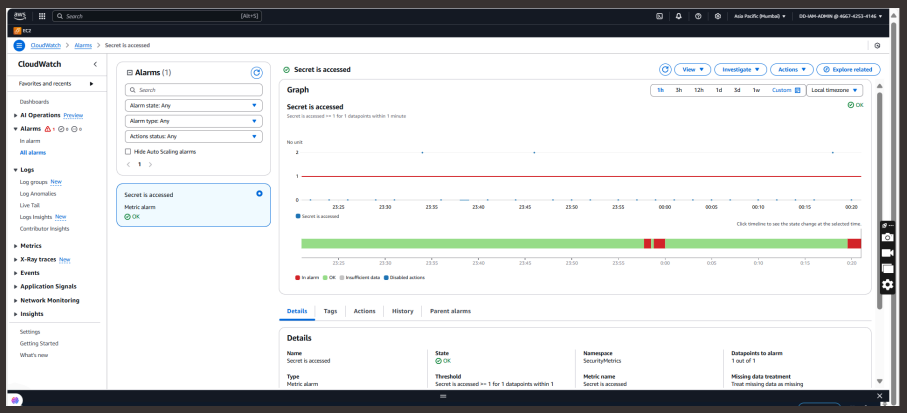
nextwork.org

# Build a Security Monitoring System

DI Dineshraj Dhanapathy



DI
**Dineshraj Dhanapathy**
NextWork Student


# Introducing Today's Project!

In this project, I will demonstrate how to monitor and respond to access attempts on secrets stored in AWS Secrets Manager. I'm doing this project to learn how to enhance security by detecting unauthorized or unexpected access. First, I will create a secret in Secrets Manager and enable AWS CloudTrail to log activity in my account. Then, I will test if CloudTrail successfully records access to the secret. Next, I will set up a CloudWatch filter to detect secret access events and configure an SNS topic and CloudWatch Alarm to send me email alerts. Finally, I'll test and troubleshoot the full setup to ensure the monitoring and alert system works end-to-end.

## Tools and concepts

Services I used were AWS CloudTrail, CloudWatch Logs, CloudWatch Alarms, and Amazon SNS. Key concepts I learnt include how CloudTrail captures API activity like GetSecretValue, how to send those logs to CloudWatch, and how to create metric filters to detect specific events. I also understood how to configure CloudWatch Alarms with the correct statistics (like Sum) and thresholds to trigger actions. Finally, I learnt how SNS delivers notifications and the importance of confirmed subscriptions.

## Project reflection

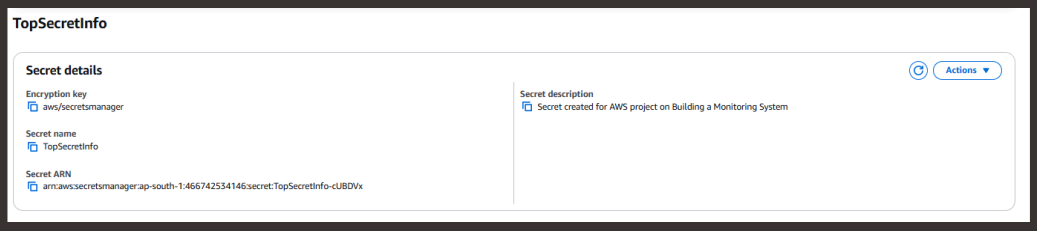
This project took me approximately 3 hours to complete. The most challenging part was troubleshooting why the CloudWatch alarm wasn't triggering—especially identifying that the metric was using the wrong statistic (average instead of sum). It was most rewarding to see the full alerting pipeline work end-to-end, from detecting the GetSecretValue event in CloudTrail to receiving an email notification via SNS, confirming my monitoring setup was successful.

DI
**Dineshraj Dhanapathy**
NextWork Student


# Create a Secret

Secrets Manager is a service from AWS that helps you securely store, manage, and access sensitive information like passwords, API keys, and database credentials. You could use Secrets Manager to avoid hardcoding secrets in your application code, which improves security and simplifies secret rotation. It also integrates with other AWS services and allows you to control access through IAM policies, helping ensure that only authorized users and applications can retrieve specific secrets.

To set up for my project, I created a secret called TopSecretInfo that contains a sample key-value pair such as a username and password. This secret simulates sensitive data, like credentials for a database or third-party service. It will be used to test whether AWS CloudTrail can detect and log access attempts, and to build a monitoring system that alerts me when the secret is accessed.



# Set Up CloudTrail

CloudTrail is a monitoring service that records all API activity across your AWS account—who did what, when, and from where. It helps with security monitoring, troubleshooting, and compliance by giving you a detailed log of every action taken in your environment. I set up a trail to tell CloudTrail exactly which events to track and where to store those event logs, such as in an S3 bucket. This trail ensures that any access to my AWS Secrets Manager secret is captured and can be reviewed later. Trails are customizable, and you can create multiple trails to monitor different types of activity across your AWS resources.

CloudTrail events include types like management events, data events, insights events, and network activity events, each giving insight into different kinds of API activity in your AWS account. Management events log actions that configure resources, such as creating an EC2 or accessing a secret—this is our focus in the project. Data events track high-volume operations like uploading to S3 or invoking a Lambda. Insights events highlight unusual activity patterns for better anomaly detection. Network activity events capture changes to network settings, such as VPC updates or subnet traffic.

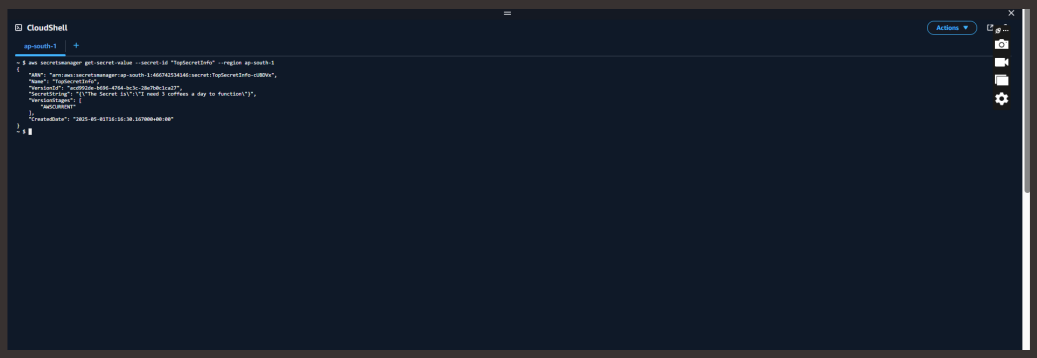
## Read vs Write Activity

Read API activity involves actions where someone views information without making any changes—like listing S3 buckets, describing EC2 instances, or viewing secret metadata. Write API activity involves actions that modify resources or trigger significant access—such as creating, updating, deleting resources, or retrieving the actual value of a secret. For this project, we need to monitor Write API activity because accessing the contents of a secret (not just viewing its metadata) is considered a Write action, and that's what we want to detect and alert on.

# Verifying CloudTrail

I retrieved the secret in two ways: First through the AWS Management Console, by navigating to AWS Secrets Manager and viewing the secret's value directly in the browser. Second, using AWS CloudShell, which provided a command-line terminal already authenticated to my AWS account. I used the AWS CLI command to fetch the secret securely. This method is faster, scriptable, and ideal for automation. Both methods allowed me to verify that secret access is being tracked by CloudTrail.

To analyze my CloudTrail events, I visited the Event history section in the CloudTrail console. I used lookup attributes to filter by event source as secretsmanager.amazonaws.com and event name as GetSecretValue. I found a recorded event showing that my secret was accessed. This tells me that CloudTrail successfully captured the secret access activity, confirming that my logging setup is working correctly and ready for monitoring and alerting.



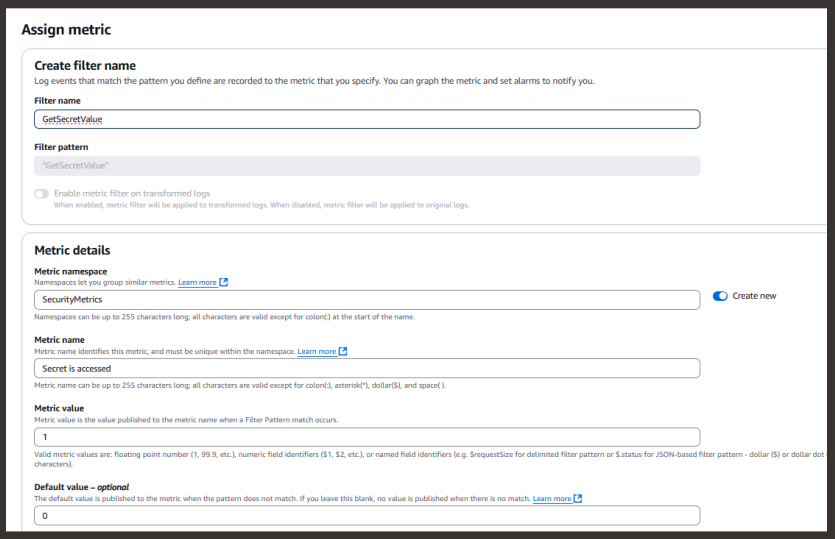
DI
**Dineshraj Dhanapathy**
NextWork Student


# CloudWatch Metrics

CloudWatch Logs is an AWS service that collects, stores, and monitors log data from various sources like CloudTrail, applications, servers, and network services. It's important for monitoring because it helps detect unusual behavior, troubleshoot issues, and analyze operational trends in real time. By centralizing logs from different AWS and non-AWS systems, it gives you visibility into your environment, enabling faster response to incidents and improved system reliability.

CloudTrail's Event History is useful for quickly viewing recent management events from the past 90 days in a searchable format, making it great for audits and basic tracking. While CloudWatch Logs are better for real-time monitoring, long-term storage, custom filtering, and triggering alerts. CloudWatch allows you to create metric filters and alarms based on specific patterns in your logs, enabling proactive responses to important security or operational events.

A CloudWatch metric is a numeric value that represents specific activity or behavior in your AWS environment. When setting up a metric, the metric value represents what gets recorded each time a filter matches a log event—like setting it to 1 for each secret access. Default value is used when there are no matching log events during a time period; setting it to 0 ensures gaps in activity are still shown in charts, giving you a full view of both active and idle periods in your monitoring dashboard.


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
GetSecretValue
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
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

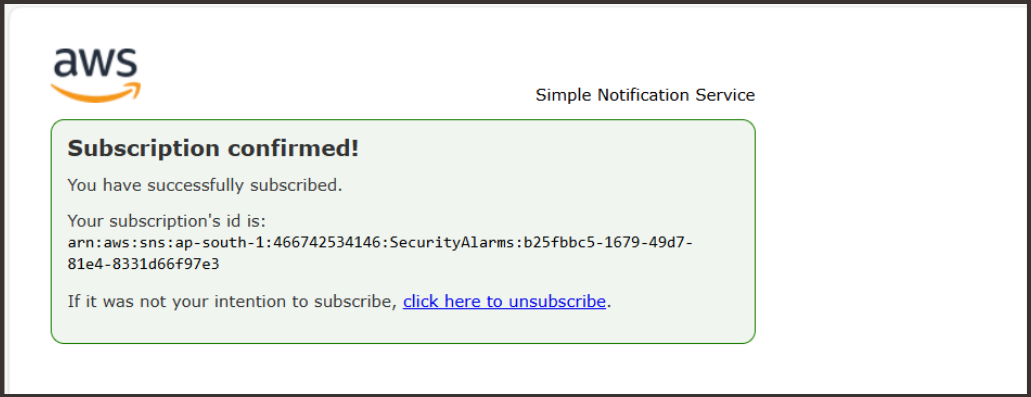
DI
**Dineshraj Dhanapathy**
NextWork Student


# CloudWatch Alarm

A CloudWatch alarm is a monitoring tool that watches a specific metric and performs an action when that metric meets a defined condition. I set my CloudWatch alarm threshold to trigger when the SecretIsAccessed metric is greater than or equal to 1 within a 5-minute period, so the alarm will trigger when even a single access to my secret occurs. This sensitive threshold is ideal for high-security scenarios where any secret access should be flagged immediately. In other environments, the threshold can be adjusted to avoid unnecessary alerts based on normal access patterns.

I created an SNS topic along the way. An SNS topic is a communication channel that allows you to send messages to multiple subscribers at once. My SNS topic is set up to send an email notification whenever the CloudWatch alarm detects access to my secret. This makes it easy to stay informed in real time. Later, I can expand it to include more subscribers—like SMS, Lambda functions, or Slack—without changing how the alert is triggered, giving me a flexible and scalable alerting system.

AWS requires email confirmation because it ensures that the recipient has consented to receive messages from the SNS topic. This helps prevent unauthorized or spam notifications from being sent to people who didn’t request them. By confirming the subscription, the email owner verifies that they trust the source and want to receive alerts. It's an important security and compliance step to protect users' privacy and avoid accidental or malicious misuse of notification services.


aws
Simple Notification Service
Subscription confirmed!
You have successfully subscribed.
Your subscription's id is:
arn:aws:sns:ap-south-1:466742534146:SecurityAlarms:b25fbbc5-1679-49d7-81e4-8331d66f97e3
If it was not your intention to subscribe, click here to unsubscribe.

# Troubleshooting Notification Errors

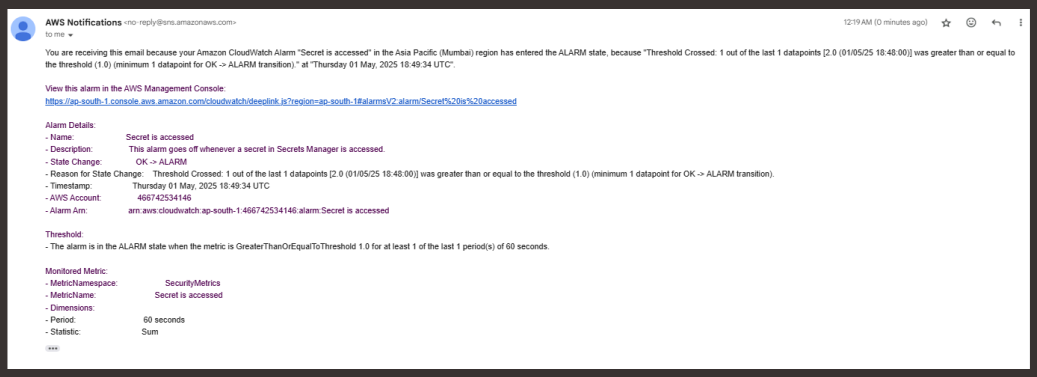
To test my monitoring system, I deliberately retrieved the secret value again to trigger the configured CloudWatch alarm. The results were that the alarm state changed as expected, but I did not receive any notification. This indicates a possible issue in the alerting pipeline. I began troubleshooting by verifying the SNS topic subscription status, checking if the email was confirmed, reviewing CloudWatch alarm actions, and ensuring IAM permissions were correctly configured to allow the alarm to publish messages to the SNS topic.

When troubleshooting the notification issues, I checked five key areas: First, I verified if CloudTrail was recording the GetSecretValue event by reviewing recent event history. Second, I confirmed that CloudTrail was properly configured to send logs to CloudWatch Logs, ensuring the correct log group was selected. Third, I reviewed the CloudWatch metric filter to ensure it was correctly set to detect the GetSecretValue event. Fourth, I checked whether CloudWatch Alarm was linked to the metric filter and properly configured to trigger. Lastly, I verified that SNS had confirmed email subscriptions and tested delivery to ensure emails weren't ending up in spam or blocked.

I initially didn't receive an email before because the CloudWatch alarm was configured with the wrong threshold—it was calculating the average number of times the secret was accessed, instead of using the sum over the specified period. As a result, the alarm never crossed the threshold to trigger. The key solution was updating the metric filter and alarm to use the "Sum" statistic, which correctly counts the total access events within the period, ensuring the alarm triggers when the secret is accessed multiple times and notifications are sent via SNS.

# Success!

To validate that my monitoring system works, I checked all components step by step. I retrieved the secret value multiple times, waited over 5 minutes, and ensured CloudTrail was logging the events and sending them to CloudWatch Logs. I verified the metric filter was correctly detecting the GetSecretValue event and that the alarm was set to use the "Sum" statistic. I received a notification email from SNS, confirming the entire pipeline—from event to alert—was functioning properly.



DI
**Dineshraj Dhanapathy**
NextWork Student


# Comparing CloudWatch with CloudTrail Notifications

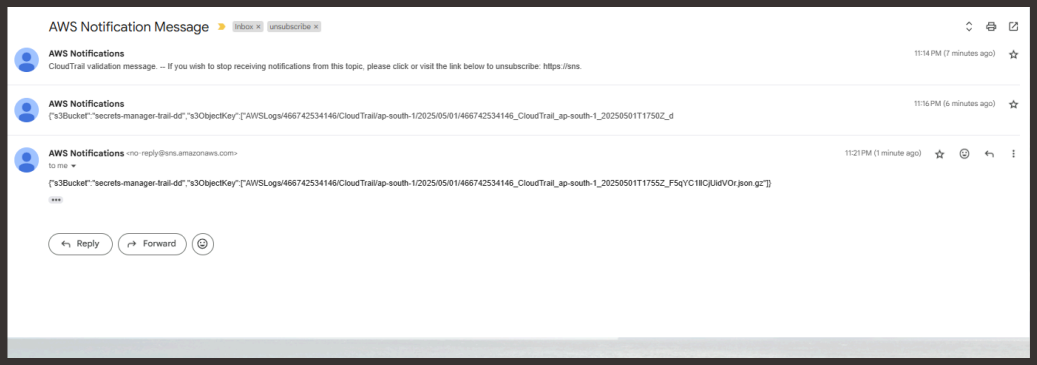
In a project extension, I updated my CloudTrail configurations to enable SNS notification delivery because I wanted to get notified whenever a new log file was delivered to my S3 bucket. This adds an extra layer of visibility into my account's activity and complements the CloudWatch Alarm, which only triggers on specific log patterns. With both in place, I can monitor both general log delivery and targeted events like secret access.

After enabling CloudTrail SNS notifications, my inbox began receiving emails each time a new log file was delivered to my S3 bucket. In terms of the usefulness of these emails, I thought they provided a helpful way to confirm that logging is active and working, but they can become overwhelming if not filtered or managed properly—especially in high-activity accounts. They're best used for audit tracking or automation triggers rather than real-time monitoring of specific events.

DI

**Dineshraj Dhanapathy**
NextWork Student





nextwork.org

# The place to learn & showcase your skills

Check out nextwork.org for more projects