



GuardDuty Root Account Misuse Alerting System

A lightweight security automation solution that detects root account activity using Amazon GuardDuty and instantly alerts administrators via Amazon SNS using a Lambda-based notification workflow.

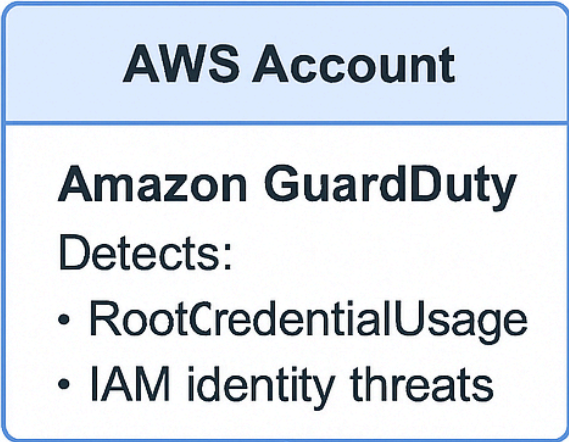
Services Used

- Amazon GuardDuty
- Amazon EventBridge
- AWS Lambda (Python)
- Amazon SNS

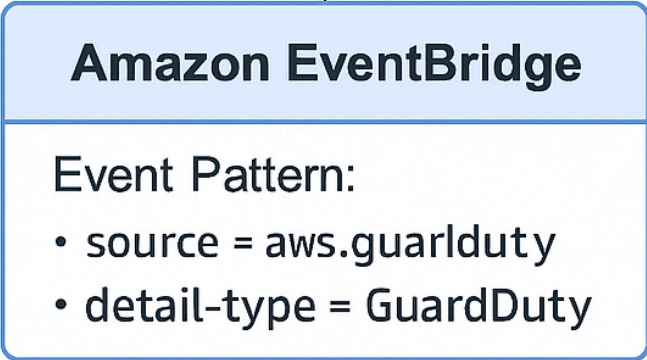
Project Highlights

- Implemented real-time detection of AWS root account usage using Amazon GuardDuty.
- Designed an EventBridge → Lambda → SNS automation pipeline to immediately alert administrators.

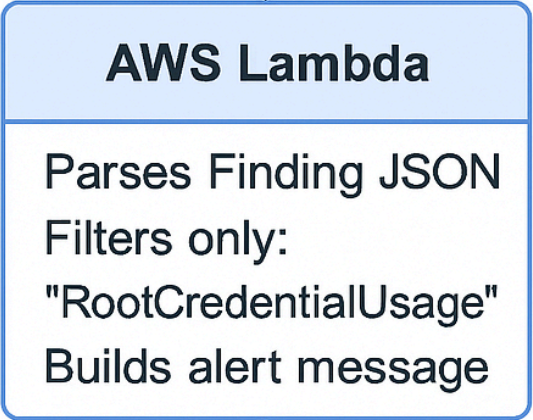
- **Built a serverless, cost-effective, fully automated security workflow with no infrastructure to maintain.**
- **Parsed and enriched GuardDuty findings to extract details like:**
 - **Finding type**
 - **Source IP**
 - **Region**
 - **Event timestamp**
- **Delivered alerts via SMS and email using Amazon SNS for rapid incident response.**
- **Built event pattern filters to match only high-risk identity misuse findings.**
- **Ensured compliance with AWS Security Best Practices by notifying stakeholders about dangerous root usage.**
- **Validated system behavior using sample GuardDuty findings and EventBridge test events.**



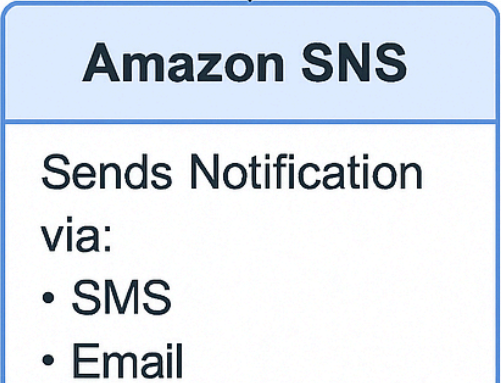
(Finding Event Published)



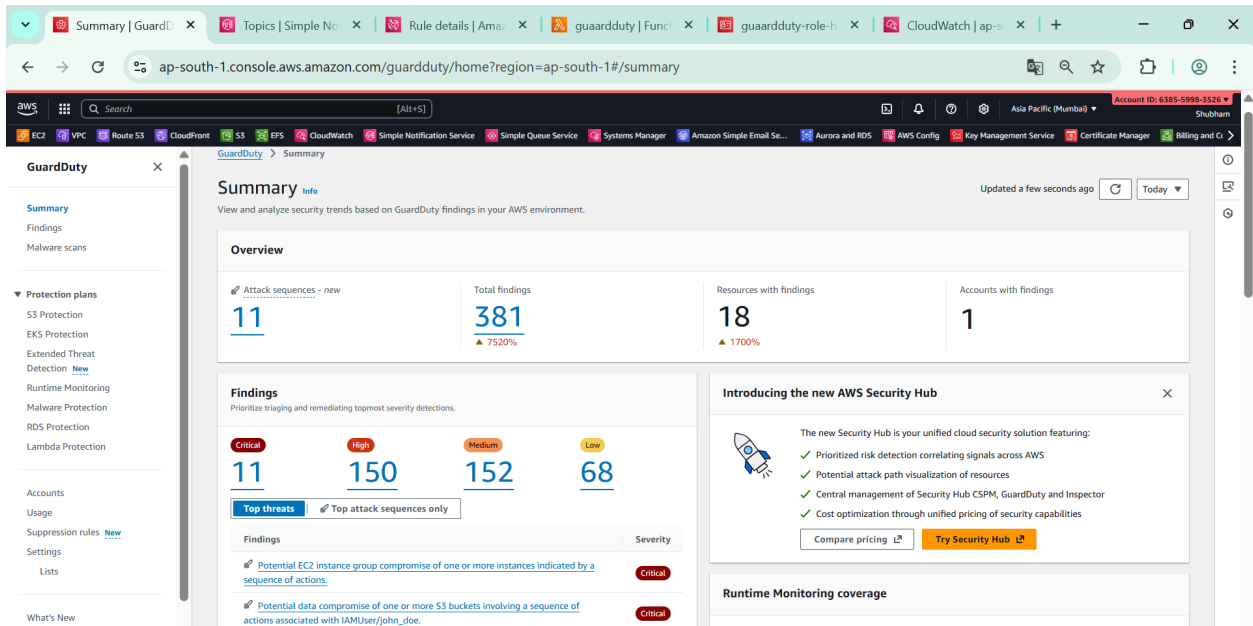
(Triggers Lambda)



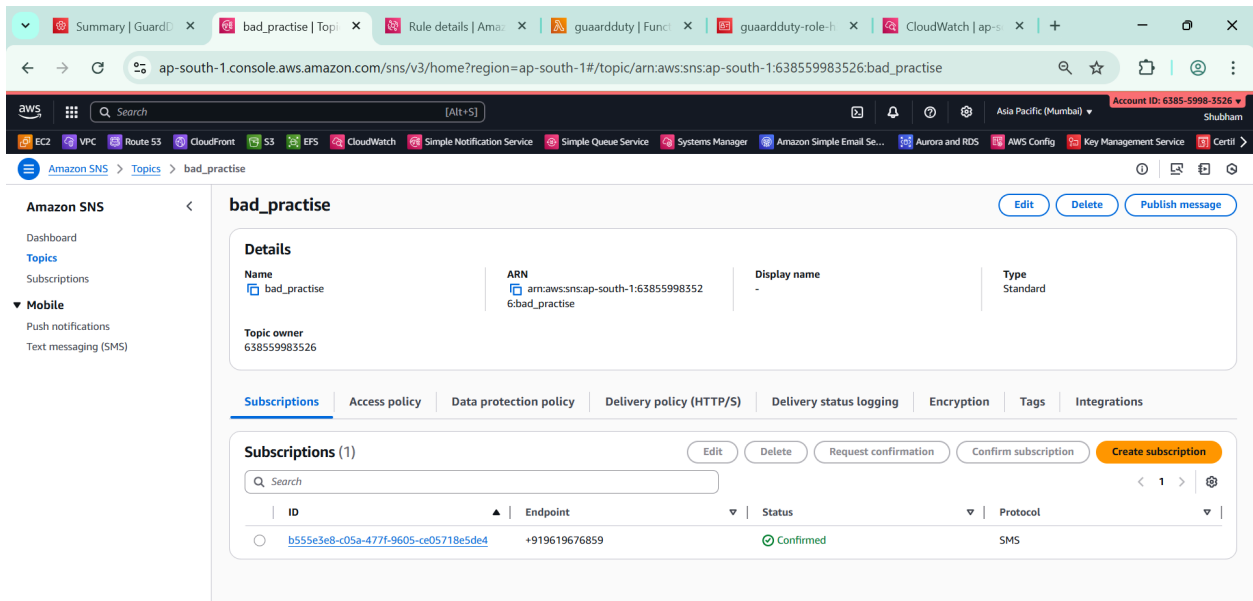
(sns.publish)



Step 1) We will enable Guard duty for our AWS account.



Step 2) We will create SNS topic and add subscriptions to that topic



Step 3) Now we will create Lambda function to automate trigger on any findings based on root account.

ap-south-1.console.aws.amazon.com/lambda/home?region=ap-south-1#/create/function?intent=authorFromScratch

Successfully deleted function: guardduty

Create function Info

Choose one of the following options to create your function.

- Author from scratch**
Start with a simple Hello World example.
- Use a blueprint**
Build a Lambda application from sample code and configuration presets for common use cases.
- Container image**
Select a container image to deploy for your function.

Basic information

Function name
Enter a name that describes the purpose of your function.

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (_).

Runtime Info
Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Architecture Info
Choose the instruction set architecture you want for your function code.
 arm64
 x86_64

Permissions Info

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Change default execution role

Execution role
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- Create a new role with basic Lambda permissions
- Use an existing role
- Create a new role from AWS policy templates

Existing role
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

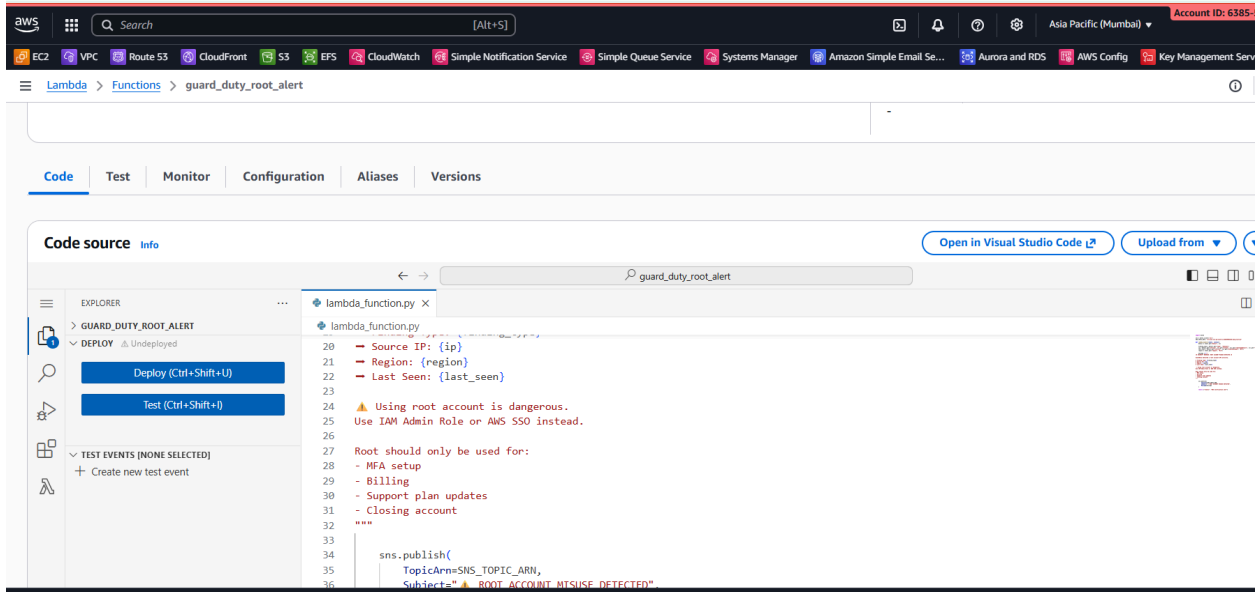
[View the guardduty-role-h33fwoc0 role](#) on the IAM console.

Additional configurations

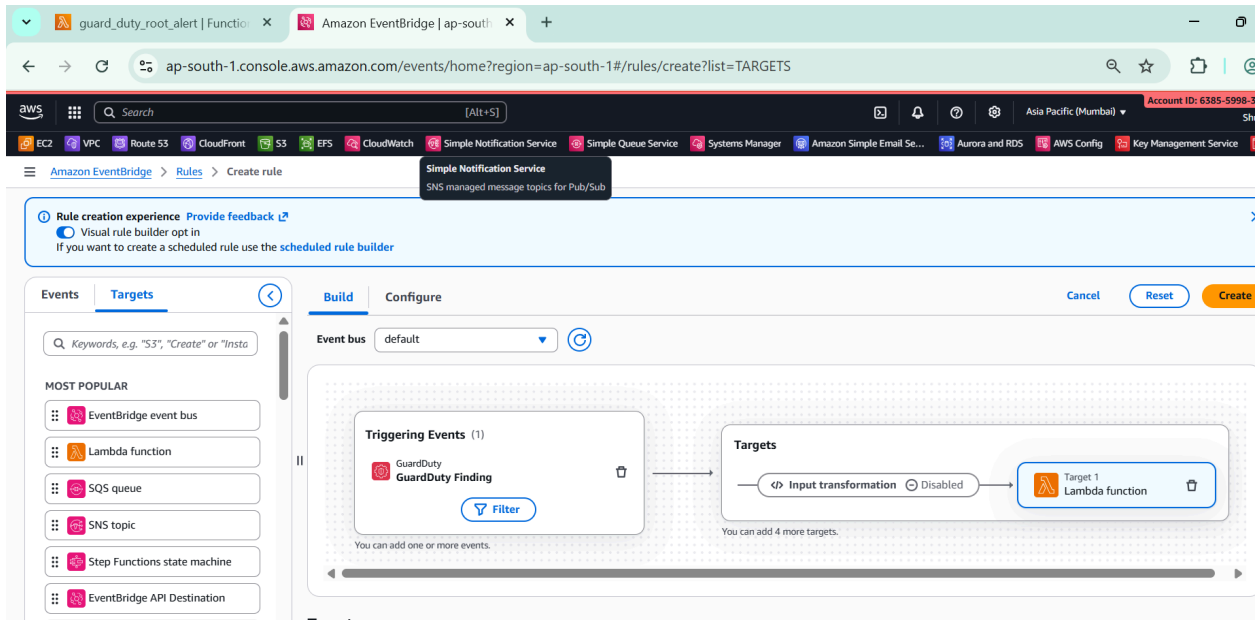
Use additional configurations to set up networking, security, and governance for your function. These settings help secure and customize your Lambda function deployment.

[Cancel](#) [Create function](#)

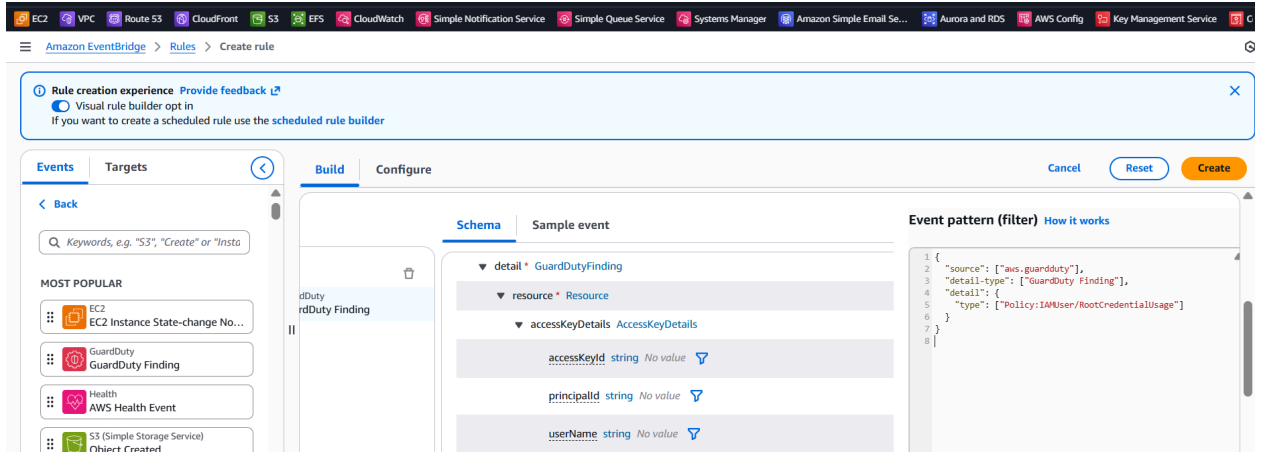
Step 4) We will place the code python.



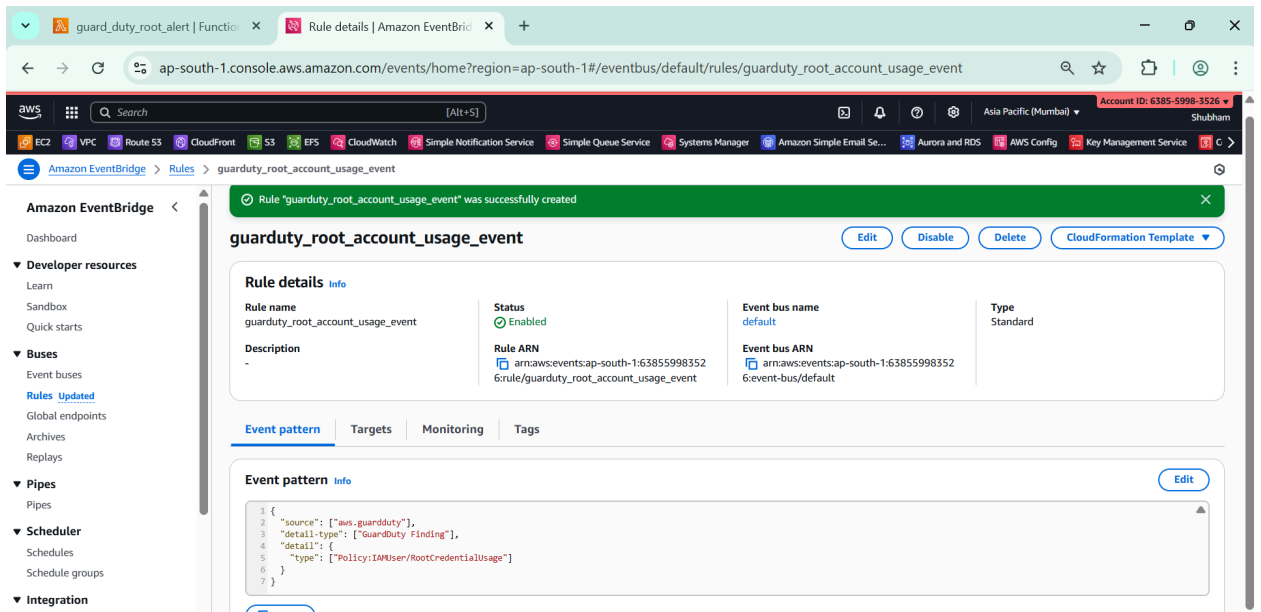
Step 5) We will now create Aws eventbridge rule and add event trigger and target as lambda function.



Step 6) We will enter event rule filter now.



Step 7) We can see the rule is created successfully.



Step 8) We will add this event as trigger for Lambda function as show below

guard_duty_root_alert Throttled

✔ The trigger guardduty_root_account_usage_event was successfully added to function guard_duty_root_alert.

▼ **Function overview** [Info](#) Export to Infrastructure

Diagram | Template

guard_duty_root_alert

Layers (0)

EventBridge (CloudWatch Events)

+ Add trigger

+ Add destination

Description

Last modified: 3 minutes ago

Function ARN: arn:aws:lambda:ap-south-1:63t_alert

Function URL: [Info](#)

Code | Test | Monitor | **Configuration** | Aliases | Versions

Step 9) Now we will run generate sample findings in aws guardduty to simulate fake threats.

Settings | GuardDuty | ap-south-1 | x | guard_duty_root_alert | Function | x | Rule details | Amazon EventBrid | x | +

ap-south-1.console.aws.amazon.com/guardduty/home?region=ap-south-1#/settings

Malware scans

Protection plans

- S3 Protection
- EKS Protection
- Extended Threat Detection **New**
- Runtime Monitoring
- Malware Protection
- RDS Protection
- Lambda Protection

Accounts

- Usage
- Suppression rules **New**
- Settings**
- Lists

What's New

✔ Successfully generated sample findings

GuardDuty uses a service role to monitor your data sources on your behalf. Open [AWS IAM console](#) to manage this role.

[View service role permissions](#)

Delegated Administrator [Info](#)

Delegate permission to manage GuardDuty for this organization.

Account ID	Organization ID
638559983526	o-cx73b32up1




[Remove](#)

Findings export options [Info](#)

Findings are automatically sent to EventBridge. You can also export findings to an S3 bucket. New findings are exported within 5 minutes. You can modify the frequency for updated findings below.


Frequency	S3 bucket
Update EventBridge and S3 every 15 minutes	-


Step 10) We can see we received notification once threat detected.


←  59039465  


 SECURITY ALERT: ROOT
ACCOUNT ACTIVITY
DETECTED 

GuardDuty has reported
activity.

 Finding Type:
Impact:Kubernetes/
SuccessfulAnonymousAcces
s

 Region: ap-south-1

 Last Seen:
2025-12-05T06:38:40.000Z

 If this is root usage, avoid
using root for daily tasks.
Use IAM roles or AWS SSO

 Text mess...   

This project successfully demonstrates an automated security response workflow using AWS GuardDuty, EventBridge, Lambda, and SNS. By detecting root account misuse in real time and instantly alerting administrators through SMS/Email, it significantly strengthens the security posture of the AWS account.

This automation ensures faster incident response, reduces human error, and enforces best practices for identity and access management across cloud environments.