# AWS CloudTrail Log Monitoring: Integrating CloudTrail Logs Into CloudWatch

## Contents

## Introduction

The purpose of this project was to implement a comprehensive monitoring solution for an Amazon EC2 instance, specifically tracking and alerting administrators when the instance is stopped more than once within a defined period. We achieved this by integrating AWS CloudTrail with CloudWatch to capture EC2 stop events and using CloudWatch Alarms and Amazon SNS for real-time notifications.

The implementation included setting up **AWS CloudTrail** to log API activity related to the EC2 instance, including **StopInstances** events. These logs were sent to **CloudWatch Logs**, where we created **Metric Filters** to track how many times the EC2 instance was stopped. When the instance was stopped more than once, a **CloudWatch Alarm** was triggered, sending email alerts via **Amazon SNS** to the administrators.

Additionally, we enhanced the monitoring by enabling **CloudWatch Metric Anomaly Detection** to identify abnormal patterns in the EC2 instance's stop events, and we visualized the instance's status and metric activity in real time using **CloudWatch Dashboards**.

This project provides a robust system for monitoring critical EC2 operations, ensuring quick detection and response to potential issues such as repeated stops, which may indicate operational problems or unauthorized activity.

## AWS Services Used

### 1. AWS CloudTrail

**Purpose**: AWS CloudTrail logs all API calls made in an AWS account, capturing changes and actions performed on AWS resources, such as launching, stopping, and terminating EC2 instances.

- **Usage in the Project**: CloudTrail logs all actions related to the EC2 instance, particularly StopInstances events, and sends the logs to CloudWatch Logs for further processing. These logs are critical for tracking the stopping of the instance.
- **Benefits**:
    - Comprehensive tracking of all AWS API calls for auditing and troubleshooting.
    - Provides detailed records of when the EC2 instance is stopped and by whom.
    - Useful for detecting suspicious or unexpected behavior.

## 2. Amazon CloudWatch

Several CloudWatch features are leveraged in this project:

### a. CloudWatch Logs

**Purpose**: CloudWatch Logs collect and store logs from AWS services like CloudTrail. Logs can be queried and analyzed to gain insights into the operational state of AWS resources.

- **Usage in the Project**: CloudTrail logs related to the stopping of the EC2 instance are sent to CloudWatch Logs. This allows for real-time log monitoring and the ability to detect stop events, which will be used to trigger alarms if the instance stops more than once.
- **Benefits**:
    - Centralized log storage with long-term retention for compliance and troubleshooting.
    - Supports real-time log streaming, enabling quick detection of important events like the stopping of EC2 instances.
    - Easy integration with CloudWatch Alarms for real-time notifications.

### b. CloudWatch Log Insights

**Purpose**: CloudWatch Log Insights allows for querying log data and analyzing patterns over time using built-in or custom queries.

- **Usage in the Project**: We used **built-in queries** in CloudWatch Log Insights to track the StopInstances events, which enabled us to analyze when the EC2 instance was stopped.
- **Benefits**:
    - Quick access to log data using queries, helping identify patterns or detect incidents.
    - Provides insights into the health and state of the EC2 instance, allowing for further analysis.

### c. CloudWatch Metric Filters

**Purpose**: CloudWatch Metric Filters convert log data into actionable metrics. In this project, the Metric Filter was configured to track how many times the EC2 instance was stopped.

- **Usage in the Project**: A Metric Filter was created to detect StopInstances events in the CloudWatch Logs and generate a custom metric. This metric was then tracked to monitor the number of stop events.
- **Benefits**:
    - Automatic generation of metrics from log data.
    - Enables detailed monitoring of specific log events like EC2 stop events.

### d. CloudWatch Alarms

**Purpose**: CloudWatch Alarms monitor metrics and notify administrators when thresholds are breached.

- **Usage in the Project**: A CloudWatch Alarm was set up to trigger when the EC2 instance was stopped more than once in a given period. The alarm sent a notification via Amazon SNS.
- **Benefits**:
    - Provides real-time alerting when predefined conditions are met.
    - Supports integration with SNS for automated notifications to administrators.

### e. CloudWatch Metric Anomaly Detection

**Purpose**: CloudWatch Metric Anomaly Detection uses machine learning to identify unusual patterns in metrics based on historical data. This helps detect when the EC2 instance stops in an abnormal pattern.

- **Usage in the Project**: We enabled **Anomaly Detection** for the metric tracking EC2 stop events. This feature was configured to automatically detect deviations from expected behavior.
- **Benefits**:
    - Advanced monitoring using machine learning to detect unexpected patterns.
    - Helps catch irregular or suspicious activity, such as unauthorized or frequent EC2 stops.

### f. CloudWatch Dashboards

**Purpose**: CloudWatch Dashboards provide a customizable, real-time interface for visualizing key metrics and alarms.

- **Usage in the Project**: A CloudWatch Dashboard was created to display the custom metric tracking EC2 stops and the alarm status. The dashboard included a line graph to visualize stop events over time.
- **Benefits**:

    - Real-time visibility of key metrics and alarms.
    - Provides a single view for monitoring the health and state of AWS resources like EC2 instances.

## 3. Amazon Simple Notification Service (SNS)

**Purpose**: Amazon SNS is a messaging service that sends notifications (via email, SMS, or mobile push) when events or alarms are triggered in the AWS environment.

- **Usage in the Project**: When a CloudWatch Alarm is triggered (due to the EC2 instance stopping more than once), SNS sends an alert to system administrators. This ensures that administrators are informed immediately and can take corrective actions to prevent further issues.
- **Benefits**:
    - Real-time notifications via multiple channels, ensuring quick response.
    - Seamless integration with CloudWatch Alarms for automatic alerting.
    - Scalable, allowing notifications to be sent to multiple stakeholders simultaneously.

## 4. Amazon EC2

**Purpose**: Amazon EC2 provides scalable virtual machines in the cloud. It allows users to run applications in a highly flexible environment, with full control over the instances' lifecycle.

- **Usage in the Project**: The EC2 instance is the primary resource being monitored. CloudTrail logs all lifecycle events, such as starting, stopping, and terminating the instance. This project focuses specifically on monitoring the StopInstances API call, which logs when the instance is stopped.
- **Benefits**:
    - Highly flexible and scalable, allowing dynamic management of computing resources.
    - Full control over the instance lifecycle (starting, stopping, and terminating).
    - Supports integration with AWS monitoring and logging services for complete operational visibility.

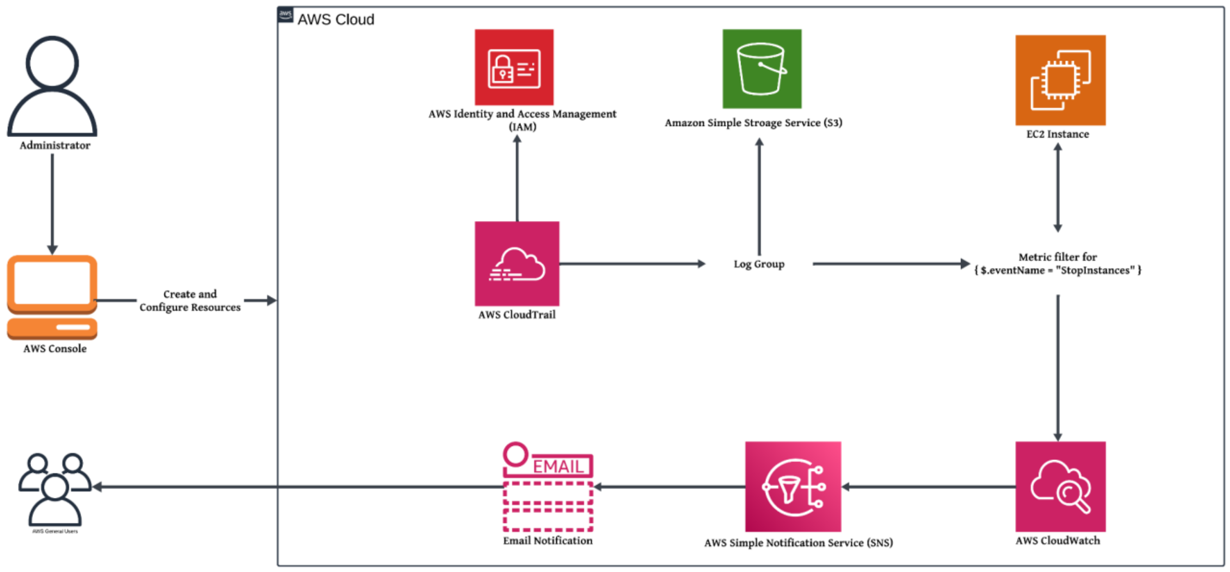
## 5. Amazon S3 Bucket

S3 was created automatically and used to store logs and backups efficiently, allowing us to retain log data for compliance and security purposes.

**6. AWS Identity and Access Management (IAM)**:

IAM provided us with secure access control over all the AWS services and resources. We enforced the principle of least privilege by ensuring that only authorized users could interact with the CloudTrail logs, CloudWatch, and SNS services.

## System Architecture


AWS Cloud
Administrator
AWS Console
Create and Configure Resources
AWS Identity and Access Management (IAM)
Amazon Simple Stroage Service (S3)
EC2 Instance
AWS CloudTrail
Log Group
Metric filter for { $.eventName = "StopInstances" }
EMAIL
Email Notification
AWS Simple Notification Service (SNS)
AWS CloudWatch
AWS General Users

The architecture diagram illustrates the workflow of monitoring an EC2 instance and alerting administrators when it is stopped more than once. Below is a breakdown of each component in the diagram and its role in the overall system.

The system architecture is designed to monitor and track when an Amazon EC2 instance is stopped, and to alert administrators when it is stopped more than once within a specified period. The following components are part of this architecture:

1. **User**:
   - The user logs in to the AWS Management Console to create and configure resources such as EC2 instances, CloudTrail, CloudWatch, and SNS. The user also receives notifications when the EC2 instance is stopped multiple times.
2. **AWS CloudTrail**:
   - CloudTrail logs API activity within the AWS account, specifically focusing on **StopInstances** events related to the EC2 instance. These logs capture when and by whom the instance was stopped, which is vital for tracking lifecycle events. A role was created to enable permissions on CloudTrail to access CloudWatch Resources.
3. **Log Group (CloudWatch Logs)**:

   - The logs generated by CloudTrail are sent to a **CloudWatch Log Group**. This serves as the central location where all logs related to the EC2 instance’s stop events are stored and analyzed.
4. **Metric Filter**:
   - A **CloudWatch Metric Filter** is applied to the log group to detect the **StopInstances** events in the log stream. Every time the EC2 instance is stopped, this filter increments a custom metric. This metric tracks how many times the instance has been stopped over a given period.
5. **CloudWatch Alarm**:
   - A **CloudWatch Alarm** is configured to monitor the custom metric created by the Metric Filter. If the metric exceeds the threshold (e.g., more than one stop within 24 hours), the alarm is triggered. This alarm ensures that repeated stops of the EC2 instance are quickly detected.
6. **Amazon SNS**:
   - When the CloudWatch Alarm is triggered, it sends a notification via **Amazon SNS** to subscribed administrators. Notifications are sent via email, allowing the team to investigate the repeated stops and take appropriate action.
7. **Email Notification to User**:
   - The user receives an email notification from SNS informing them that the EC2 instance has been stopped multiple times. This provides the user with real-time awareness of the issue, enabling them to take corrective action.
8. **CloudWatch Metric Anomaly Detection**:
   - **Anomaly Detection** is enabled on the custom metric tracking EC2 stop events. This feature uses machine learning to detect deviations from the normal behavior of the instance. If the instance exhibits unusual stop patterns that do not align with historical data, administrators are alerted to possible underlying issues.
9. **CloudWatch Dashboard**:
   - A **CloudWatch Dashboard** provides real-time visual monitoring of the EC2 instance's activity. The dashboard includes a graph displaying the custom metric for stop events and the status of the alarm. This visualization helps administrators monitor the health of the instance and quickly identify abnormal activity.

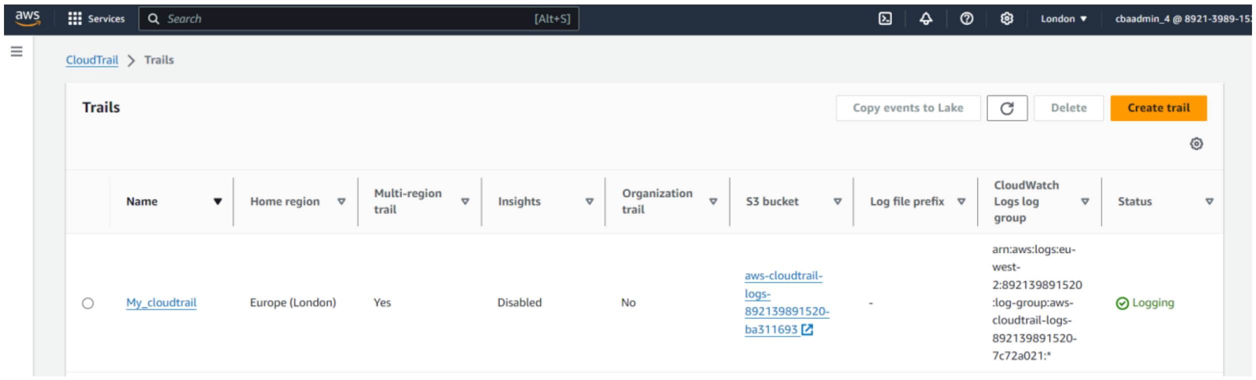
## Implementation

### 1. Creating a CloudTrail

The first step in monitoring the EC2 instance is to set up **AWS CloudTrail**, which logs all API activity. The process involved:

**Creating the Trail**:

- We enabled AWS CloudTrail to log all events (management events) related to the EC2 instance, particularly focusing on instance start and stop operations.
- The logs were configured to be sent to a CloudWatch Log Group for further processing and monitoring.



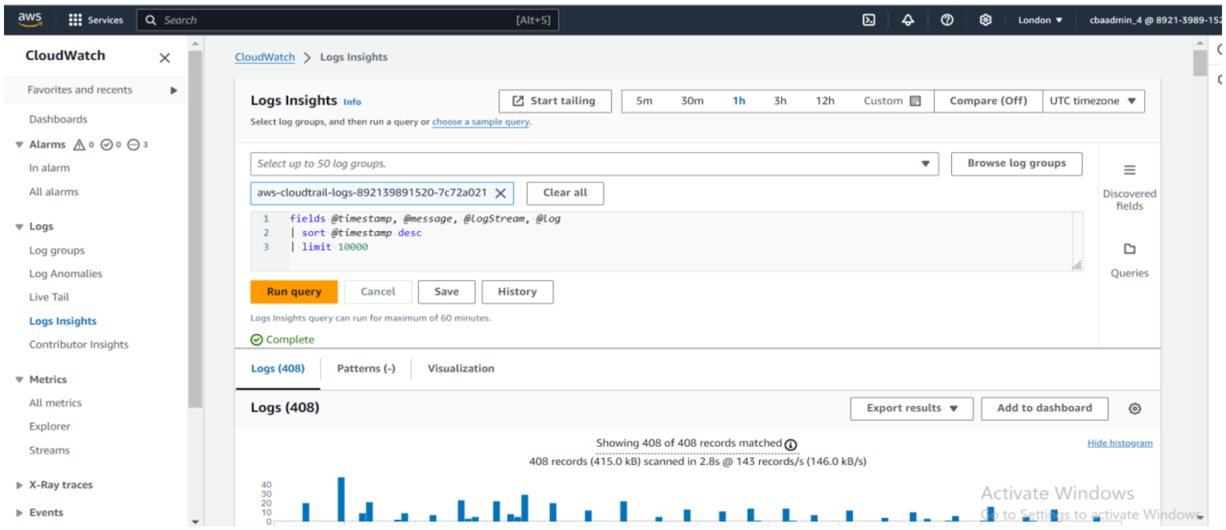
| | Name | Home region | Multi-region trail | Insights | Organization trail | S3 bucket | Log file prefix | CloudWatch Logs log group | Status |
|---|---|---|---|---|---|---|---|---|---|
| ○ | My_cloudtrail | Europe (London) | Yes | Disabled | No | aws-cloudtrail-logs-892139891520-ba311693 | - | arn:aws:logs:eu-west-2:892139891520:log-group:aws-cloudtrail-logs-892139891520-7c72a021:* | Logging |

## 2. Using CloudWatch Log Insights

**CloudWatch Log Insights** is a powerful tool for querying and analyzing logs stored in CloudWatch. In this step:

- We navigated to **Logs** and selected **Log Insights**.
- From the **Select log group(s)** dropdown, we chose the CloudWatch Log Group where CloudTrail logs are being sent.
- Instead of writing custom queries, we used **built-in queries** provided by AWS CloudWatch to detect key events related to EC2 instances (e.g., instance stop events).
- After selecting the built-in query, we clicked **Run query**, and the results were displayed. This allowed us to visualize and analyze the EC2 stop events.

The query results were used to monitor security incidents and operational issues.



Select log groups, and then run a query or choose a sample query.

Select up to 50 log groups.

aws-cloudtrail-logs-892139891520-7c72a021

```
fields @timestamp, @message, @logStream, @log
| sort @timestamp desc
| limit 10000
```

Logs Insights query can run for maximum of 60 minutes.

Complete

Logs (408)

Showing 408 of 408 records matched

408 records (415.0 kB) scanned in 2.8s @ 143 records/s (146.0 kB/s)

## 3. Creating Metric Filters for Log Groups in CloudWatch

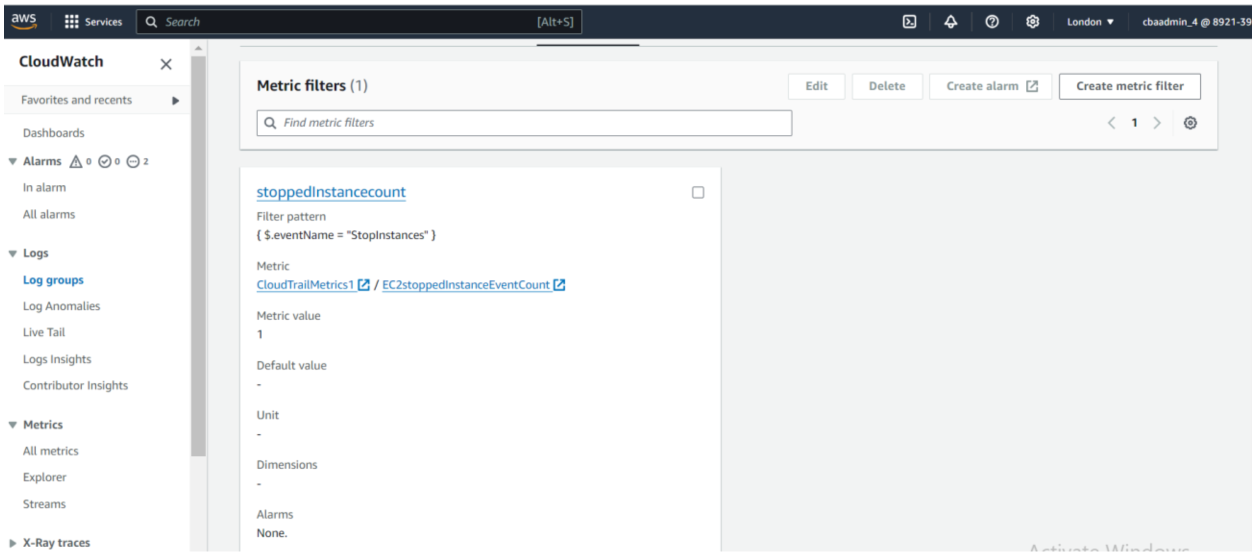
To convert the EC2 stop events into actionable metrics, we set up **CloudWatch Metric Filters**:

- We navigated to the **Log Group** where CloudTrail logs are stored.

A Metric Filter was created using the following pattern to detect EC2 stop events:
{ $.eventName = "StopInstances" }

- This Metric Filter converted each **StopInstances** event from the CloudTrail logs into a custom metric that can be tracked over time.

Once this filter was in place, the system began tracking how many times the EC2 instance was stopped.



Metric filters (1)

stoppedInstancecount

Filter pattern
{ $.eventName = "StopInstances" }

Metric
CloudTrailMetrics1 / EC2stoppedInstanceEventCount

Metric value
1

Default value
-

Unit
-

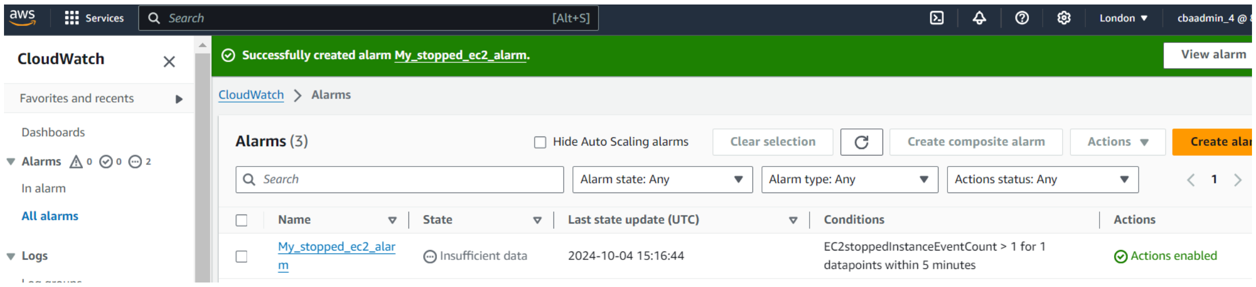
Dimensions
-

Alarms
None.

## 4. Creating an Alarm

With the custom metric created, we set up a **CloudWatch Alarm** to monitor EC2 stop events:

- The alarm was configured to trigger if the custom metric (tracking stop events) detected more than one stop in a specified period (e.g., 24 hours).
- The alarm sent notifications via **Amazon SNS** to alert the team when the EC2 instance was stopped multiple times.


Successfully created alarm My_stopped_ec2_alarm.

CloudWatch > Alarms

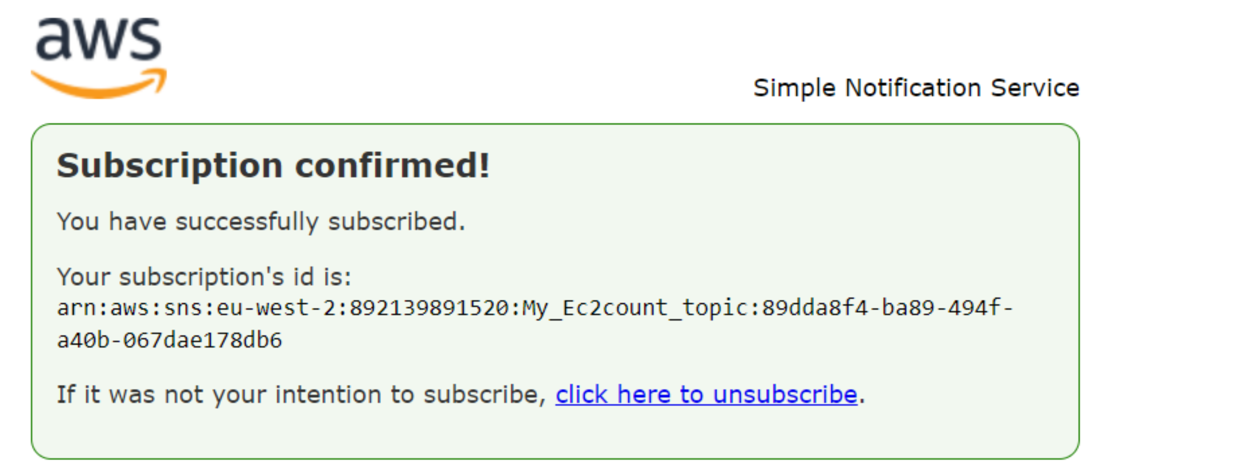
Alarms (3)

| | Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|---|
| | My_stopped_ec2_alarm | Insufficient data | 2024-10-04 15:16:44 | EC2stoppedInstanceEventCount > 1 for 1 datapoints within 5 minutes | Actions enabled |

## 5. Amazon SNS for Notifications

After configuring the CloudWatch Alarm, we set up **Amazon SNS** for notifications:

- An SNS topic was created, and email addresses of administrators were subscribed to this topic.
- When the CloudWatch Alarm was triggered (due to repeated EC2 stop events), SNS sent an email notification to alert the team. This allowed immediate intervention.


Simple Notification Service

**Subscription confirmed!**

You have successfully subscribed.

Your subscription's id is:
arn:aws:sns:eu-west-2:892139891520:My_Ec2count_topic:89dda8f4-ba89-494f-a40b-067dae178db6

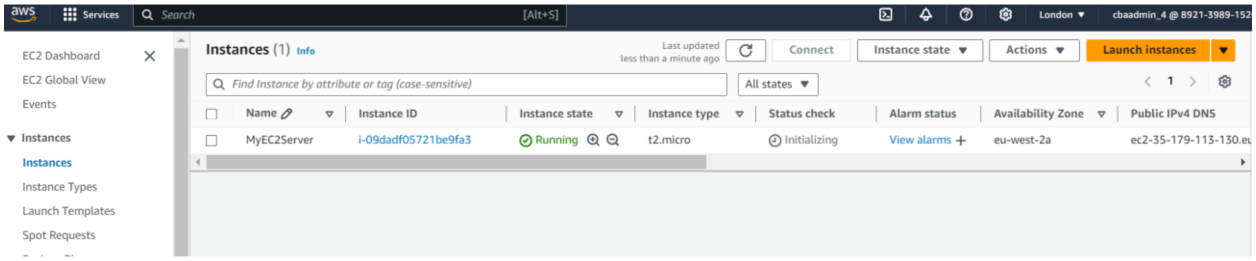
If it was not your intention to subscribe, click here to unsubscribe.

## 6. Creating an EC2 Instance to Trigger the Alarm

To test the monitoring setup, we created an EC2 instance that would be used to trigger the alarm.

- The EC2 instance was first **started** and allowed to run.
- Then, we manually **stopped and restarted the EC2 instance two times** to trigger the CloudWatch Alarm.

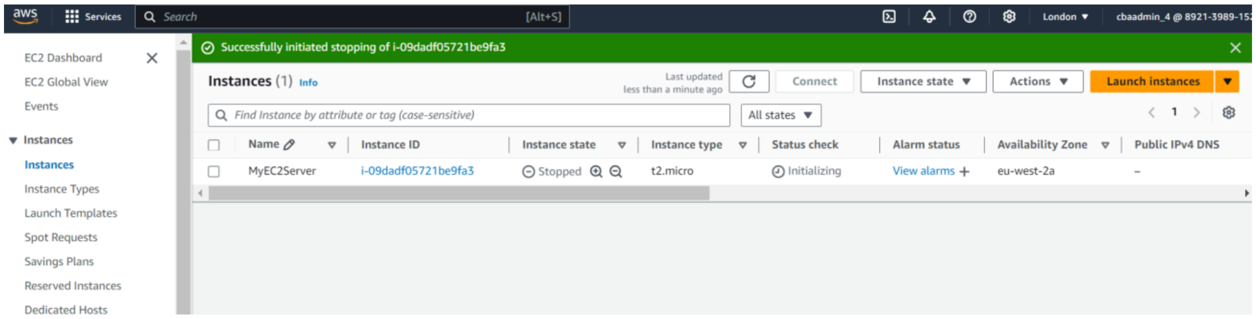
### Instance started



Instances (1) Info

| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS |
|---|---|---|---|---|---|---|---|
| MyEC2Server | i-09dadf05721be9fa3 | Running | t2.micro | Initializing | View alarms + | eu-west-2a | ec2-35-179-113-130.eu |

## Instance stopped and restarted 2 times



Successfully initiated stopping of i-09dadf05721be9fa3

Instances (1) Info

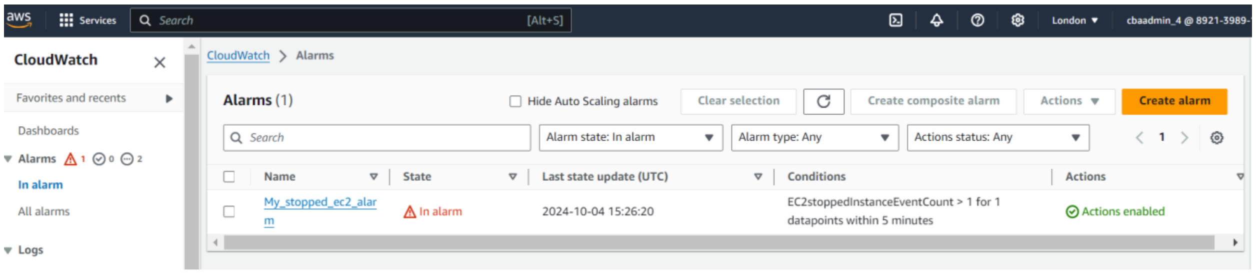
| Name | Instance ID | Instance state | Instance type | Status check | Alarm status | Availability Zone | Public IPv4 DNS |
|---|---|---|---|---|---|---|---|
| MyEC2Server | i-09dadf05721be9fa3 | Stopped | t2.micro | Initializing | View alarms + | eu-west-2a | – |

## 7. Alarm and Notifications

After stopping the EC2 instance multiple times, the **CloudWatch Alarm** was triggered:

- The custom metric (tracking EC2 stops) exceeded the threshold (more than one stop), which activated the alarm.
- The alarm sent an **email notification** to administrators via SNS, informing them of the repeated EC2 stops.

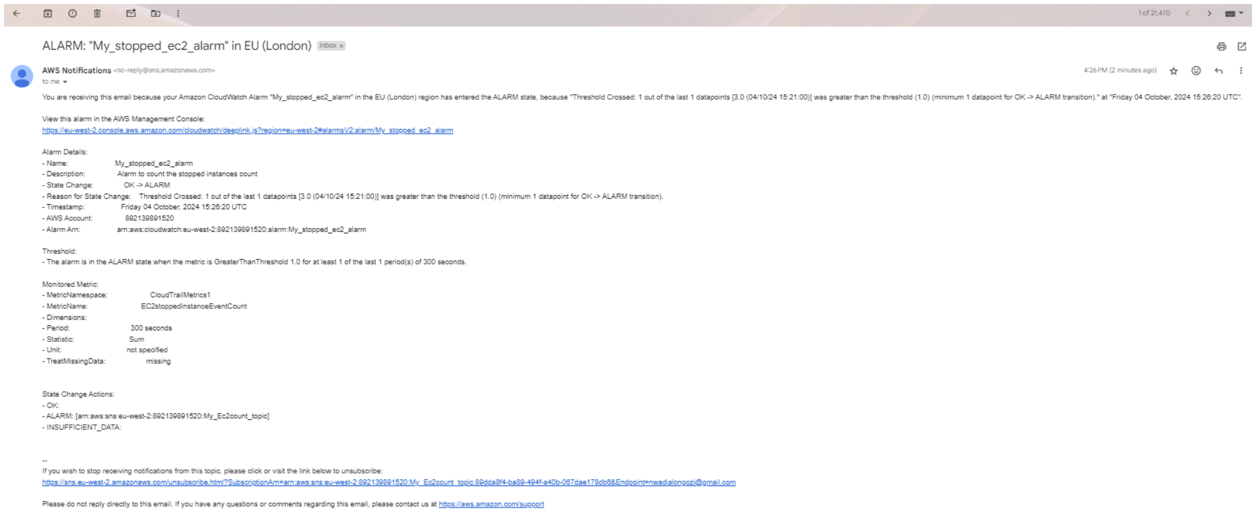
## Alarm in Cloudwatch



Alarms (1)

| Name | State | Last state update (UTC) | Conditions | Actions |
|---|---|---|---|---|
| My_stopped_ec2_alarm | In alarm | 2024-10-04 15:26:20 | EC2stoppedInstanceEventCount > 1 for 1 datapoints within 5 minutes | Actions enabled |

## Alarm notification in email



ALARM: "My_stopped_ec2_alarm" in EU (London)

AWS Notifications <no-reply@sns.amazonaws.com>
to me

4:26 PM (2 minutes ago)

You are receiving this email because your Amazon CloudWatch Alarm "My_stopped_ec2_alarm" in the EU (London) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [3.0 (04/10/24 15:21:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Friday 04 October, 2024 15:26:20 UTC".

View this alarm in the AWS Management Console:
https://eu-west-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=eu-west-2#alarmsV2:alarm/My_stopped_ec2_alarm

```
Alarm Details:
- Name:                      My_stopped_ec2_alarm
- Description:               Alarm to count the stopped instances count
- State Change:              OK -> ALARM
- Reason for State Change:   Threshold Crossed: 1 out of the last 1 datapoints [3.0 (04/10/24 15:21:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp:                 Friday 04 October, 2024 15:26:20 UTC
- AWS Account:               892139891520
- Alarm Arn:                 arn:aws:cloudwatch:eu-west-2:892139891520:alarm:My_stopped_ec2_alarm

Threshold:
- The alarm is in the ALARM state when the metric is GreaterThanThreshold 1.0 for at least 1 of the last 1 period(s) of 300 seconds.

Monitored Metric:
- MetricNamespace:           CloudTrailMetrics1
- MetricName:                EC2stoppedInstanceEventCount
- Dimensions:
- Period:                    300 seconds
- Statistic:                 Sum
- Unit:                      not specified
- TreatMissingData:          missing


State Change Actions:
- OK:
- ALARM: [arn:aws:sns:eu-west-2:892139891520:My_Ec2count_topic]
- INSUFFICIENT_DATA:
```

--
If you wish to stop receiving notifications from this topic, please click or visit the link below to unsubscribe:
https://sns.eu-west-2.amazonaws.com/unsubscribe.html?SubscriptionArn=arn:aws:sns:eu-west-2:892139891520:My_Ec2count_topic:89dda8f4-ba89-494f-a40b-067dae178db6&Endpoint=nwadialongozi@gmail.com

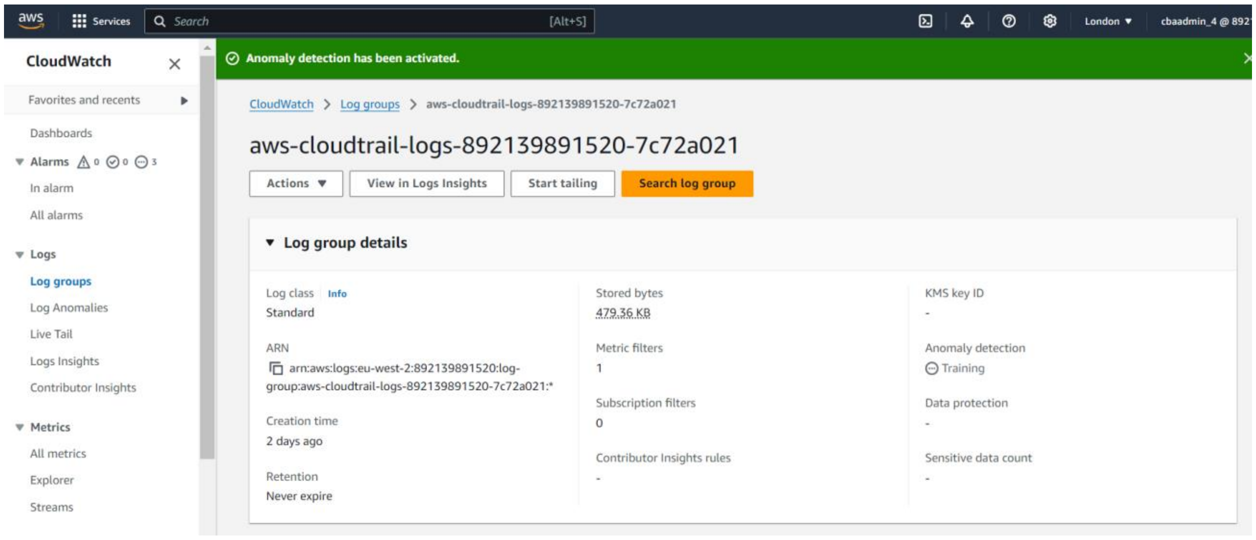
Please do not reply directly to this email. If you have any questions or comments regarding this email, please contact us at https://aws.amazon.com/support

## 8. Setting Up CloudWatch Metric Anomaly Detection

In addition to creating alarms, we enabled **CloudWatch Metric Anomaly Detection** for more sophisticated monitoring:

- We went to **Log Groups** and selected the concerned log group.
- Under **Actions**, we selected **Anomaly Detection** and clicked on **Configure**.
- Anomaly detection sensitivity was adjusted based on historical data to ensure that unusual patterns in EC2 stops would be detected automatically.

This added another layer of monitoring, using machine learning to detect unexpected behavior in instance stops.



Anomaly detection has been activated.

aws-cloudtrail-logs-892139891520-7c72a021

Log group details

| | | |
|---|---|---|
| Log class Info<br>Standard | Stored bytes<br>479.36 KB | KMS key ID<br>- |
| ARN<br>arn:aws:logs:eu-west-2:892139891520:log-group:aws-cloudtrail-logs-892139891520-7c72a021:* | Metric filters<br>1 | Anomaly detection<br>Training |
| Creation time<br>2 days ago | Subscription filters<br>0 | Data protection<br>- |
| Retention<br>Never expire | Contributor Insights rules<br>- | Sensitive data count<br>- |

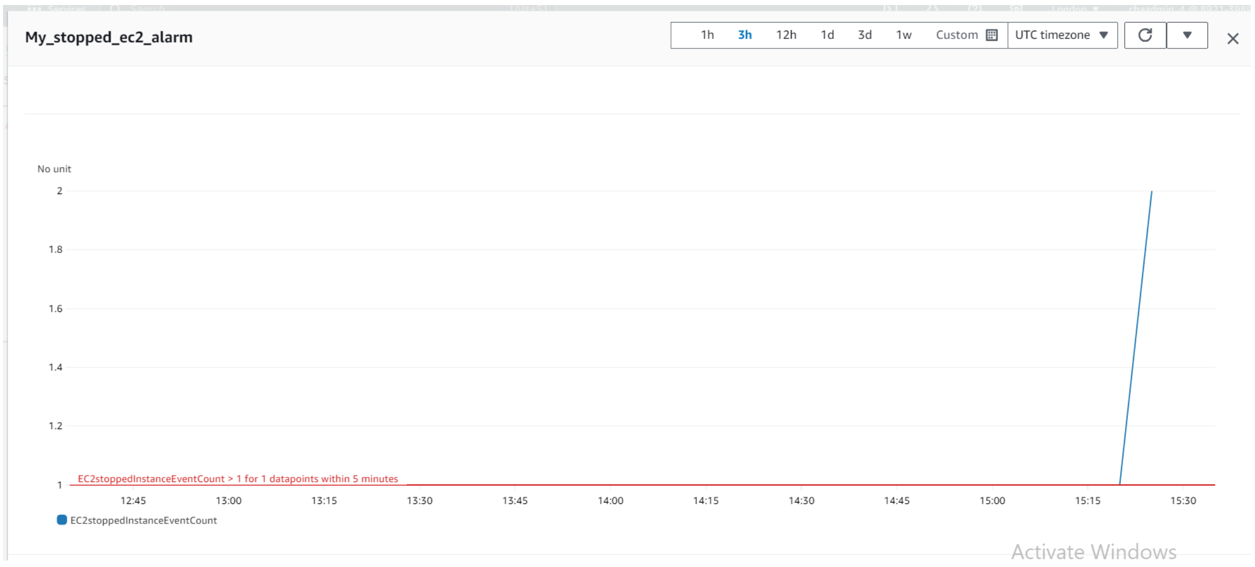
## 9. Creating a CloudWatch Dashboard

To visualize and monitor the state of the EC2 instance in real time, we created a **CloudWatch Dashboard**:

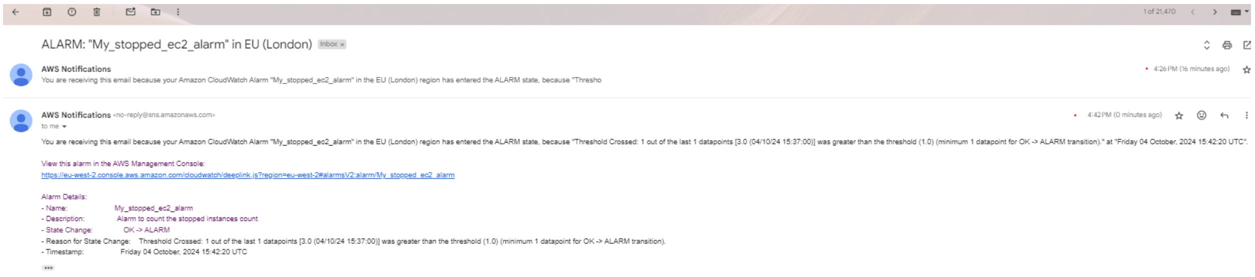
- We selected the alarm and clicked on the dropdown arrow next to **Action**.
- From the dropdown, we selected **Add to dashboard**.
- We created a new dashboard, entered a name, and selected **line graph** as the widget type to display the metric.
- The EC2 stop events were then visualized in the dashboard, allowing the team to see how frequently the instance was stopped and track the alarm's status.

## Dashboard After Stopping the Instance Again

After stopping and starting the EC2 instance multiple times, we received another email notification from SNS. The **CloudWatch Dashboard** reflected the repeated stop events, showing the updates in the **Alarm widget** and highlighting any significant deviations detected by the anomaly detection feature.



### Stopped and started the ec2instance again and here's the email



ALARM: "My_stopped_ec2_alarm" in EU (London) Inbox ×

1 of 21,470

**AWS Notifications**
You are receiving this email because your Amazon CloudWatch Alarm "My_stopped_ec2_alarm" in the EU (London) region has entered the ALARM state, because "Thresho

4:26 PM (16 minutes ago)

**AWS Notifications** <no-reply@sns.amazonaws.com>
to me

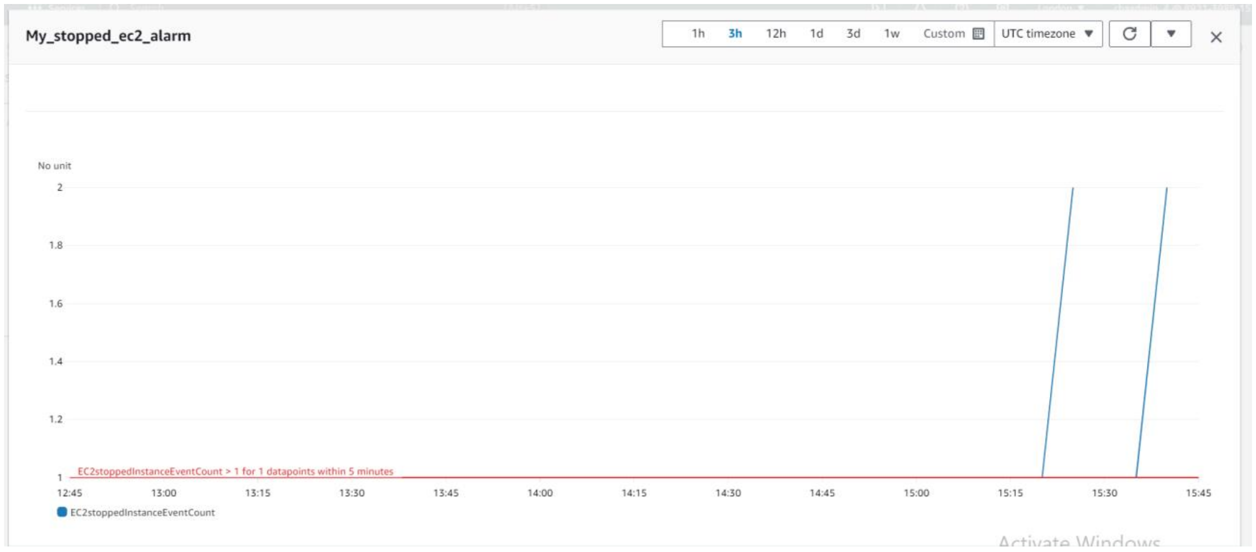
4:42 PM (0 minutes ago)

You are receiving this email because your Amazon CloudWatch Alarm "My_stopped_ec2_alarm" in the EU (London) region has entered the ALARM state, because "Threshold Crossed: 1 out of the last 1 datapoints [3.0 (04/10/24 15:37:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition)." at "Friday 04 October, 2024 15:42:20 UTC".

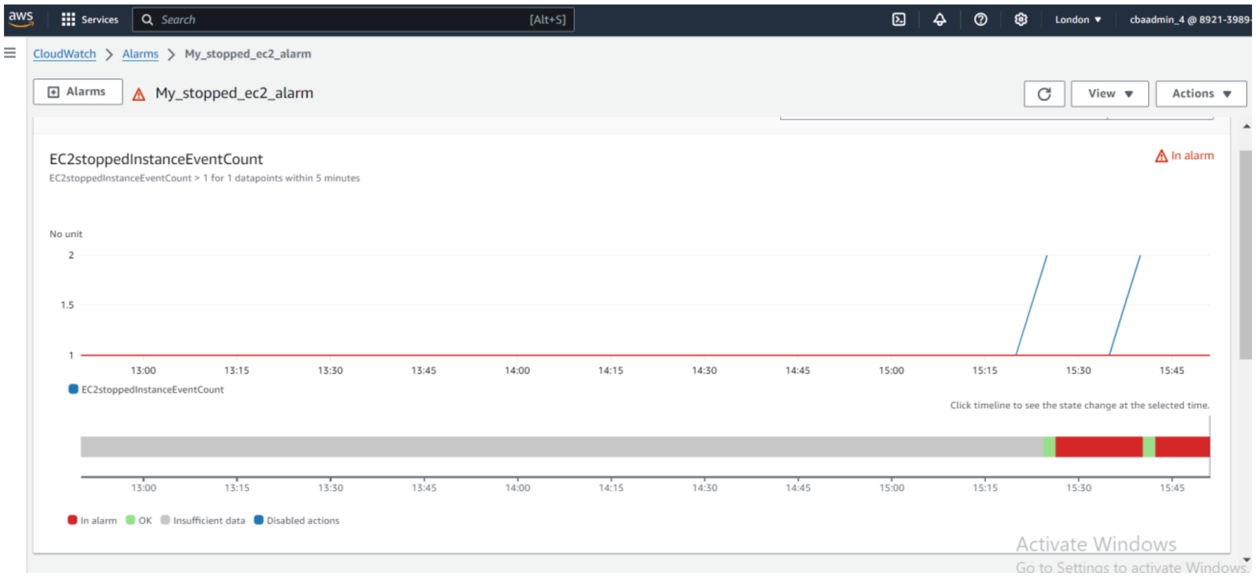
View this alarm in the AWS Management Console:
https://eu-west-2.console.aws.amazon.com/cloudwatch/deeplink.js?region=eu-west-2#alarmsV2:alarm/My_stopped_ec2_alarm

```
Alarm Details:
- Name:                     My_stopped_ec2_alarm
- Description:              Alarm to count the stopped instances count
- State Change:             OK -> ALARM
- Reason for State Change:  Threshold Crossed: 1 out of the last 1 datapoints [3.0 (04/10/24 15:37:00)] was greater than the threshold (1.0) (minimum 1 datapoint for OK -> ALARM transition).
- Timestamp:                Friday 04 October, 2024 15:42:20 UTC
```

**The dashboard after the 2nd operation**

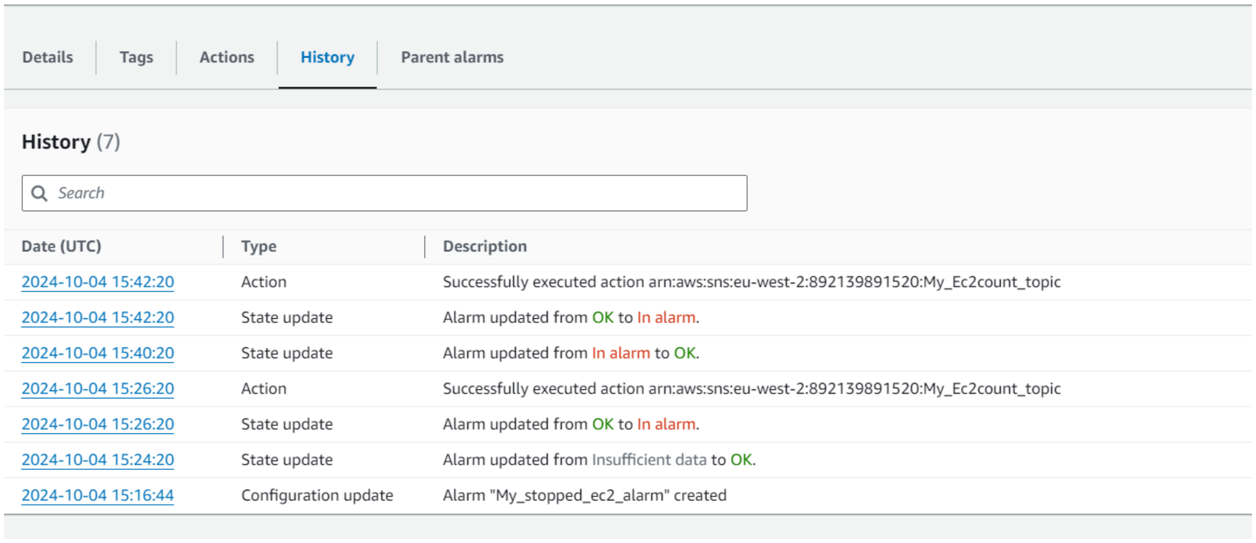


**Alarm widget added to better explain movement in dashboard**



## 10. History of Actions

The CloudWatch Dashboard displayed a **history of actions** related to the alarm and metric tracking, allowing administrators to easily review when the instance was stopped and how many times the alarm was triggered.

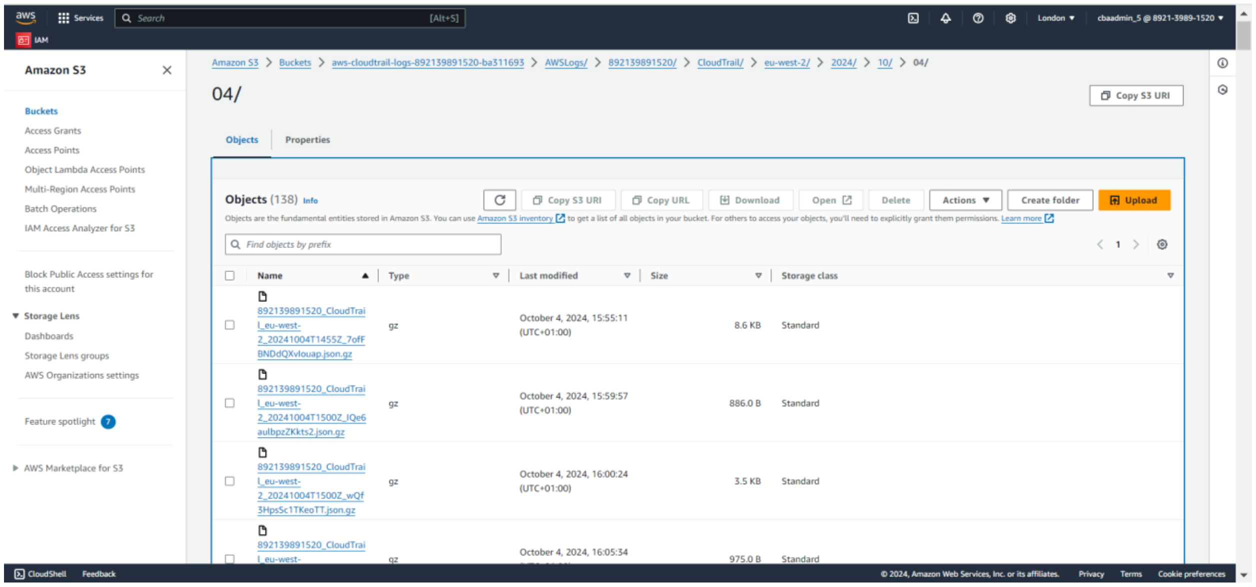


History (7)

| Date (UTC) | Type | Description |
|---|---|---|
| 2024-10-04 15:42:20 | Action | Successfully executed action arn:aws:sns:eu-west-2:892139891520:My_Ec2count_topic |
| 2024-10-04 15:42:20 | State update | Alarm updated from OK to In alarm. |
| 2024-10-04 15:40:20 | State update | Alarm updated from In alarm to OK. |
| 2024-10-04 15:26:20 | Action | Successfully executed action arn:aws:sns:eu-west-2:892139891520:My_Ec2count_topic |
| 2024-10-04 15:26:20 | State update | Alarm updated from OK to In alarm. |
| 2024-10-04 15:24:20 | State update | Alarm updated from Insufficient data to OK. |
| 2024-10-04 15:16:44 | Configuration update | Alarm "My_stopped_ec2_alarm" created |

## S3 Bucket for storage of Logs

Event logs generated are recorded and delivered to S3 bucket within the assigned threshold of 5 minutes



Amazon S3 > Buckets > aws-cloudtrail-logs-892139891520-ba311693 > AWSLogs/ > 892139891520/ > CloudTrail/ > eu-west-2/ > 2024/ > 10/ > 04/

04/

Objects (138) Info

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

| Name | Type | Last modified | Size | Storage class |
|---|---|---|---|---|
| 892139891520_CloudTrail_eu-west-2_20241004T1455Z_7ofFBNDdQXvlouap.json.gz | gz | October 4, 2024, 15:55:11 (UTC+01:00) | 8.6 KB | Standard |
| 892139891520_CloudTrail_eu-west-2_20241004T1500Z_IQe6aulbpzZKkts2.json.gz | gz | October 4, 2024, 15:59:57 (UTC+01:00) | 886.0 B | Standard |
| 892139891520_CloudTrail_eu-west-2_20241004T1500Z_wQf3HpsSc1TKeoTT.json.gz | gz | October 4, 2024, 16:00:24 (UTC+01:00) | 3.5 KB | Standard |
| 892139891520_CloudTrail_eu-west- | gz | October 4, 2024, 16:05:34 | 975.0 B | Standard |

## Features Utilized

Throughout this project, we used the following CloudWatch features:

1. **CloudWatch Log Insights**: For querying and analyzing CloudTrail logs.
2. **CloudWatch Metric Filters**: For converting log data into metrics that can be monitored.
3. **CloudWatch Metric Alarms**: To alert the team when the EC2 instance was stopped multiple times.
4. **CloudWatch Metric Anomaly Detection**: To detect unexpected patterns in EC2 stop events.

5. **CloudWatch Dashboards**: To visualize the EC2 instance's activity and monitor alarm status in real time.

### Features Not Used

- **CloudWatch Log Subscription Filters**: This feature was not used because we did not need to stream CloudTrail log events to a different service.

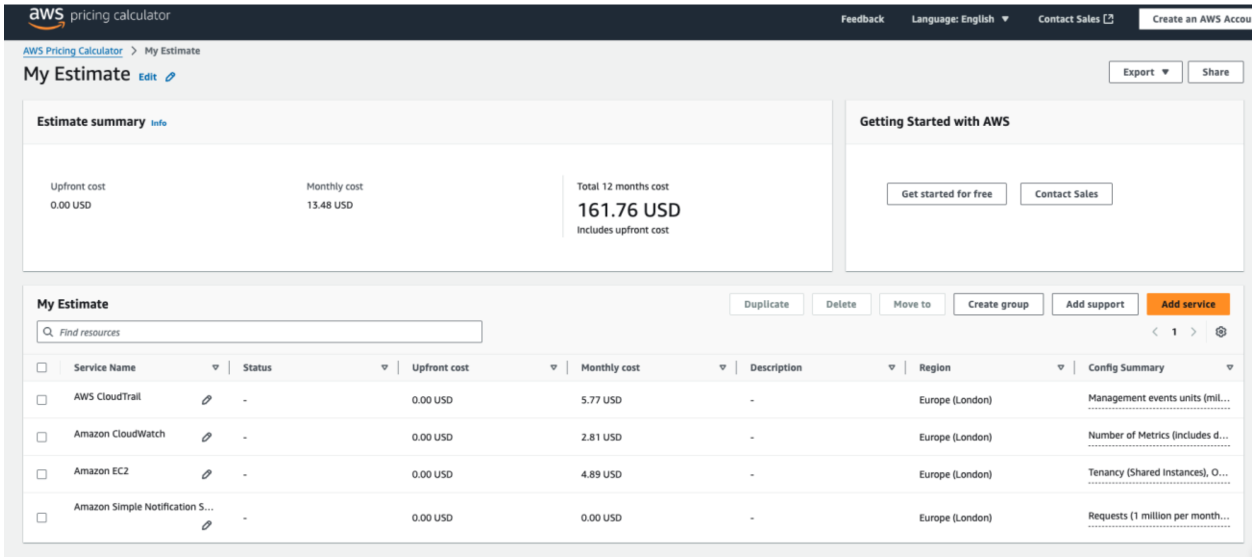
## Security Considerations

Security is a key aspect of this monitoring setup. Here are some important security considerations:

- **IAM Permissions**:
    - Enforce the principle of least privilege by ensuring that only authorized users have access to the CloudWatch Logs, CloudTrail, and SNS resources.
    - Limit permissions for starting, stopping, and terminating EC2 instances to trusted administrators only.
- **Encryption**:
    - Enable encryption at rest for all CloudTrail logs using AWS KMS to protect sensitive data.
    - Ensure encryption in transit is enabled for communication between AWS services and for notifications sent via SNS.
- **Audit Logging**:
    - CloudTrail’s detailed audit logs provide an invaluable resource for tracking changes and ensuring accountability.
    - Retain logs for a suitable period to meet compliance requirements and for security investigations.
- **Log Retention and Rotation**:
    - Implement appropriate log retention policies to balance storage costs with the need for keeping historical data for auditing.

## Cost

Free Credits



My Estimate

Estimate summary

| Upfront cost | Monthly cost | Total 12 months cost |
|---|---|---|
| 0.00 USD | 13.48 USD | 161.76 USD Includes upfront cost |

Getting Started with AWS

My Estimate

| Service Name | Status | Upfront cost | Monthly cost | Description | Region | Config Summary |
|---|---|---|---|---|---|---|
| AWS CloudTrail | - | 0.00 USD | 5.77 USD | - | Europe (London) | Management events units (mil... |
| Amazon CloudWatch | - | 0.00 USD | 2.81 USD | - | Europe (London) | Number of Metrics (includes d... |
| Amazon EC2 | - | 0.00 USD | 4.89 USD | - | Europe (London) | Tenancy (Shared Instances), O... |
| Amazon Simple Notification S... | - | 0.00 USD | 0.00 USD | - | Europe (London) | Requests (1 million per month... |

## Future Enhancements and Recommendations

1. **Automated Remediation:** Integrating AWS Lambda functions with CloudWatch Alarms would allow for automated remediation. For example, if an alarm is triggered because the EC2 instance has stopped multiple times, a Lambda function could automatically restart the instance or alert a secondary system.
2. **Enhanced Security Monitoring:** Integrate AWS GuardDuty to extend security monitoring beyond instance stops. GuardDuty provides threat detection and continuous monitoring for malicious activity, such as unauthorized API calls or compromised EC2 instances.
3. **Cost Optimization:** Implement strategies to reduce operational costs, such as reviewing log retention policies, compressing logs, and filtering unnecessary events before they are sent to CloudWatch Logs. Customizing your metric collection and alerting for only critical events can reduce expenses while maintaining efficient monitoring.