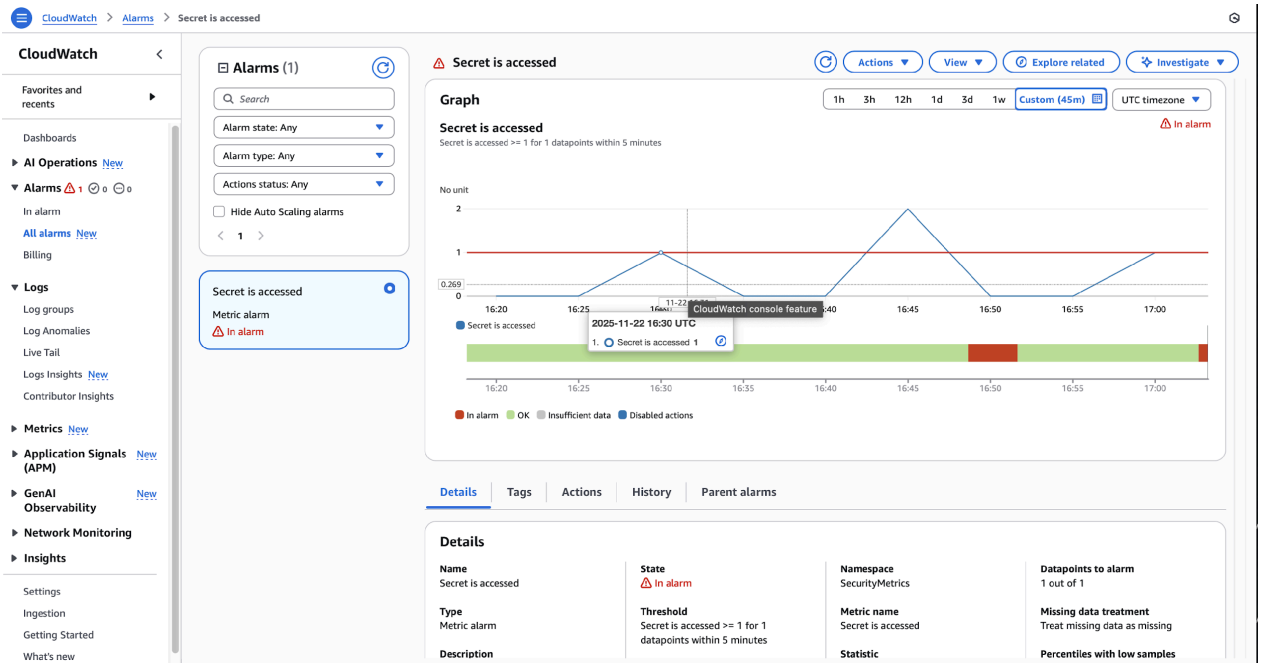
Build a Security Monitoring System

Erik Gonzalez



Introducing Today's Project!

In this project, I will demonstrate how to set up a monitoring system in AWS using CloudTrail, CloudWatch and SNS! I'm doing this project to learn how security and monitoring services in AWS work, plus have a working system that actually sends me emails too.

Tools and concepts

Services I used were cloudtail, cloudwatch, sns. I also used IAM roles, secret managers and S3 buckets. Key concepts: I learned about secret storing, cloudwatch vs. cloudtrail, what are notifications, different kinds of endpoints, how to create cloudwatch filters.

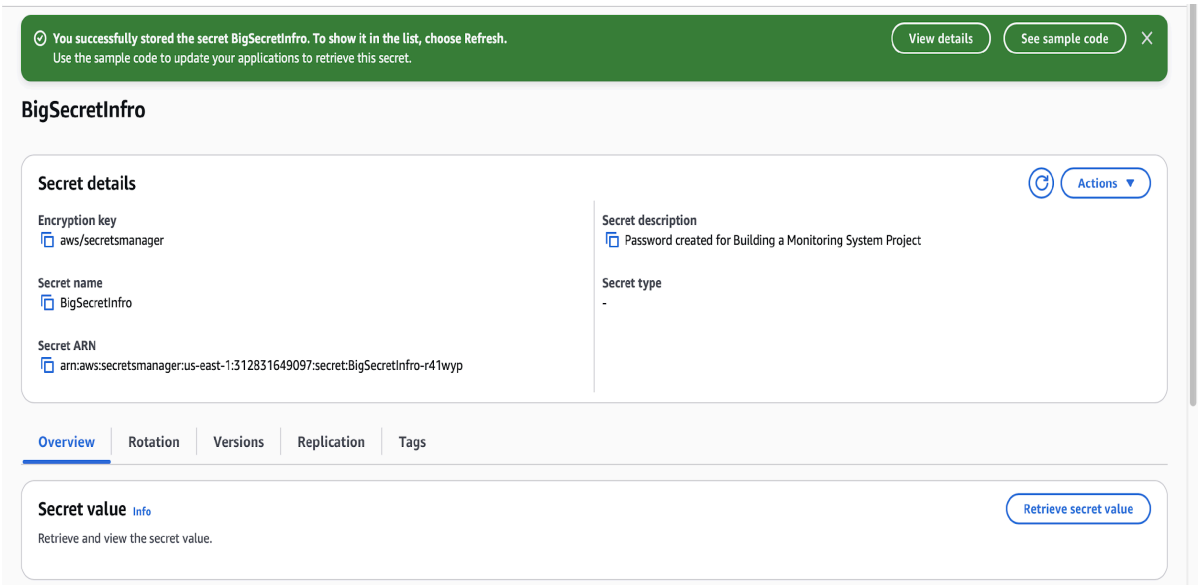
Project reflection

This project took me approximately 3 hours to compete. The most challenging part was troubleshooting why the email wasn't delivered.

Create a Secret

Secrets Manager is AWS' security service for storing secrets i.e. database credentials, account IDs, API keys… anything that is sensitive information that would cause damage/trouble if it got leaked and shouldn't be lying around in code.

To set up for my project, I created a secret called BigSecretInfo in Secrets Manager. This secret is a string that contains a statement from me: I like mangos.



Set Up CloudTrail

CloudTrail is a monitoring service – it's used to track events and activities in my AWS account. These logs are very helpful for security (i.e. detecting suspicious activity), compliance (i.e. proving that you're following the rules for something), and troubleshooting (figuring out what changed when something breaks).

CloudTrail events include types like management, data, insights and network activity events. In this project, I set up my trail to track Management events because accessing a secret falls into that category. It is not a data event (which captures high-volume actions performed on resources) because management events are free to track.

Read vs Write Activity

Read API activity involves accessing, reading, opening a resource. Write API activity involves creating, deleting, updating a resource. For this project, I ticked both to learn about both types of activities, but I really only need the Write activity (accessing a secret is considered a Write activity because of its importance).

Verifying CloudTrail

I retrieved my secret in two ways: First through the Secrets Manager console, where I could easily just select a “Retrieve secret value” button. The second way was using the AWS CLI, i.e. running a get-secret-value command in CloudShell.

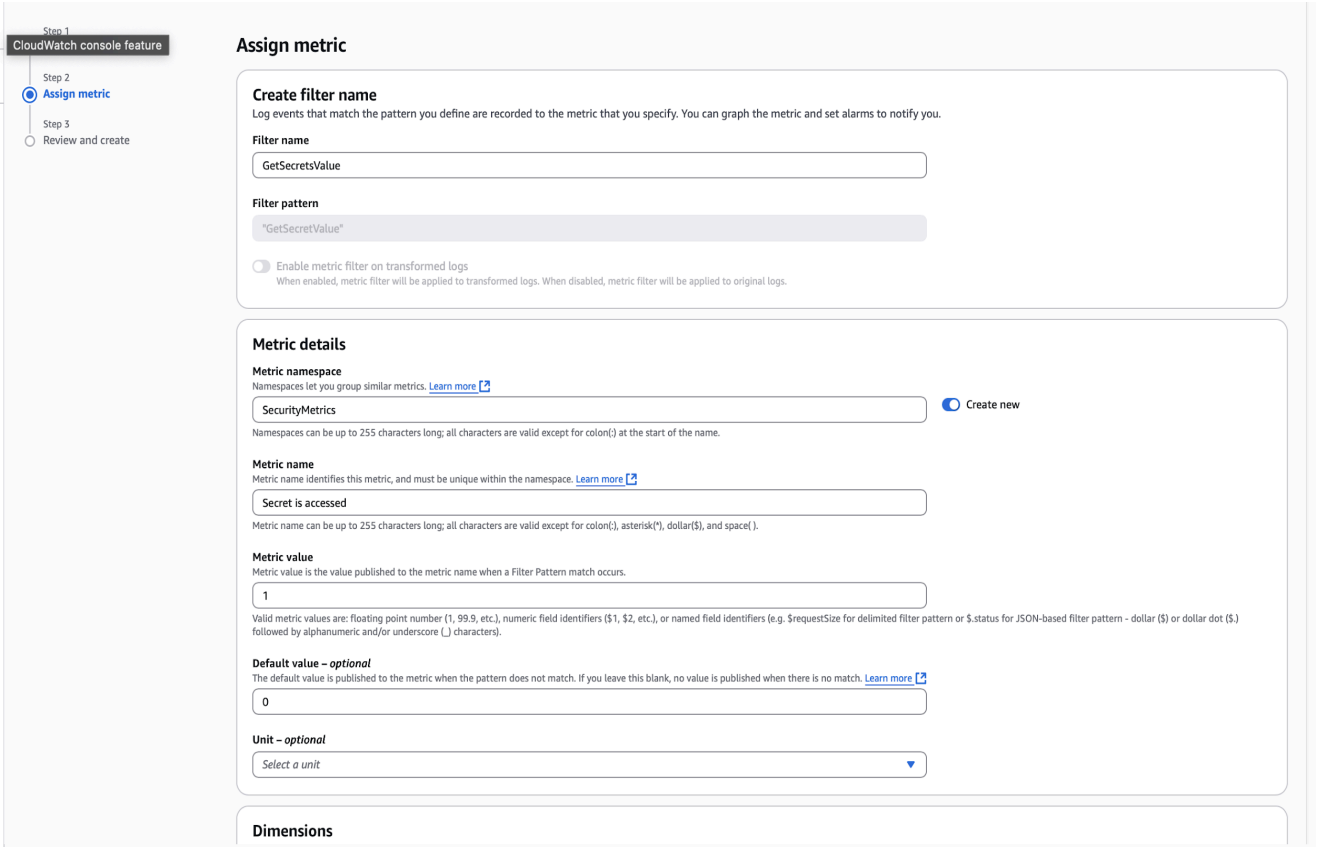
To analyze my CloudTrail events — i.e. see the event where I got my secret’s value — I visited the Event History in CloudTrail. I found that there was a GetSecretValue event tracked regardless of whether I did it over the CLI or over the console. This tells me that CloudTrail can definitely see and track when I open my Secrets Manager Key.

CloudWatch Metrics

CloudWatch Logs is a monitoring service that brings together logs from other AWS services (including CloudTrail) to help me analyze and create alarms. It's important for monitoring because I get to create insights and get alerted based on events that happen in my account.

CloudTrail’s Event History is useful for quickly reading management events that happened in the last 90 days, while CloudWatch Logs is better for combining and analyzing logs from different sources, accessing logs for longer than 90 days, and advanced filtering.

A CloudWatch metric is a specific way that I count or track events that are in a log group. When setting up a metric, the metric value represents how I increment or ‘count’ an event when it passes my filters (in my case, I want to increment my metric value by one whenever the secret is accessed.) The default value is used when the event we are tracking does not occur.

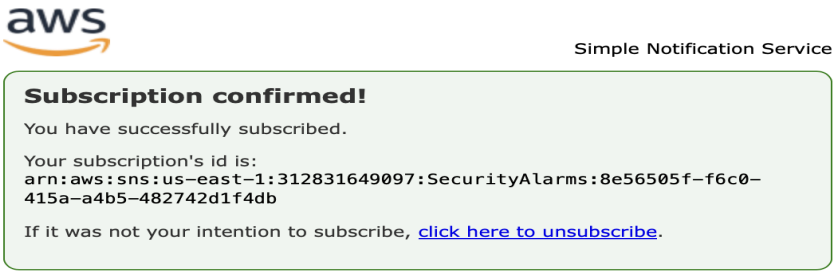

Step 1
CloudWatch console feature
Step 2
Assign metric
Step 3
Review and create
Assign metric
Create filter name
Log events that match the pattern you define are recorded to the metric that you specify. You can graph the metric and set alarms to notify you.
Filter name
GetSecretsValue
Filter pattern
"GetSecretValue"
Enable metric filter on transformed logs
When enabled, metric filter will be applied to transformed logs. When disabled, metric filter will be applied to original logs.
Metric details
Metric namespace
Namespaces let you group similar metrics. Learn more
SecurityMetrics
Create new
Namespaces can be up to 255 characters long; all characters are valid except for colon(:) at the start of the name.
Metric name
Metric name identifies this metric, and must be unique within the namespace. Learn more
Secret is accessed
Metric name can be up to 255 characters long; all characters are valid except for colon(:), asterisk(*), dollar($), and space( ).
Metric value
Metric value is the value published to the metric name when a Filter Pattern match occurs.
1
Valid metric values are: floating point number (1, 99.9, etc.), numeric field identifiers ($1, $2, etc.), or named field identifiers (e.g. $requestSize for delimited filter pattern or $.status for JSON-based filter pattern - dollar ($) or dollar dot ($.) followed by alphanumeric and/or underscore (_) characters).
Default value – optional
The default value is published to the metric when the pattern does not match. If you leave this blank, no value is published when there is no match. Learn more
0
Unit – optional
Select a unit
Dimensions

CloudWatch Alarm

A CloudWatch alarm is a feature and alert system in CloudWatch that's designed to “go off,” i.e. indicate when certain conditions have been met in my log group. I set my CloudWatch alarm threshold to be about how many times the GetSecretValue event happens in a 5-minute window, so the alarm will trigger when that threshold is reached.

I created an SNS topic along the way. An SNS topic is like a newsletter/broadcast channel that emails, phone numbers, functions, and apps can subscribe to (so they get notified when SNS has a new update to share). My SNS topic is set up to send me an email when my secret gets accessed.

AWS requires email confirmation because it will not automatically start emailing addresses that I subscribe to an SNS topic. This helps prevent any unwanted subscriptions for recipients (i.e. people who are receiving those emails).



Troubleshooting Notification Errors

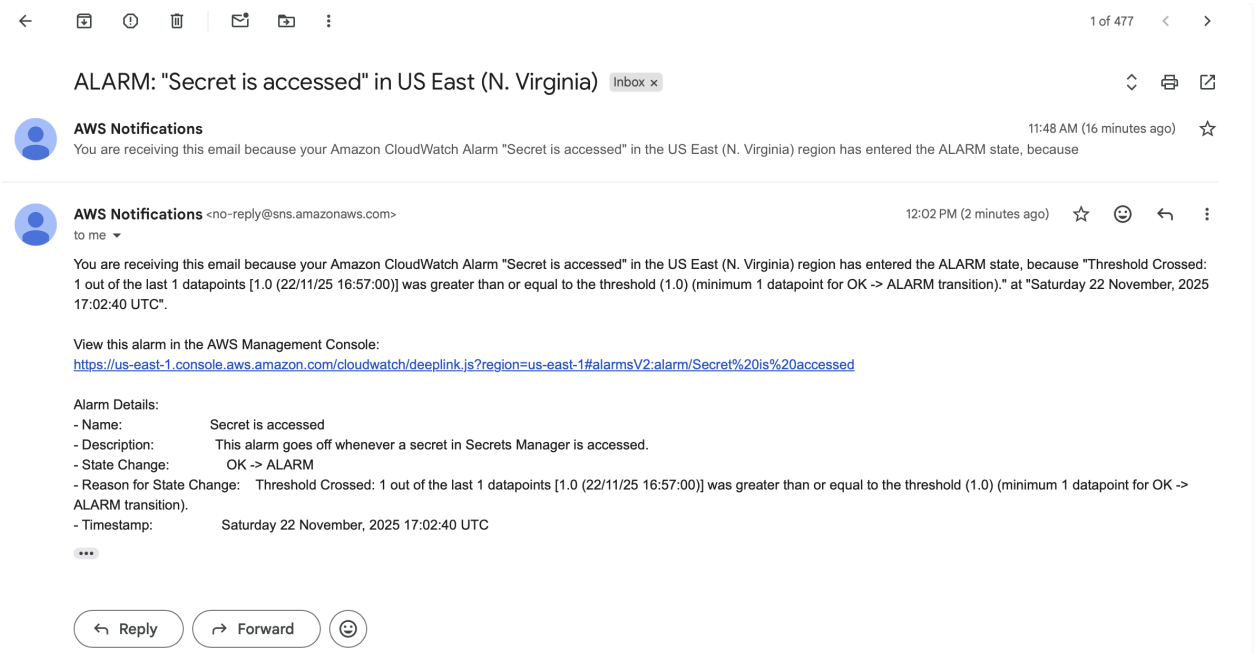
To test my monitoring system, I opened and accessed my secret again! The results weren’t successful — I didn’t get any emails or notifications about my secret getting accessed.

When troubleshooting the notification issues, I investigated every single part of my monitoring system — whether CloudTrail is picking up on events that are happening when I access my secret, whether CloudTrail is sending logs to CloudWatch, whether the filter is accidentally rejecting the correct events, whether the alarm gets triggered, and whether triggering the alarm sends an email.

I initially didn’t receive an email because CloudWatch was configured to use the wrong threshold. Instead of calculating the AVERAGE number of times a secret was accessed in a time period, it should’ve been using the SUM!

Success!

Once I confirmed that my system can successfully detect and alert when my secret is accessed, I checked my secret's value one more time. I received an email within 1–2 minutes of the event! My alarm in CloudWatch is also in the "In alarm" state.



Comparing CloudWatch with CloudTrail Notifications

In a project extension, I enabled SNS notification delivery in CloudTrail because this lets me evaluate CloudTrail vs. CloudWatch for notifying me about events like my secret getting accessed.

After enabling CloudTrail SNS notifications, my inbox was very quickly filled with new emails from SNS (as it was notified by CloudTrail). In terms of the usefulness of these emails, I thought that I was receiving LOTS of them, and the logs themselves don't show what happened in terms of the management events that occurred. I only see that new logs have been stored in my bucket.