

Add a filter +

Selected Fields

message

Available Fields

@timestamp

functionName

lambdaVersion

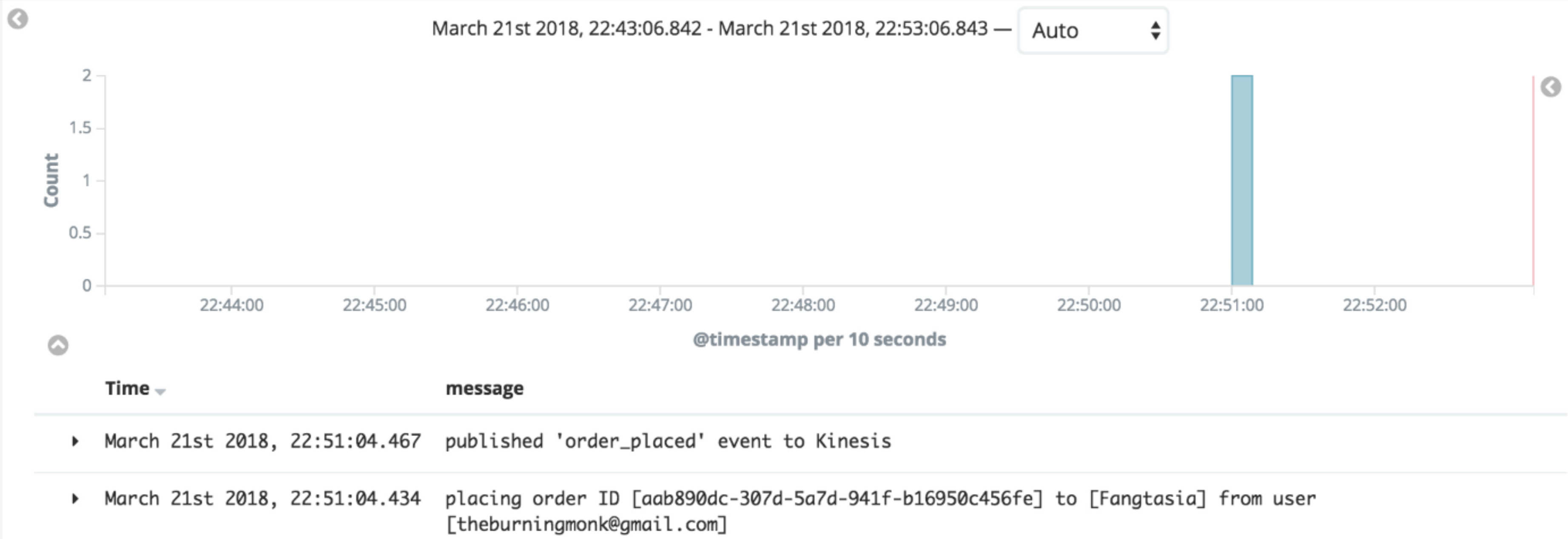
level

logGroup

logStream

tags

type



Time ▾

message

▶ March 21st 2018, 22:51:04.467 published 'order_placed' event to Kinesis


▼ March 21st 2018, 22:51:04.434 placing order ID [aab890dc-307d-5a7d-941f-b16950c456fe] to [Fangtasia] from user [theburningmonk@gmail.com]

Table

JSON

[View surrounding documents](#) [View single document](#)

🕒	@timestamp	🔍	🔍	📄	✱	March 21st 2018, 22:51:04.434
t	functionName	🔍	🔍	📄	✱	big-mouth-dev-place-order
t	lambdaVersion	🔍	🔍	📄	✱	\$LATEST
t	level	🔍	🔍	📄	✱	debug
t	logGroup	🔍	🔍	📄	✱	/aws/lambda/big-mouth-dev-place-order
t	logStream	🔍	🔍	📄	✱	2018/03/21/[\$LATEST]3e1e1c403ff2438e9a499199bb0818d3
t	message	🔍	🔍	📄	✱	placing order ID [aab890dc-307d-5a7d-941f-b16950c456fe] to [Fangtasia] from user [theburningmonk@gmail.com]
t	tags	🔍	🔍	📄	✱	_logz_json_tcp_5050
t	type	🔍	🔍	📄	✱	cloudwatch

Time ▾	message
▶ March 21st 2018, 22:51:04.467	published 'order_placed' event to Kinesis
▼  March 21st 2018, 22:51:04.434	placing order ID [aab890dc-307d-5a7d-941f-b16950c456fe] to [Fangtasia] from user [theburningmonk@gmail.com]

Table

JSON

[View surrounding documents](#)
[View single document](#)

🕒 @timestamp					March 21st 2018, 22:51:04.434
t functionName					big-mouth-dev-place-order
t lambdaVersion					\$LATEST
t level					debug
t logGroup					/aws/lambda/big-mouth-dev-place-order
t logStream					2018/03/21/[\$LATEST]3e1e1c403ff2438e9a499199bb0818d3
t message					placing_order ID [aab890dc-307d-5a7d-941f-b16950c456fe] to [Fangtasia] from user [theburningmonk@gmail.com]
t tags					_logz_json_tcp_5050
t type					cloudwatch

use **structured logging** with JSON



What is structured logging and why developers need it

MATT WATSON | JANUARY 31, 2017 | [DEVELOPER TIPS, TRICKS & RESOURCES](#) |

[LEAVE A COMMENT](#)



Log files are one of the most valuable assets that a developer has. Usually when something goes wrong in production the first thing you hear is “send me the logs”. The goal of structured logging is to bring a more defined format and details to your

<https://stackify.com/what-is-structured-logging-and-why-developers-need-it/>

Log Everything as JSON. Make Your Life Easier.

April 26, 2012

The Story of an Engineer.

Here is an anecdote. I am sure some of you have had a similar experience.

- Alex, an engineer, logs all kinds of events. Since he is the primary consumer of the log, the format is optimized for human-readability. One canonical example is Apache logs: 10.0.1.22 -- [15/Oct/2010:11:46:46 -0700] "GET /favicon.ico HTTP/1.1" 404... 10.0.1.22 -- [15/Oct/2010:11:46:58 -0700] "GET / HTTP/1.1" 200... This looks great. Time stamp, URL, HTTP status code... each line gives Alex a lot of information to work with if the service is having issues.
- Bob, a business analyst, asks Alex for the number of daily unique users. Alex writes a parser for the Apache log and crontabs the script. He also builds a little Web interface so that his colleague can query the parsed data on his own. Bob finds the interface super useful.
- Bob comes back a few weeks later and complains that the web interface is broken. Alex scratches his head and takes a look at the logs, only to realize that someone added an extra field in each line, breaking his custom parser. He pushes the change and tells Bob that everything is okay again. Instead of writing a new feature, Alex has to go back and has to fill back the missing data.
- Every 3 weeks or so, repeat Step 3.

What is Wrong With This?

<https://blog.treasuredata.com/blog/2012/04/26/log-everything-as-json/>

8 Handy Tips to Consider When Logging in JSON

By [JASON SKOWRONSKI](#) | 04 Jan 2017

Logging in JSON transforms your logs from raw text lines to a database of fields you can search, filter, and analyze. This gives you way more power than you would get with only raw logs.

Why Use JSON?

JSON makes it easier to search and analyze your data. Say this is a portion of your log file:


```
1 Hoover, 29, 251 Kearny Street, San Francisco, CA, 2012-09-29
2 Fred, 21, 123 Great Avenue, Teton, ID, 2012-09-29
```

gistfile1.txt hosted with  by GitHub

[view raw](#)

I want to figure out how many 29 year olds are from San Francisco. This may look familiar to people who are used to dealing with raw unstructured logs:

```
1 $ grep 29 file.log | cut -d , -f 2 | sort | uniq -c | sort -nr
```

gistfile1.sh hosted with  by GitHub

[view raw](#)

With the above approach, you'll end up with log entries that include any text that has "29" in it, including the date for Fred who is only 21 years old. So then you end up with an incorrect count, or an even more complicated command.

<https://www.loggly.com/blog/8-handy-tips-consider-logging-json/>



CloudWatch Logs

\$0.50 per GB ingested

\$0.03 per GB archived per month



CloudWatch Logs



1M invocation of a 128MB function =
 $\$0.0000000208 * 1M + \$0.20 =$
 $\$0.408$

\$0.50 per GB ingested

\$0.03 per GB archived per month

DON'T leave debug logging **ON** in production

what we'll do in this module

- replace *console.log* with a JSON logger

what we'll do in this module

- replace *console.log* with a JSON logger
- allow log level to be configurable by environment