Enhanced AWS Security Monitoring Dashboard: Implementation Guide

Objective

To build a Free-Tier-compatible, real-time AWS security monitoring system using **CloudTrail, CloudWatch, SNS, and Lambda**, with a focus on **RootAccountUsage**, unauthorized API activity, and sensitive infrastructure changes.

Updated Metrics Monitored

Metric Name	Description
`RootAccountUsageCount`	Counts usage of the highly privileged root account (updated from IAM usage)
`FailedConsoleLoginCount`	*(Omitted – policy could not be implemented)*
`UnauthorizedAPICallCount`	Tracks attempts to use AWS services without permission
`SecurityGroupChangeCount`	Monitors modifications to EC2 security groups
`IAMPolicyChangeCount`	Tracks IAM policy attachments, detachments, creations, and deletions

---

Project Architecture & Components

1. **S3 + CloudTrail** – Collect logs
2. **CloudWatch Logs** – Process log events
3. **CloudWatch Metric Filters** – Generate custom metrics
4. **SNS Topics** – Send email alerts for critical/warning events

5. **Lambda Function** – Enhances alert messages
6. **CloudWatch Dashboard** – Visualize security metrics

[User Activity] → [CloudTrail Logs] → [S3] + [CloudWatch Logs]

↓

[Metric Filters + Alarms on Key Events]

↓

[SNS Topics: Critical/Warning]

↓

[Lambda: Alert Parsing + Forwarding]

↓

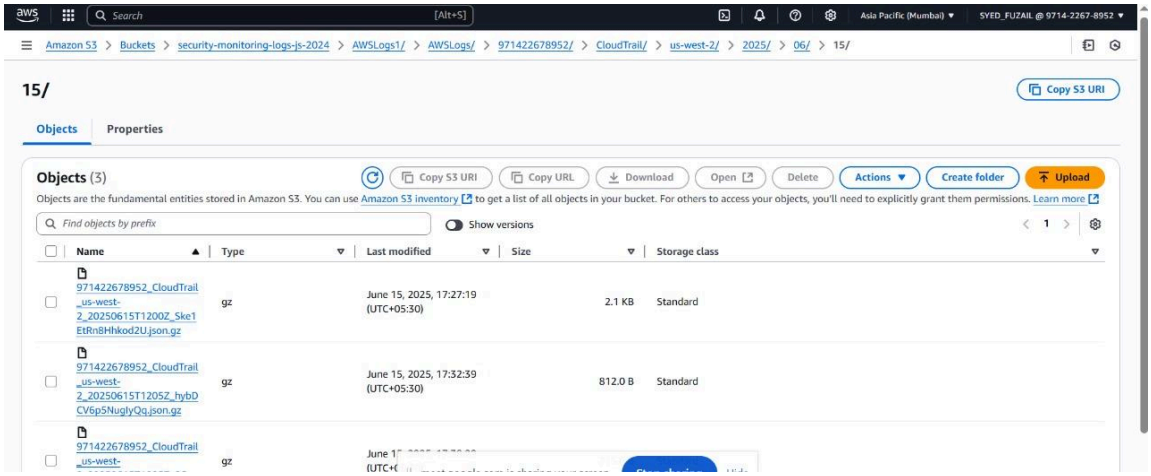
[Email Notifications + Dashboard]

Phase 1: CloudTrail Logging Setup

1.1 Create S3 Bucket for Logs

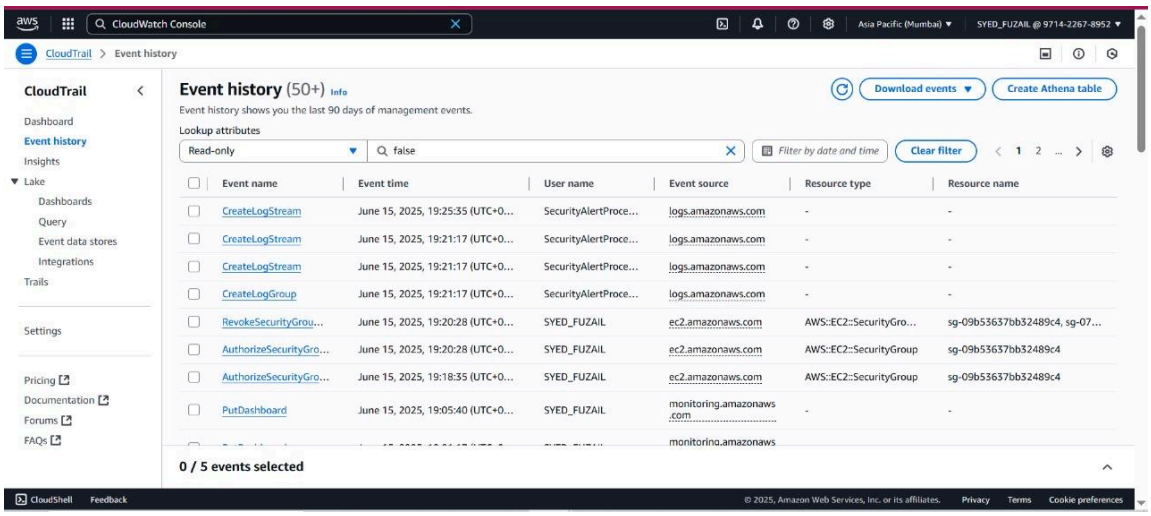
- Name: `security-monitoring-logs-<your-tag>`
- Enable versioning and SSE-S3 encryption

- Block public access



1.2 Create CloudTrail Trail

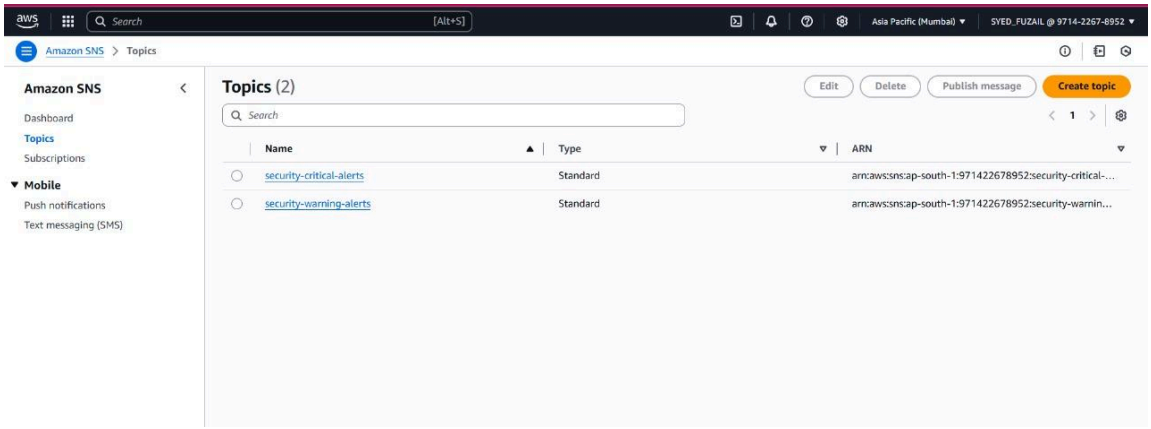
- Name: `security-monitoring-trail`
- Apply trail to **all regions**
- Enable log file validation
- Log destination: Use the S3 bucket from above
- Enable CloudWatch Logs integration



Phase 2: SNS for Alerting

Create Two Topics:

- `security-critical-alerts`
- `security-warning-alerts`



Subscribe Emails:

- Protocol: `Email`

Phase 3: CloudWatch Filters & Alarms

3.1 Metric Filters

Root Account Usage

json
CopyEdit

```
{ $.userIdentity.type = "Root" && $.eventType != "AwsServiceEvent" }
```

- Metric: `RootAccountUsageCount`

Unauthorized API Calls

json
CopyEdit

```
{ ($.errorCode = "*UnauthorizedOperation") || ($.errorCode =
"AccessDenied*") }
```

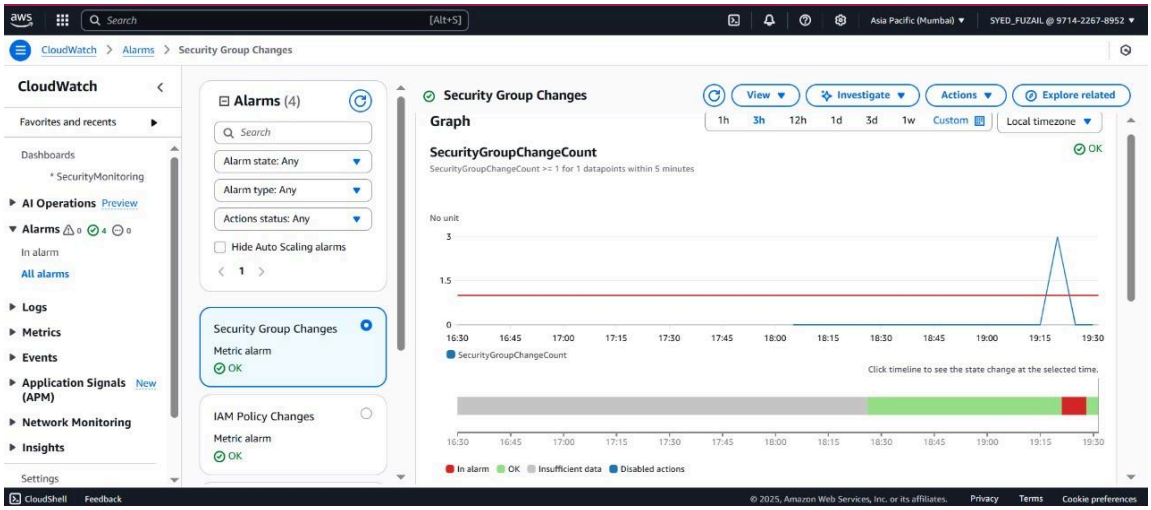
- Metric: `UnauthorizedAPICallCount`

Security Group Changes

json
CopyEdit

```
{ ($.eventName = AuthorizeSecurityGroupIngress) ||
  ($.eventName = RevokeSecurityGroupIngress) ||
  ($.eventName = CreateSecurityGroup) ||
  ($.eventName = DeleteSecurityGroup) }
```

- Metric: `SecurityGroupChangeCount`

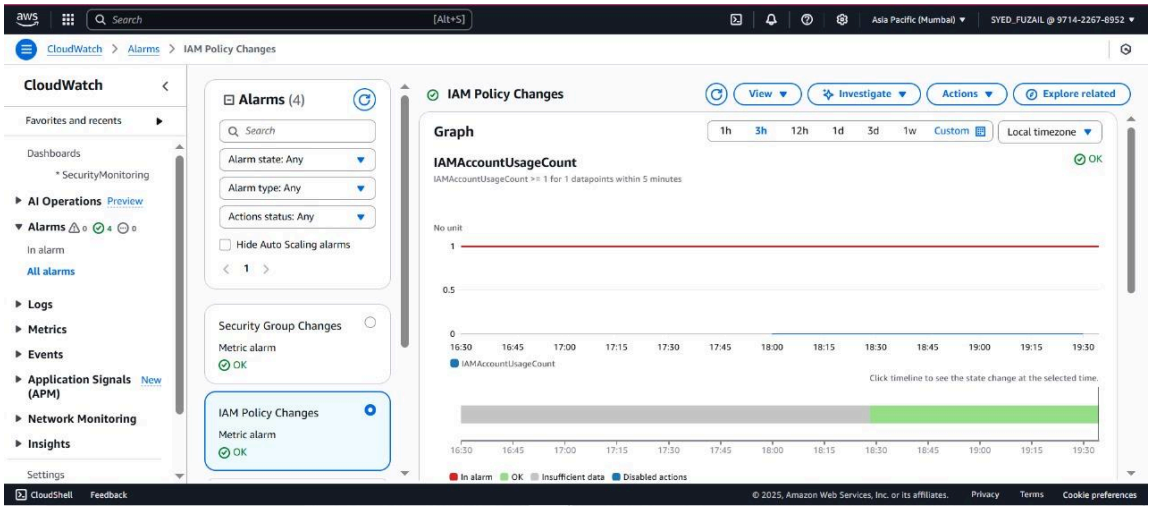


IAM Policy Changes

json
CopyEdit

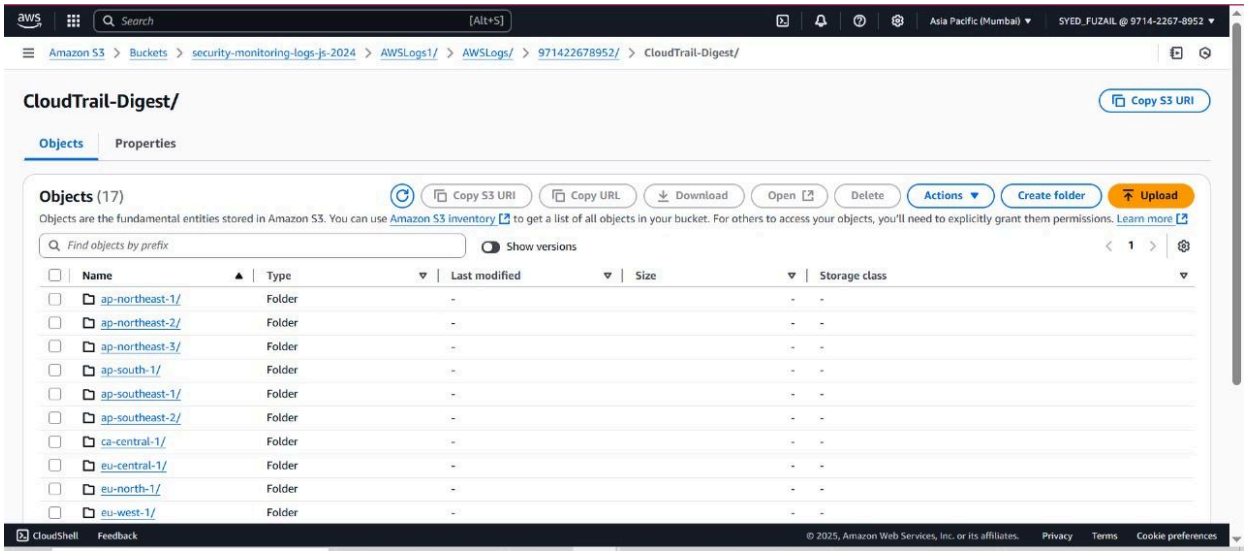
```
{ ($.eventName=PutUserPolicy) ||
  ($.eventName=AttachRolePolicy) ||
  ($.eventName=CreatePolicy) ||
  ($.eventName=DeletePolicy) }
```

- Metric: `IAMPolicyChangeCount`



3.2 CloudWatch Alarms

Alarm Name	Metric Name	Threshold	Alert Type
`Root Account Usage Alert`	RootAccountUsageCount	≥ 1	Critical (SNS)
(Omitted)	FailedConsoleLoginCount	N/A	(Policy issue)
`Unauthorized API Calls`	UnauthorizedAPICallCount	≥ 10	Warning (SNS)
`Security Group Changes`	SecurityGroupChangeCount	≥ 1	Warning (SNS)
`IAM Policy Changes`	IAMPolicyChangeCount	≥ 1	Critical (SNS)



Phase 4: Lambda-Based Alert Enrichment

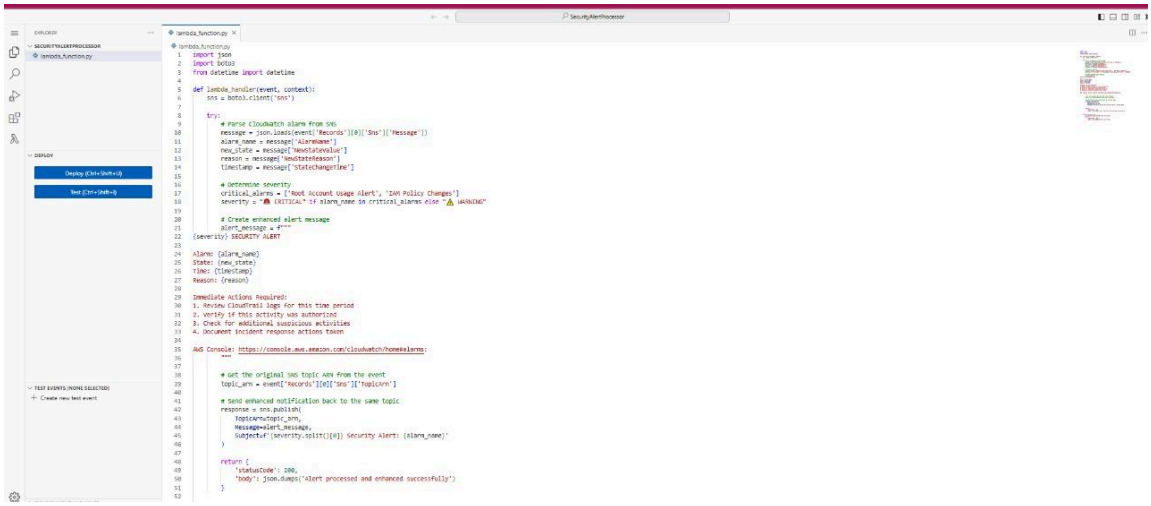
Lambda Function Name: `SecurityAlertProcessor`

- Triggered by both SNS topics
- Enhances raw alarm messages
- Categorizes into CRITICAL or ⚠WARNING

Language: Python 3.9
Role: `SecurityMonitoringLambdaRole`
Permissions:

- `sns:Publish`
- `cloudwatch:PutMetricData`
- `s3:GetObject`

- `logs:PutLogEvents`

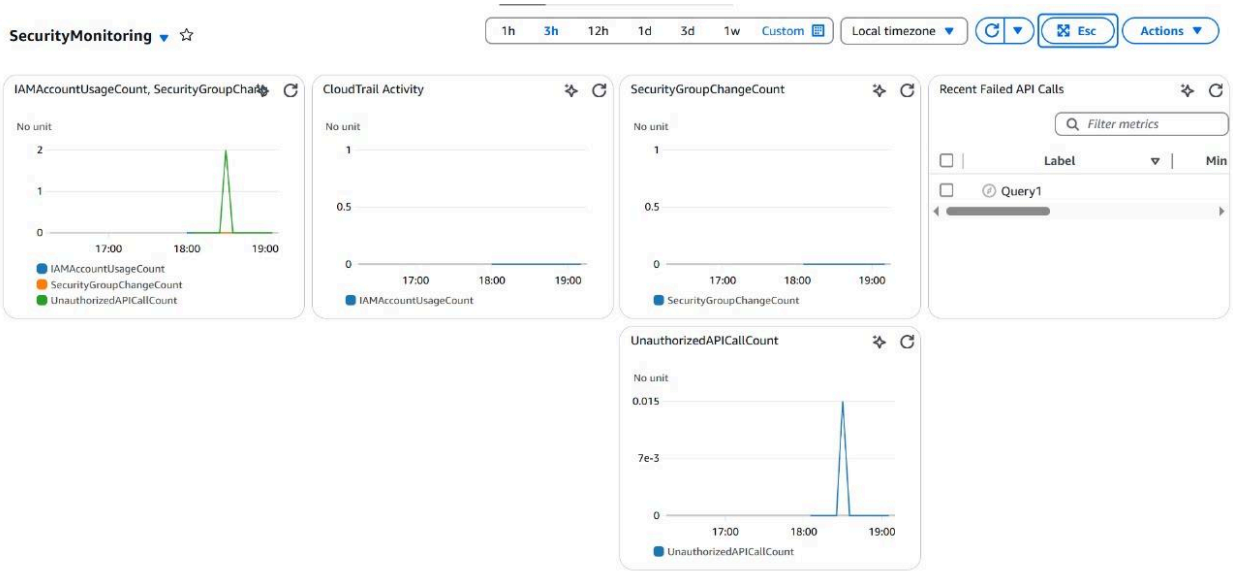


Phase 5: CloudWatch Dashboard

Dashboard: `SecurityMonitoring`

Widgets:

- **Timeline Line Graph** – All 4 active metrics
- **Single Value Panels** – e.g., `RootAccountUsageCount` in past 24h
- **CloudTrail Activity Graph**
- **Log Insights Query Table** for failed API calls



Phase 6: Testing & Verification

1. Trigger `RootAccountUsageCount` manually via console or CLI using root user.
2. Simulate a **Security Group Change** (e.g., open HTTP to all).
3. Trigger **Unauthorized API Call** using an underprivileged IAM user.
4. Verify:
    - Metrics incremented in dashboard
    - Alarms triggered
    - Email alerts received
    - Lambda logs show enriched notifications

Maintenance & Recommendations

Daily

- Check dashboard metrics
- Review triggered alarms

Weekly

- Adjust thresholds if needed
- Check for missed alerts or false positives

Monthly

- Analyze security trends
- Evaluate IAM roles and unused policies

Key Changes in This Version

Original Metric	Updated/Removed	Reason
`IAMAccountUsageCount`	`RootAccountUsageCount`	Focus on root activity (more critical)
`FailedConsoleLoginCount`	Removed	Policy setup issues, not implemented
Others	Retained as per original design	Core part of threat detection