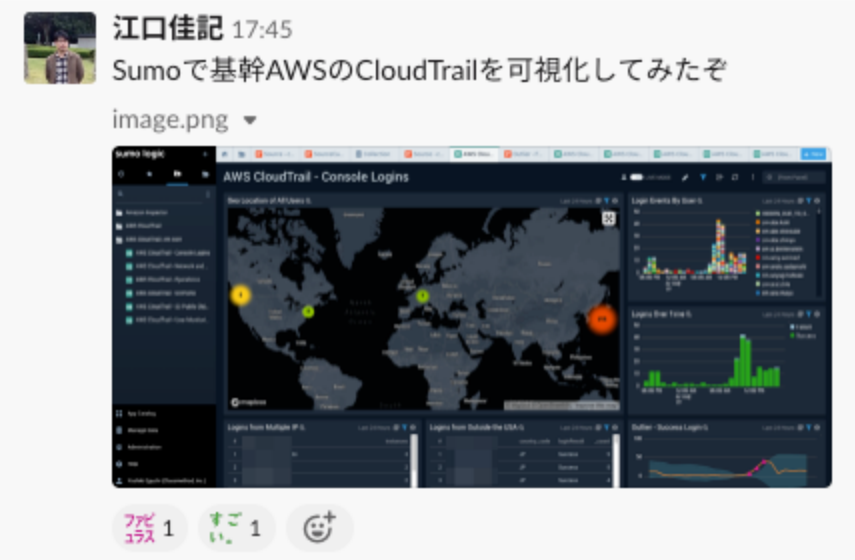
Sumo Logicで社内の基幹AWS環境の情報を可視化してみた（CloudTrail編） | DevelopersIO

dev.classmethod.jp/articles/sumo-analyze-cloudtrail-dashboard

江口佳記

August 30, 2019



This is Eguchi from the Operations Department.

Our company is engaged in what is known as dogfooding, which is the practice of trying out our own services and the services we handle ourselves. As part of this, someone within the company suggested that it might be interesting to try using Sumo Logic, a log management and analysis tool we handle, to monitor our core AWS accounts, and so we decided to give it a try.

First, we visualized CloudTrail and tried various things while sharing them on the operations department's Slack, so I would like to share them with you.

What is our core AWS account?

This is a shared account that manages the AWS accounts of our employees. Each of our engineers has their own AWS testing environment, but after logging in to this core AWS account, they can switch and roll their own AWS testing environment.

By using Sumo Logic to visualize the CloudTrail information for this core AWS account, you should be able to more easily check and detect suspicious behavior around the account.

setting

There is already a detailed article on developers.io about how to get CloudTrail information in Sumo Logic, so please refer to it here.

> I tried CloudTrail security monitoring super easily [Sumo Logic]

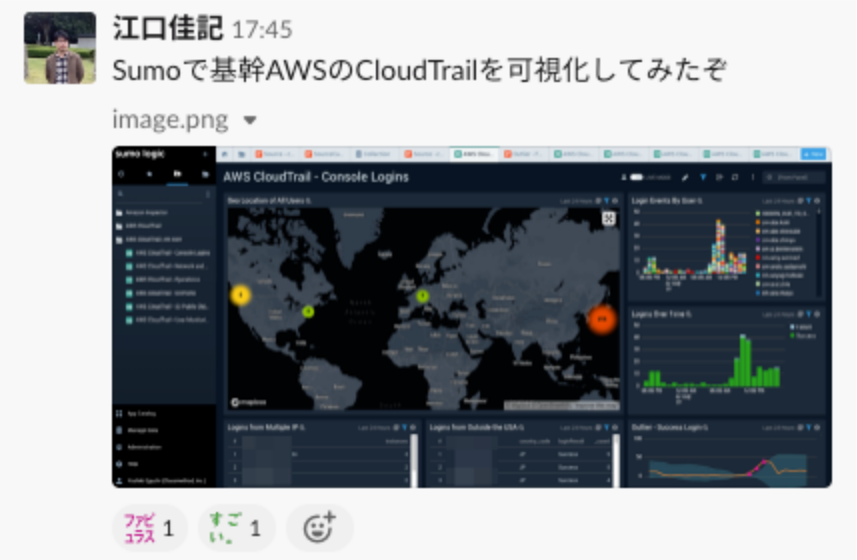
The above article introduced a method using IAM user access keys, but Sumo Logic has since added support for access using IAM roles. This method is currently recommended as it is more secure (we used this method to set it up this time as well). There is also an article on how to set up using IAM roles, so please refer to that.

> Sumo Logic Now Supports IAM Roles

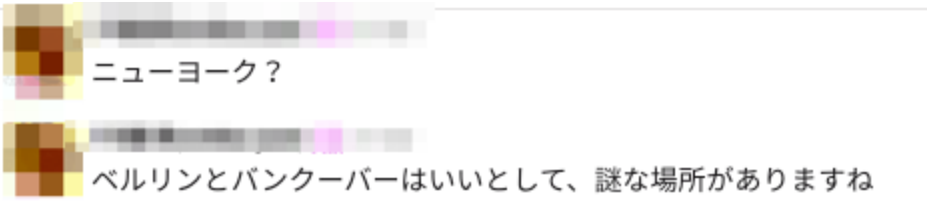
By the way, the information you can get from the CloudTrail dashboard is explained in the blog post above , so please see here for more details.

I tried it

Once I set it up and the information started appearing on the dashboard, I quickly reported it on the operations team's Slack channel.

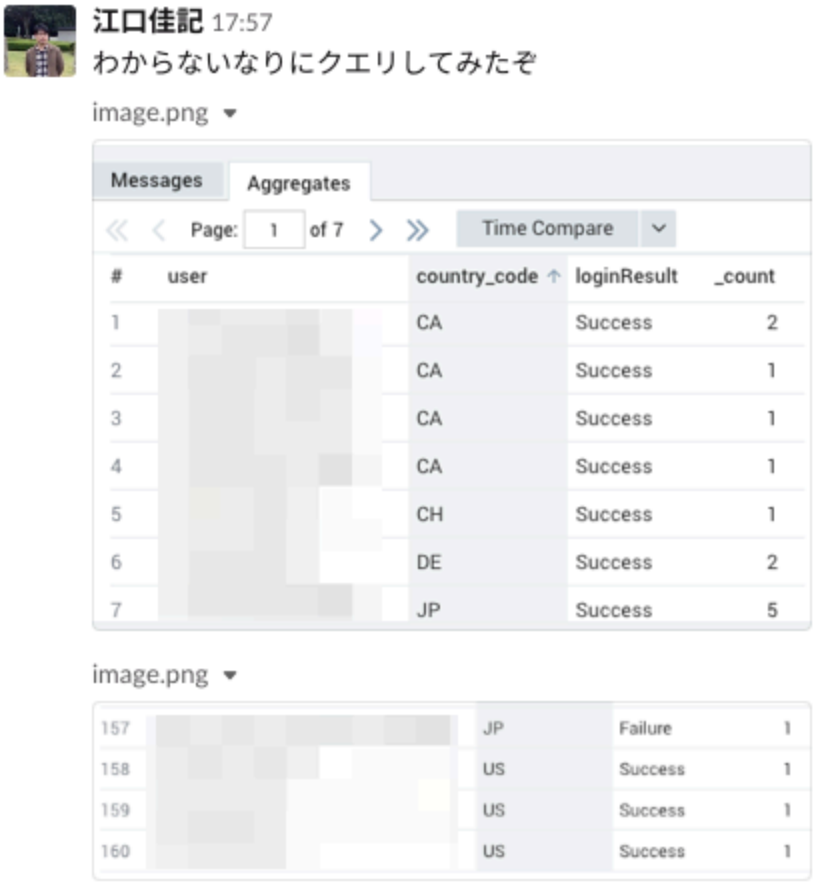


The screenshot shared was of a dashboard called "Console Logins" that visualizes login information, but at this point, someone pointed out something concerning about the map information showing the location of logged-in accounts.



Classmethod has branches in Vancouver and Berlin, but it doesn't have a branch in New York, so that's strange.

Clicking on the map here allows you to drill down and quickly see who accessed the site from New York... which is convenient, but unfortunately, clicking on the map only zooms in. So I created my own query to get the information.



Although it's blurred, you can see which country each account logged in from. The query looks like this. "Console Logins" has a panel that lists logins from outside the US, so I simply removed the part that filters out "non-US" from the query for that.

```
_sourceCategory = cm-core/cloudtrail  "ConsoleLogin"
| parse "\"eventName\":\"*\"" as eventName nodrop
| where eventName="ConsoleLogin"
| json "sourceIPAddress"
| parse "\"userName\":\"*\"" as user_name nodrop
| json field=_raw "userIdentity.principalId" as principal_id nodrop
| parse regex field = principal_id ":(?<user_principal>.+)" nodrop
| if (user_name="", user_principal, user_name) as user
| json field=_raw "responseElements.ConsoleLogin" as loginResult nodrop
| parse "\"MFAUsed\":\"*\"" as mfaUsed nodrop
| count by sourceIPAddress, user, loginresult
| lookup latitude, longitude, country_code, country_name, region, city, postal_code
from geo://location on ip = sourceIPAddress
| fields user, country_code, loginresult, _count
| sort by _count, country_code asc, user, loginresult
```

We also received a request to "check only accounts that failed authentication," so we responded to that request.

This simply adds a filter before the last line of the query above that tests if `loginResult`the value of the field indicating console login success/failure is failure ( ).`Failure`

```
_sourceCategory = cm-core/cloudtrail  "ConsoleLogin"
（略：前掲のクエリと同じ）
| where loginResult == "Failure"
| sort by _count, country_code asc, user, loginresult
```

You can also check information about operations where access was denied, rather than console logins. Out of curiosity, I wondered which User Agents were causing the most access denials, so I created a query and found that there were an awful lot of access

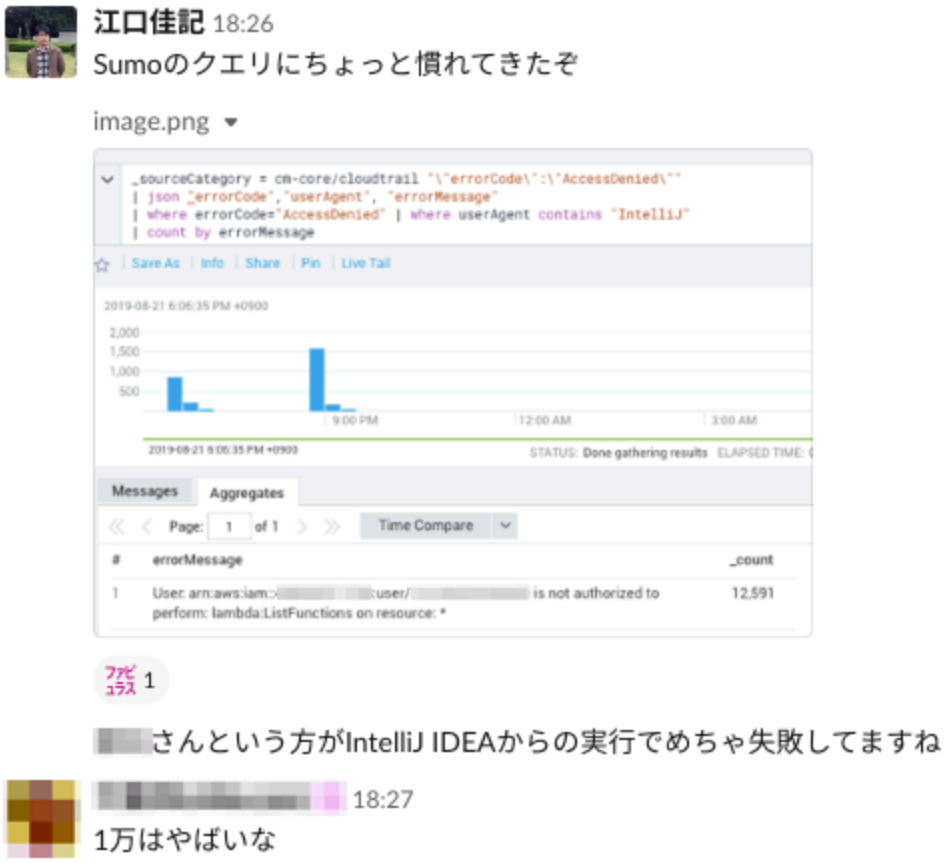
denials from IntelliJ IDEA.

*The above says "Login failed," but more accurately it means access was denied.

The query looks like this: Simple... it filters by AccessDenied, extracts the userAgent field as JSON, and counts the number of events found for each userAgent within the specified period.

```
_sourceCategory = cm-core/cloudtrail  "\"errorCode\":\"AccessDenied\""
| json "userAgent"
| count by userAgent
```

I wondered if there were multiple error messages, but when I looked, I found there was only one. The error message also included the ARN, so I realized that the error was likely occurring in IntelliJ IDEA by the same user.



The query looks like this: (It's a little different from the screenshot, but you can still extract the desired information.)

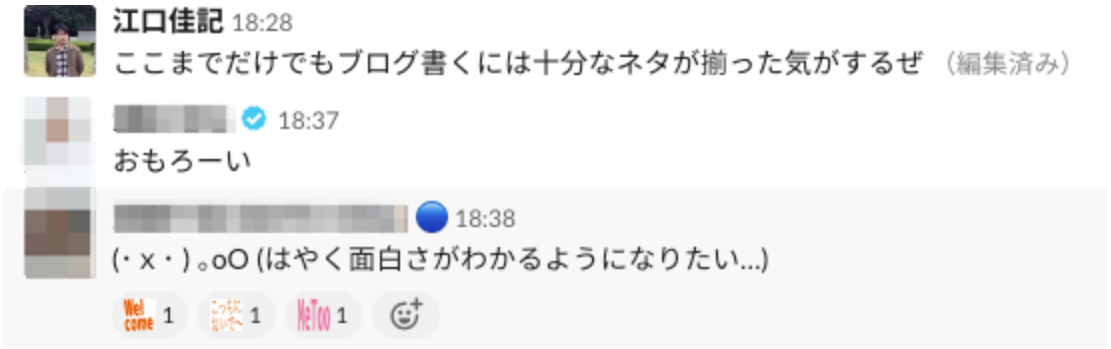
```
_sourceCategory = cm-core/cloudtrail  "\"errorCode\":\"AccessDenied\""
| json "userAgent","errorMessage"
| where userAgent contains "IntelliJ"
| count by errorMessage
```

`where`In the section, we filter only those where userAgent is IntelliJ and aggregate by error message. If you want to aggregate by user name, you can do it with the following query.

```
_sourceCategory = cm-core/cloudtrail  "\"errorCode\":\"AccessDenied\""
| json "userAgent","errorMessage" | json "userIdentity.userName" as user_name
| where userAgent contains "IntelliJ"
| count by user_name
```

Conclusion

In fact, all this discussion took place in about 30 minutes after we created the CloudTrail dashboard. I think it's amazing how by visualizing data and discussing it together, we can perform a variety of analyses in a short amount of time.



In addition to this, we are also trying out visualization with GuardDuty, and there are talks within the company about whether we can actually incorporate it into our operations in the future. I would like to continue sharing more about Sumo Logic dogfooding.

That's all for now. Thank you for reading!