Make sense of your logs with Sumologic and Cloudwatch Insights

medium.com/pixel-and-ink/make-sense-of-your-logs-with-sumologic-and-cloudwatch-insights-f023747e04f9

Andrew Davis December 6, 2020

Andrew Davis

Logs are a crucial part of monitoring the performance of your production environment when it's running normally and figuring out what went wrong when it's not. We do some reasonably sophisticated stuff with our logs but it still doesn't take much activity to generate more log data than you can expect to sit and read.

Tools like Sumologic and and Cloudwatch Insights exist to help you sort through the haystack to find the needle you require. Each has their own flavour of log query syntax and their own way to tackling common tasks.

Task 1 — Figure Out What The Hell Happened

Something has gone awry with order 17409 and you need to figure out what. Alternatively, the string "17409" may be an error message someone received.

First, search for references to that order ID. Then extract the correlation IDs.

Sumo logic

```
_sourceCategory=Prod/*/importantAPI "17409"| json "correlationId"
```

A star is a wildcard. In this case the categories "Prod/sales/importantAPI" and "Prod/inventory/importantAPI" would match. You can also do things like "Prod/*/*API" to match all APIs across all production systems or "Prod/*/*" to search every system in production.

The correlation ID extraction part of this query is mostly just to show how you can extract a single field from a json log entry. You could just as easily remove that line and see whole log entries. You should however definitely have correlation IDs in your log entries.

Now you have the correlation IDs you can search again and see the entire life cycle of each of the requests that touched the problem order. Start request, verifying received parameters, checking the account is valid etc etc. Having extracted only the relevant requests you now want to see as much detail as possible.

```
_sourceCategory=Prod/*/importantAPI ("correlationId1" or "correlationId2")
```

With any luck you should now be on the path to solving the mystery.

Cloudwatch Insights

First note that the category is not included in the query text itself. You need to select one or more categories above the query text area to run your query against.

Insights uses regular expressions. You can make your regexs however complicated you like. You also need to explicitly order your results.

```
fields @timestamp, @message| filter @message like /17409/| sort @timestamp desc| display correlationId| limit 200
```

And then to search using the correlation IDs.

```
fields @timestamp, @message| filter  like /correlationId1/ or  like /correlationId2/| sort  desc| limit 20
```

Task 2 — Figure Out How Fast Your System Is (And Spot When It Gets Worse)

A simple graph can tell you how slow or fast your system is. Crucially, it can also tell you when it suddenly gets slower.

Sumologic

```
_sourceCategory=Prod/*/*API and “end request” and “method\”:\”GET” and !”/health-check”| json auto| timeslice 5m| avg(responseTime) as avg_duration by _timeslice, _sourceCategory| order by _timeslice| transpose row _timeslice column _sourceCategory
```

This searches API logs and extracts the end request log entries as those typically have some information about request duration.

I’m limiting it to ‘get’ requests as those are the most performance critical in my use case. You may want to separate gets, posts etc or you may be happy to have them all combined.

I’m excluding requests for ‘/health-check’ which is a route the load balancer uses to determine the host’s health. They are super fast requests that make the API look faster than it really is.

The remaining log entries are grouped into 5 minute buckets and response time is averaged. The row transposition is a bit of a Sumologic-ism required to produce a graph. You can look at the rows output by your query to see what it is doing if you are interested :)

Cloudwatch Insights

```
fields @timestamp, @message, req.method| filter (msg = 'end request')| filter
@message not like /\/health-check/| filter (req.method = 'GET')| sort @timestamp
desc| stats avg(responseTime) by bin(5m)
```

The above does essentially the same thing in a, in my opinion, far more readable way.

In the case of both Sumologic and Cloudwatch Insights you should be able to expand the time period you're searching to be several weeks or more so you can look for unexplained changes. If Sumologic complains about the volume of rows being returned change "timeslice 5m" to "timeslice 1h" for one hour time slices instead of 5 minutes.

Task 3 — Figure out whether the volume of errors is getting better or worse

Ideally your production environment would log errors so rarely that each incidence warrants an in-depth investigation. More likely however you have a steady low background rate of errors being logged that represent normal operation.

If this rate of errors suddenly changed it sure would be handy to know that as soon as it happens.

Sumo logic

```
(_sourceCategory=Prod/*/WWW or _sourceCategory=Prod/*/*API) | json auto| where level
> 40 |timeslice 5m| count by _timeslice, _sourceCategory| transpose row _timeslice
column _sourceCategory
```

This is searching the production web tier and the APIs for any log entry with a level above 40. That usually means only errors. These are divided into 5 minute buckets and made ready to appear on a graph.

Cloudwatch Insights

Make very sure to select all of your production log groups selected for this.

```
fields @timestamp, @message, msg| filter (level > 40)| stats count(*) by bin(5m)
```

Task 4 — Spot Slow Database Queries

Although its not a substitute for tools like Postgres' pg_state_statements for deep diving into query performance your logging can give you a general sense of whether things are getting better or worse.

This is reliant on you explicitly logging query duration.

These search the API logs for log messages containing the string “queryDuration”. Where query duration is more than 100ms, group them into 5 minute buckets and graph the number of these queries.

Sumologic

```
_sourceCategory=Prod/*/importantAPI “queryDuration”| json auto | where queryDuration
> 100| timeslice 5m| count _sourceCategory, _timeslice| transpose row _timeslice
column _sourceCategory
```

Cloudwatch Insights

```
fields @timestamp, @message| filter @message like /queryDuration/| filter
queryDuration > 100| sort @timestamp desc| stats count() by bin(5m)
```

Now Build A Dashboard

Although you should get comfortable with the search syntax you probably don’t want to have to type these queries out over and over.

Both Sumologic and Cloudwatch allow you to build and save dashboards to satisfy your routine monitoring needs. A dashboard is a screen filled with pretty graphs and aggregated data.

Spending a few hours putting together a production overview dashboard was arguably my most productive use of time ever. Yours should have graphs showing key metrics like error volume, the speed of your various tiers (web tier, APIs), and the volume of key events like sales occurring, user sign ups etc where the absence of data indicates an urgent problem.

With the help of your dashboard you will spot a surge of errors seconds after it starts. You will also notice when your key metric suddenly drops off a cliff. Your dashboard doesn’t tell you everything but it tells you when something is wrong and needs to be investigated.