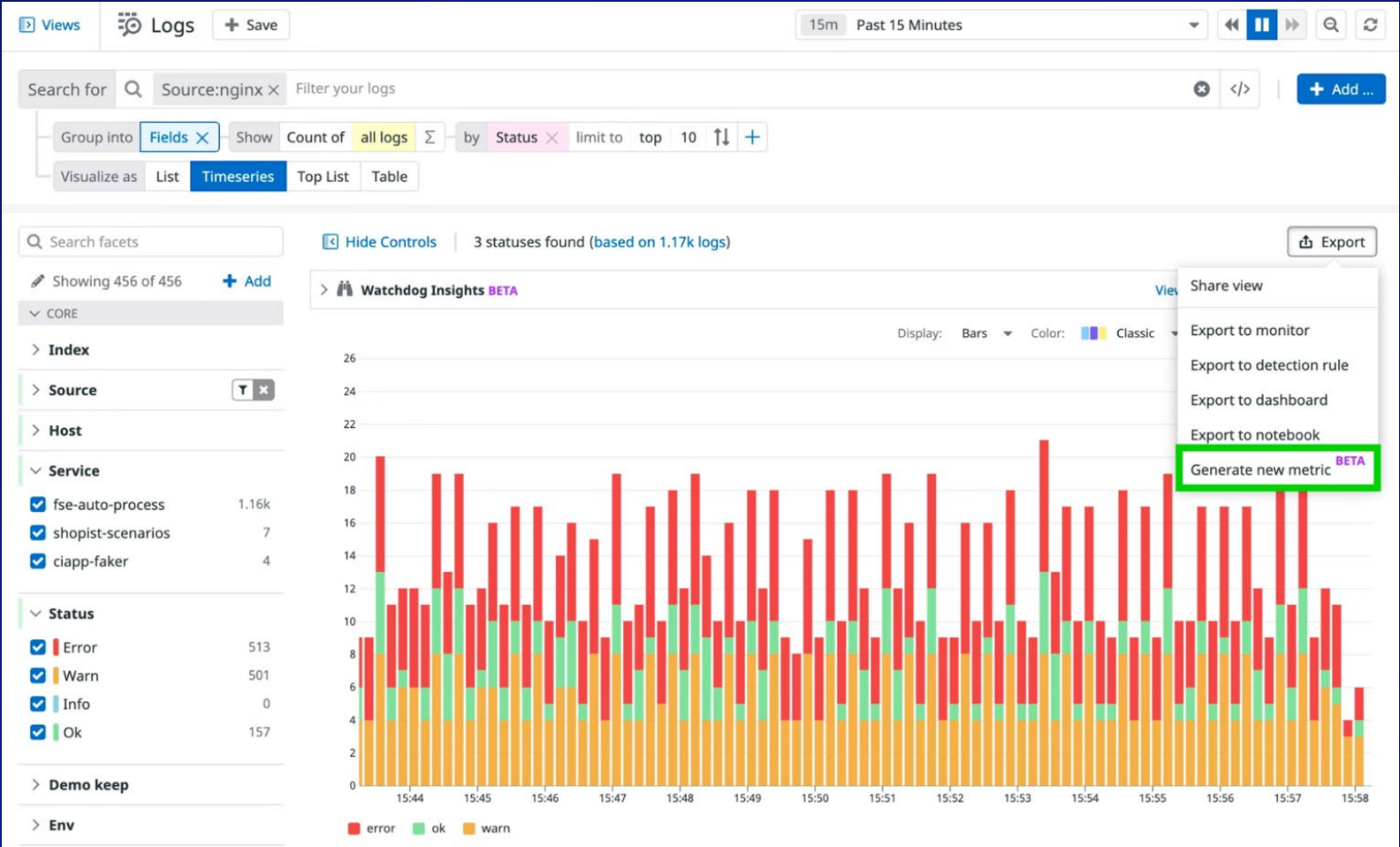
```
access.combined %{access.common} "%{_referer}" "%{_user_agent}"( "%{_x_forwarded_for}")?.*
```

```
error.format %{date("yyyy/MM/dd HH:mm:ss"):date_access} \[%{word:level}\] %{data:error.message}  
(, %{data::keyvalue(": ", ",")})?
```

#Sample

```
#127.0.0.1 - frank [13/Jul/2016:10:55:36 +0000] "GET /apache_pb.gif HTTP/1.0" 200 2326  
#172.17.0.1 - - [06/Jan/2017:16:16:37 +0000] "GET /datadoghq/company?test=var1%20P1 HTTP/1.1"  
200 612 "http://www.perdu.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/55.0.2883.87 Safari/537.36" "-"  
#2017/09/26 14:36:50 [error] 8409#8409: *317058 "/usr/share/nginx/html/sql/sql-admin/index.html"  
is not found (2: No such file or directory), client: 217.92.148.44, server: localhost, request:  
"HEAD http://174.138.82.103:80/sql/sql-admin/ HTTP/1.1", host: "174.138.82.103"
```


Generate metrics from logs



Third step: Configuring alerts

Creating alerts for CRON logs

1 Define the search query

Q Service: coffee-house x

2 Set alert conditions

Trigger when the metric is the threshold during the last

Alert threshold:

Warning threshold:

3 Say what's happening

☐ Preview ☒ Edit

The application Coffee House is sending too much logs

Coffee-house is sending too much log. Go to the Index filters page and activate the coffee-house-noisy filter: <https://app.datadoghq.com/logs/pipelines>

{{#is_warning}}@slack-coffee-house-team{{/is_warning}}

{{#is_alert}}@pagerduty-coffee-house{{/is_alert}}

Configuring alert conditions

ALERT

Monitor status since 10 hours and 14 minutes ago (23 Mar, 7:03:11 Europe/Paris)

Mute

Resolve

Declare Incident

erreur

Properties

Log Monitor

ID:

Created at Feb 28, 2022, 5:50 pm

by

TAGS

BOTS

PRIORITY Not Defined

QUERY

logs("service: status:error").index("*").rollup("count").by("container_name").last("1d") > 1

MESSAGE

@alert@ @webhook-Discord-Bots

2 notified

alert@.com Webhook: Discord-Bots

Filter monitor groups and their events

Alert 2

Warn 0

No Data 0

OK 0

2 of 2 groups

4w

UTC+01:00

Feb 23, 5:15 pm – Mar 23, 5:15 pm

Status & History

GROUP STATUS

Showing 2 of 2 groups


Sort by Triggered

NAME	VALUE
container_name:\	5


aws

© 2023, Amazon Web Services, Inc. or its affiliates. All rights reserved.

And be alerted by Slack or by mail


**Datadog** APP 2:17 PM

Triggered: Backend returns too many errors
Follow this runbook to investigate:
<https://app.datadog.com/notebook/123/Backend-Error-Troubleshooting>







More than 1 log events matched in the last 5m against the monitored query:
source:java AND @http.status_code:[400 TO 599]



Tags
total

Notified


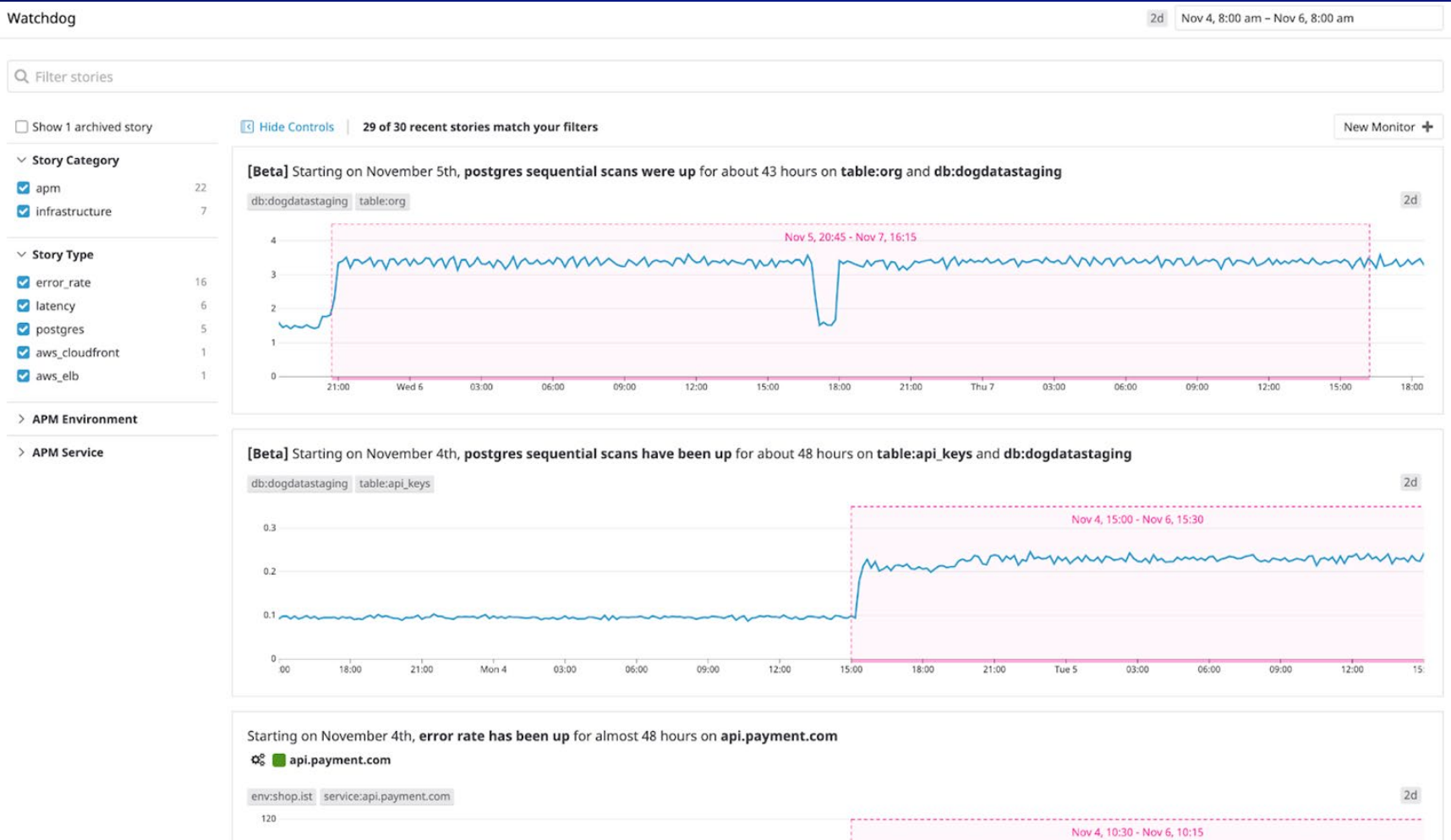
Time	Host

12:16:20 UTC	i-0d6a8 
172.20.203.39 - - [26/Sep/2018:12:16:20 +0000] "POST	
/api/a  /analytics/reque	

12:15:20 UTC	i-07b919 
172.20.203.39 - - [26/Sep/2018:12:15:20 +0000] "POST	
/api/c  /analytics/reque	

12:14:20 UTC	i-054604 
172.20.203.39 - - [26/Sep/2018:12:14:20 +0000] "POST	
/api/c  /analytics/reque	

Watchdog insights

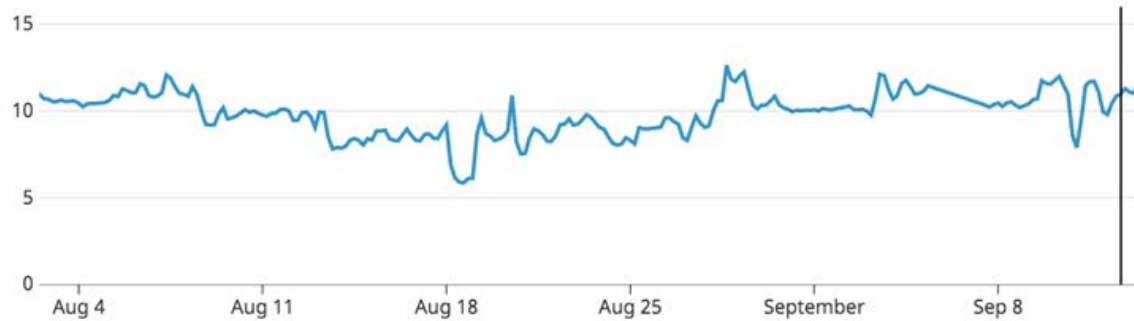


Generating error rates based on metrics

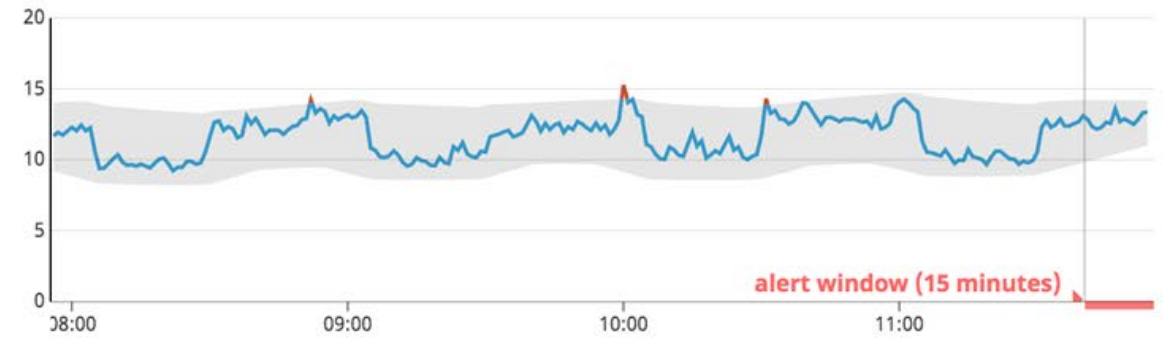


Using anomaly detections

HISTORICAL VIEW



EVALUATION PREVIEW



Alert when **100** % out of the bounds for the last **15m**
Recover when **0** % out of the bounds for the last **15m**

0 % out of bounds

**Fourth Step:
Enjoy your nights while these
tools work for you**

Thank you!

Mohamed ElNokali
Enterprise Solutions Architect
Datadog



Please complete the
session survey