Identity and Access Management (IAM) Implementation on AWS

Problem Statement

A healthcare technology organization managing sensitive patient data identified inconsistent identity and access controls across its AWS environment. Multiple teams had administrative privileges, IAM policies were not standardized, and access reviews were performed manually without central visibility. Due to regulatory requirements such as data privacy and healthcare compliance standards, the organization required a structured IAM governance framework to enforce least privilege and centralized access control. I designed and implemented this framework to strengthen access controls, enforce security standards, automate identity workflows, and improve operational visibility. The solution supports secure operations, regulatory compliance, and long-term scalability.

1. Objectives

The primary objectives of this implementation were to:

- Enforce least privilege access across all AWS accounts
- Require multi-factor authentication (MFA) for all users
- Automate IAM access key rotation
- Centralize logging and monitoring
- Improve audit readiness
- Reduce identity-related security risks
- Establish consistent identity governance

2. User and Access Management

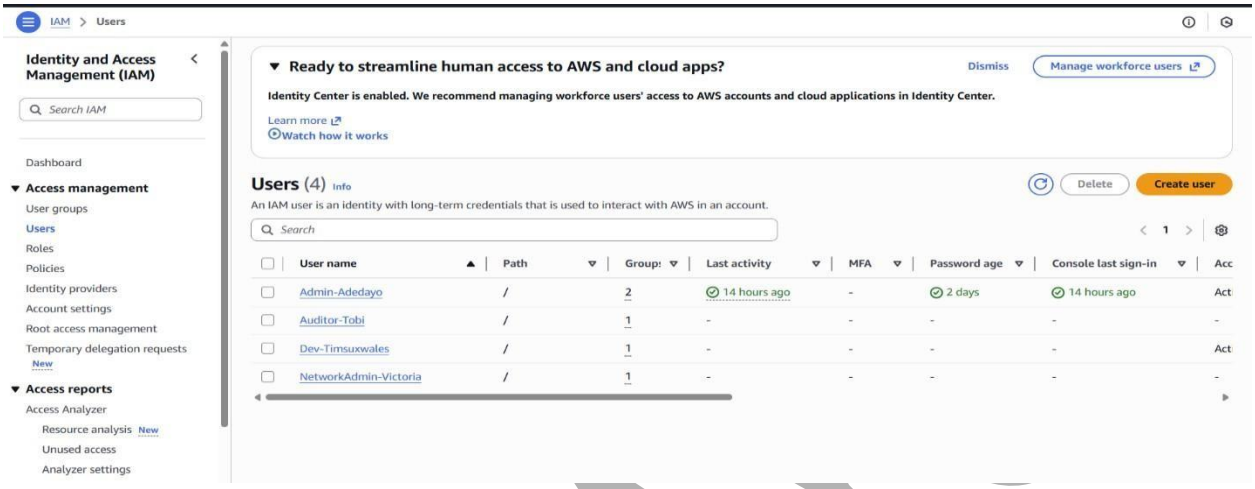
2.1 User Provisioning

I provisioned IAM users based on operational roles within the organization:

- Administrator
- Developer

- Auditor
- Network Administrator

Each user was assigned permissions aligned with job responsibilities and business requirements.



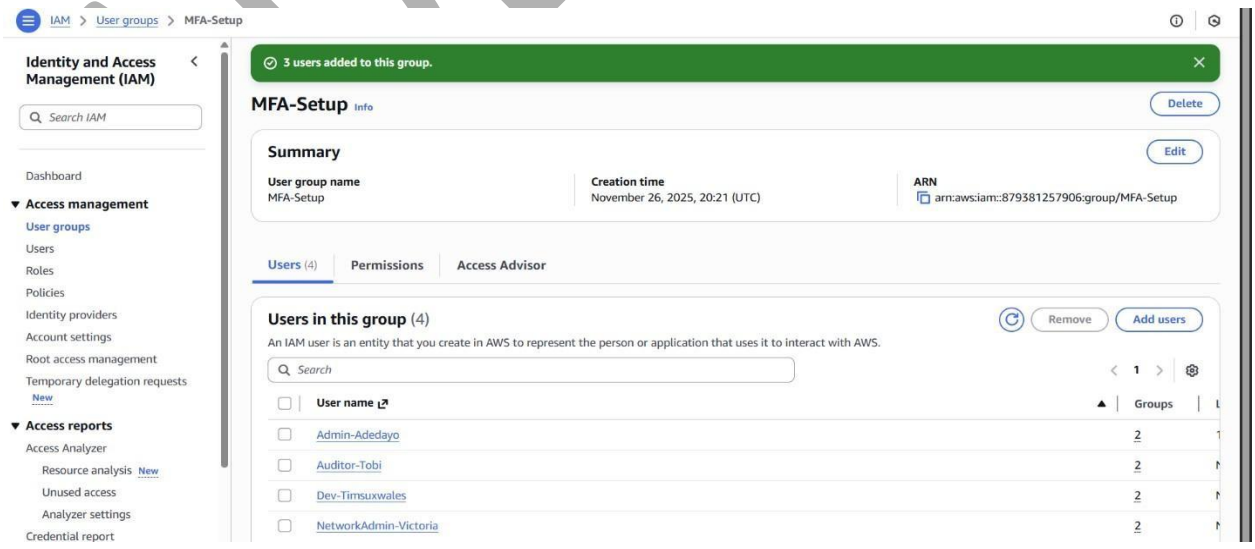
[Figure 1: IAM Users]

2.2 MFA Enforcement

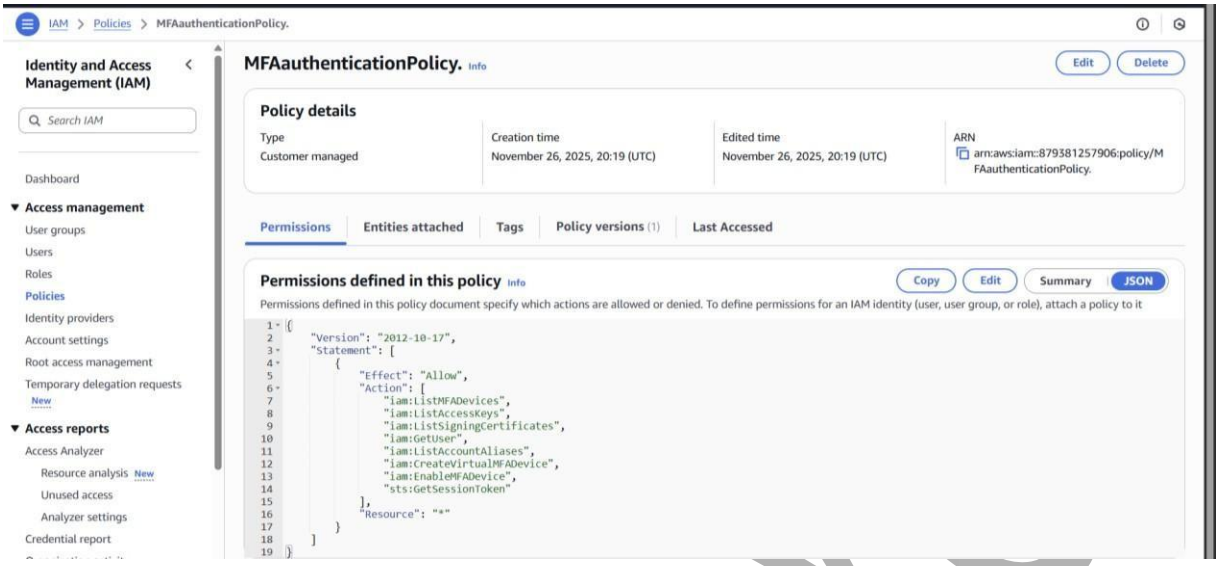
I implemented a mandatory MFA onboarding process.

All new users were placed in a restricted onboarding group and required to complete MFA enrollment before accessing any AWS resources. Until MFA was enabled, access remained limited.

This approach ensured consistent enforcement of strong authentication controls.

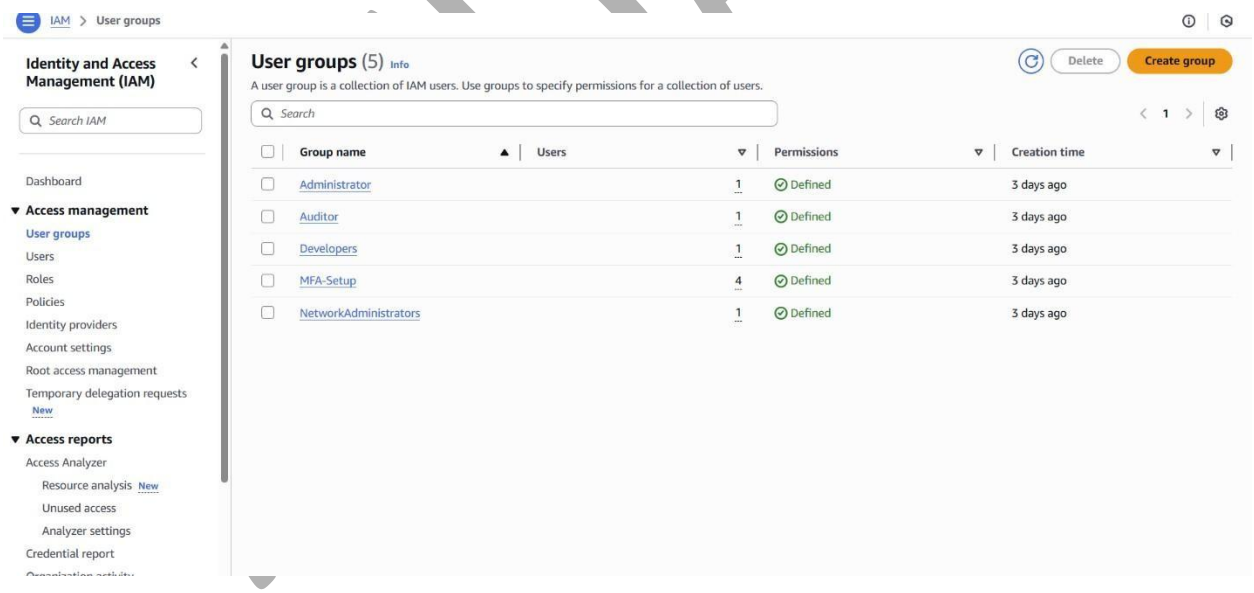


[Figure 2: MFA Group Configuration]



3.3 Role-Based Group Structure and Access Lifecycle

Permanent access groups were created for each role.



During initial setup and troubleshooting, temporary administrative permissions were granted to allow efficient configuration. Once the environment stabilized, I removed permanent administrative access and replaced it with controlled role assumption.

This reduced long-term exposure to privileged access.

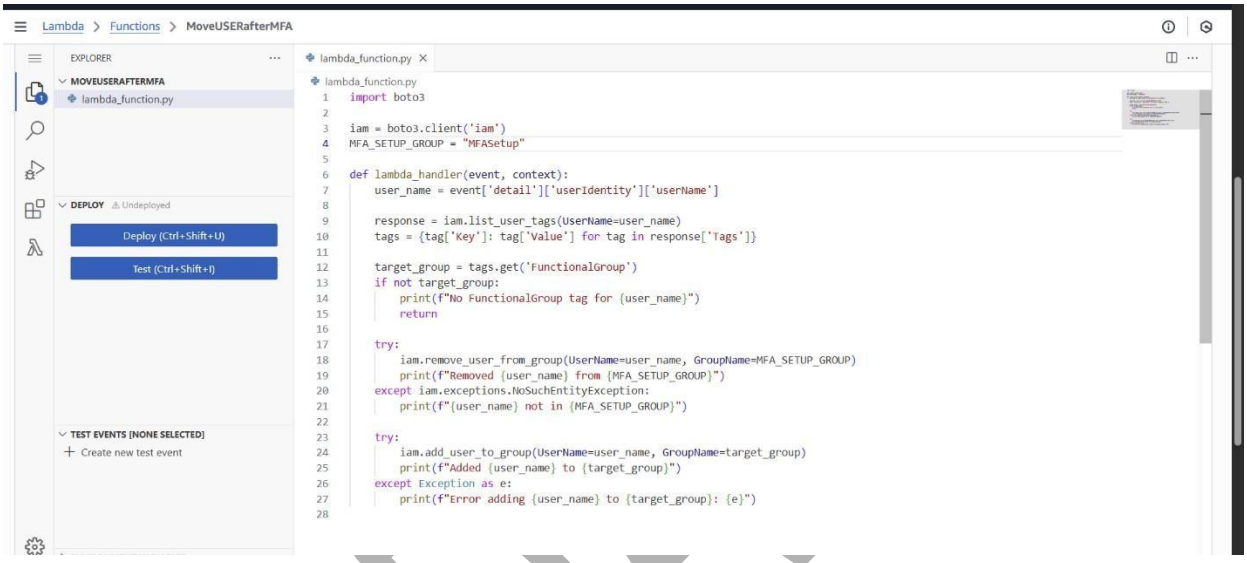
All access changes were documented to support audit and compliance reviews.

3. MFA Automation and User Lifecycle Management

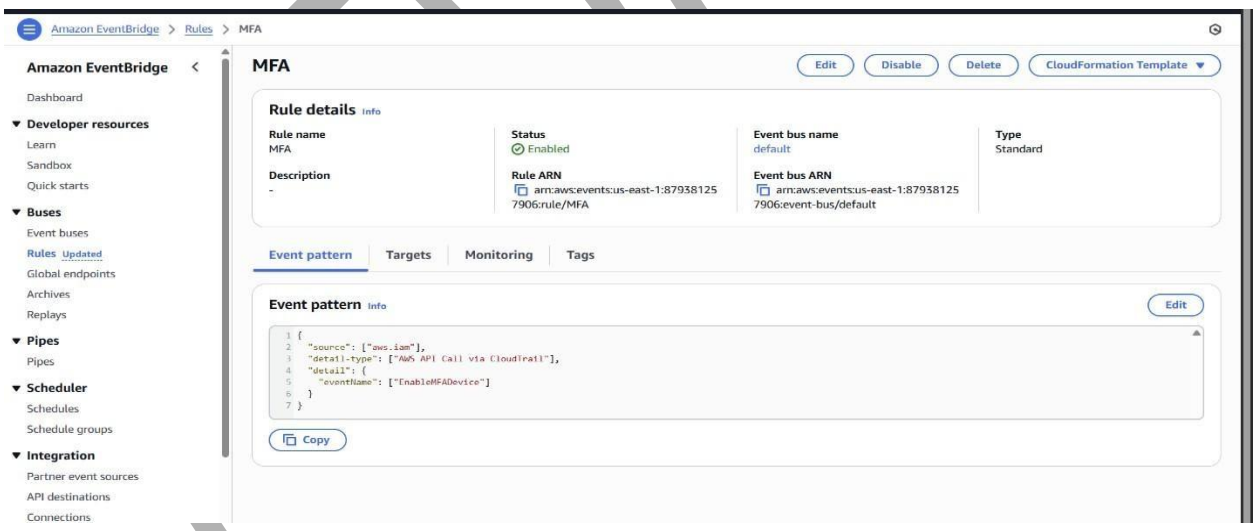
3.1 Automation Architecture

I designed and deployed an automated user lifecycle workflow using: •

AWS Lambda



- Amazon Event Bridge.



- IAM Roles and Policies
- User Tagging

A dedicated automation role and Lambda function managed user transitions between groups after MFA activation.

3.2 Tag-Based Access Assignment

I implemented standardized tagging for all IAM users.

Tags defined each user's operational role and department. The automation workflow relied on these tags to determine final group placement.

Tag compliance became a mandatory part of the user provisioning process.

4.3 Operational Challenges and Resolution

Several issues were identified and resolved during implementation.

Missing User Tags

Some users did not migrate after MFA activation due to missing metadata.

Resolution:
I enforced mandatory tagging standards and updated onboarding procedures.

Incorrect Group Reference

An incorrect group name in the Lambda function prevented proper removal from the onboarding group.

Resolution:
I corrected the group reference in the automation logic and revalidated the workflow.

Validation and Cleanup

After remediation, I re-tested user migrations using multiple roles and manually cleaned up legacy group assignments.

This ensured consistency across all accounts.

4. IAM Access Key Lifecycle Management

4.1 Credential Auditing

I regularly reviewed IAM Credential Reports to evaluate:

- Access key age
- MFA status

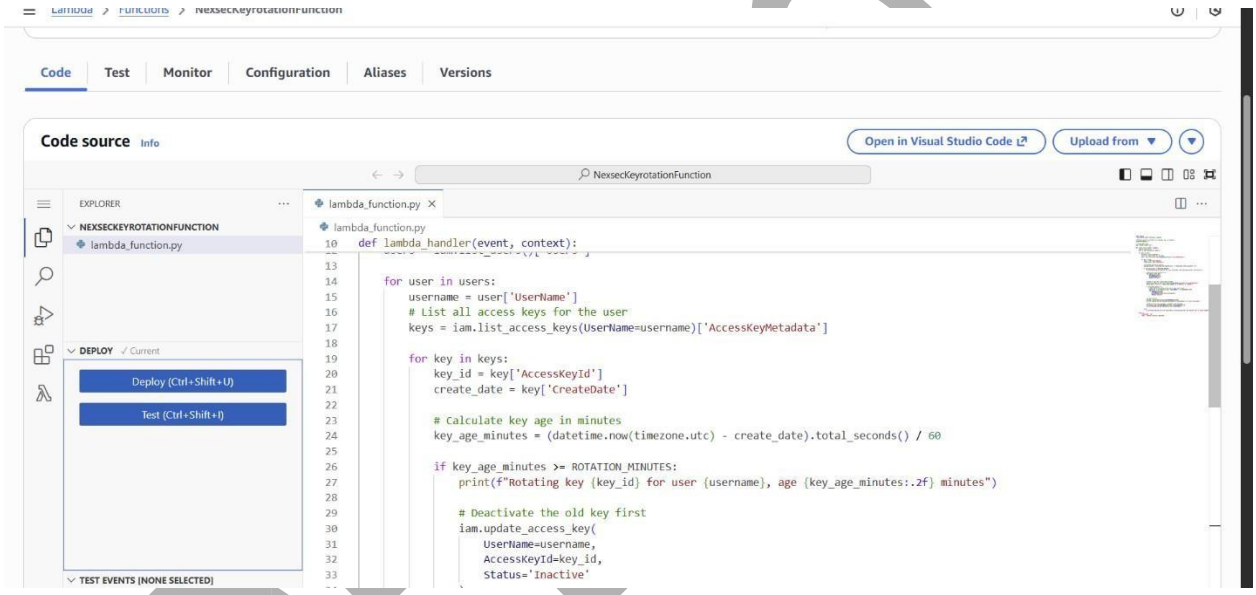
- Password compliance
- Credential usage

This supported proactive identification of weak or unused credentials.

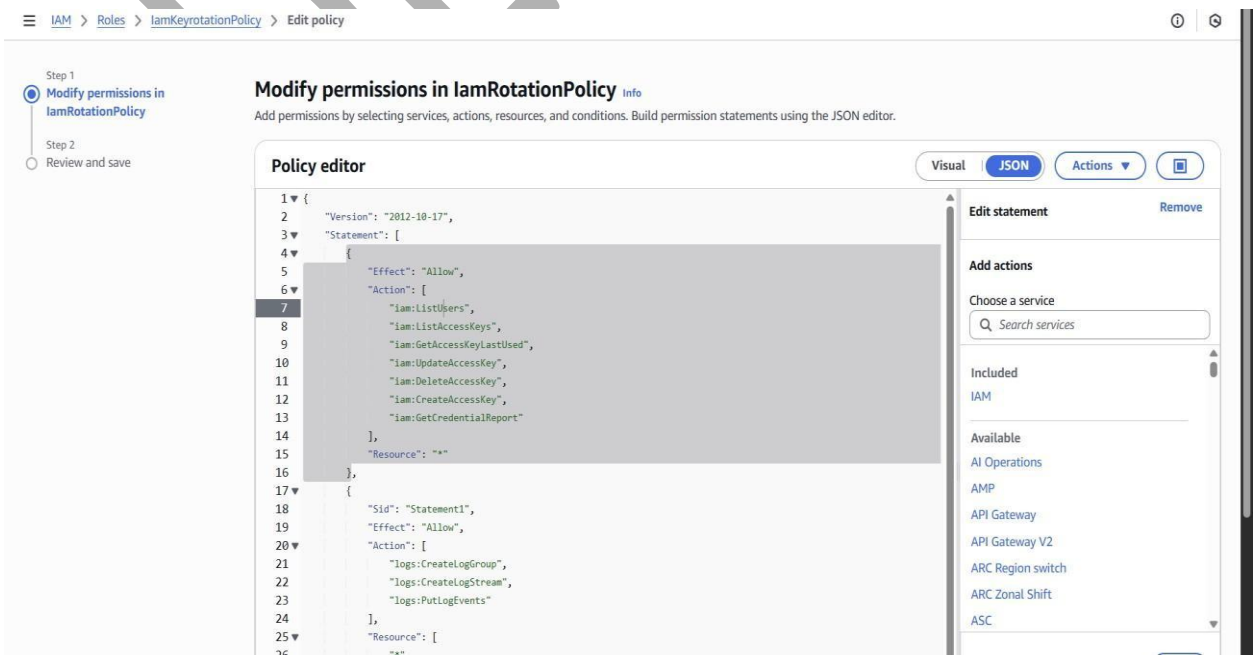
4.2 Automated Key Rotation

I implemented an automated access key rotation framework using:

- AWS Lambda



- IAM custom policies



- Amazon EventBridge scheduling

Rotation logic was validated using short execution intervals before being deployed with standard production schedules.

This reduced the risk associated with long-lived credentials.

5. Role-Based Access Control (RBAC)

5.1 Administrative Access Model

I replaced permanent administrative privileges with controlled role assumption using AWS Security Token Service (STS).

A dedicated administrative role was created for elevated access.

Users received temporary privileges only when required.

5.2 Trust Policy Models

Two trust models were implemented:

- Direct user-based trust policies
- Group-based trust policies

This supported both centralized and distributed administrative workflows.

6.3 Least Privilege Enforcement and Access Analysis I

created custom roles for:

- Development
- Audit
- Network Operations

Each role followed least-privilege principles and enforced MFA requirements.

I enabled IAM Access Analyzer to continuously evaluate policies for:

- Excessive permissions
- Wildcard usage
- Unused access

- Public exposure

Analyzer findings were reviewed regularly and remediated when necessary.

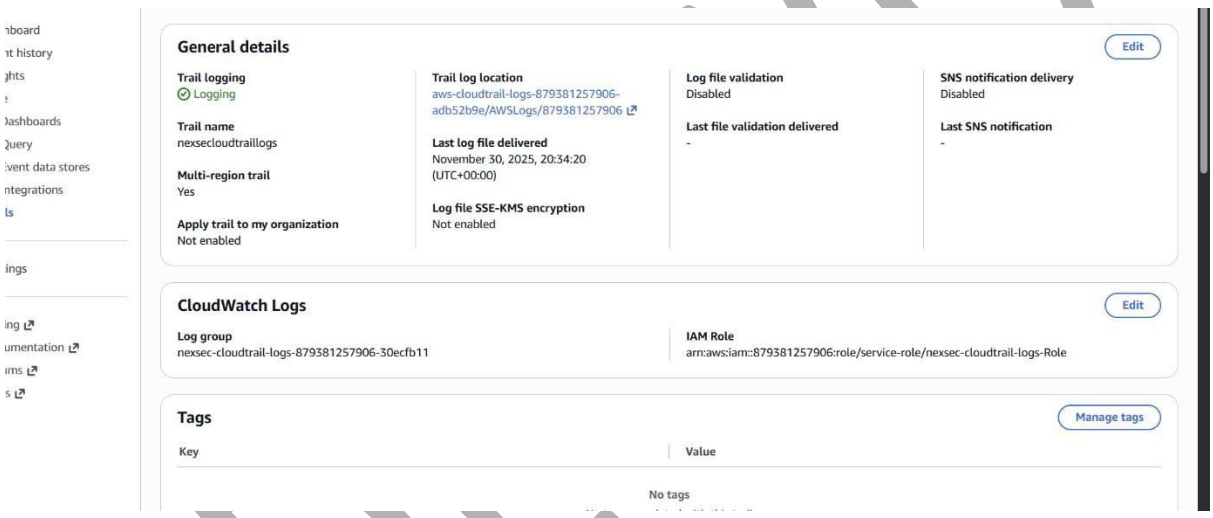
6. Logging, Monitoring, and Operational Visibility

6.1 CloudTrail Implementation

I configured a centralized CloudTrail trail to capture all API activity.

CloudTrail Insights and Data Events were enabled to detect abnormal behavior, privilege escalation attempts, and automation-related changes.

This provided a complete audit trail.

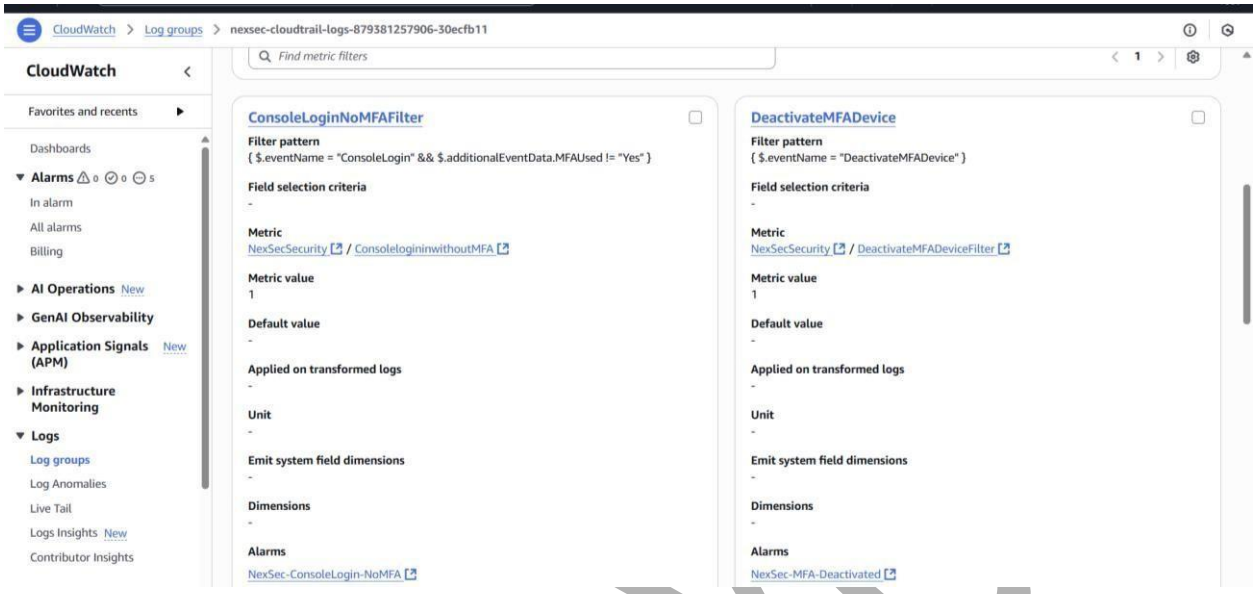


7.2 CloudWatch Integration and Metric Validation CloudTrail logs were

streamed into CloudWatch Log Groups.

I created metric filters for critical IAM actions, including:

- Unauthorized console access
- Access key creation and modification
- MFA deactivation



- Role assumption

Initially, metrics were not visible because no matching events had occurred.

To resolve this, I manually generated test IAM actions to trigger CloudTrail logs. Once events were recorded, metrics became available and alarms could be created.

This validation ensured monitoring worked as intended before production use.

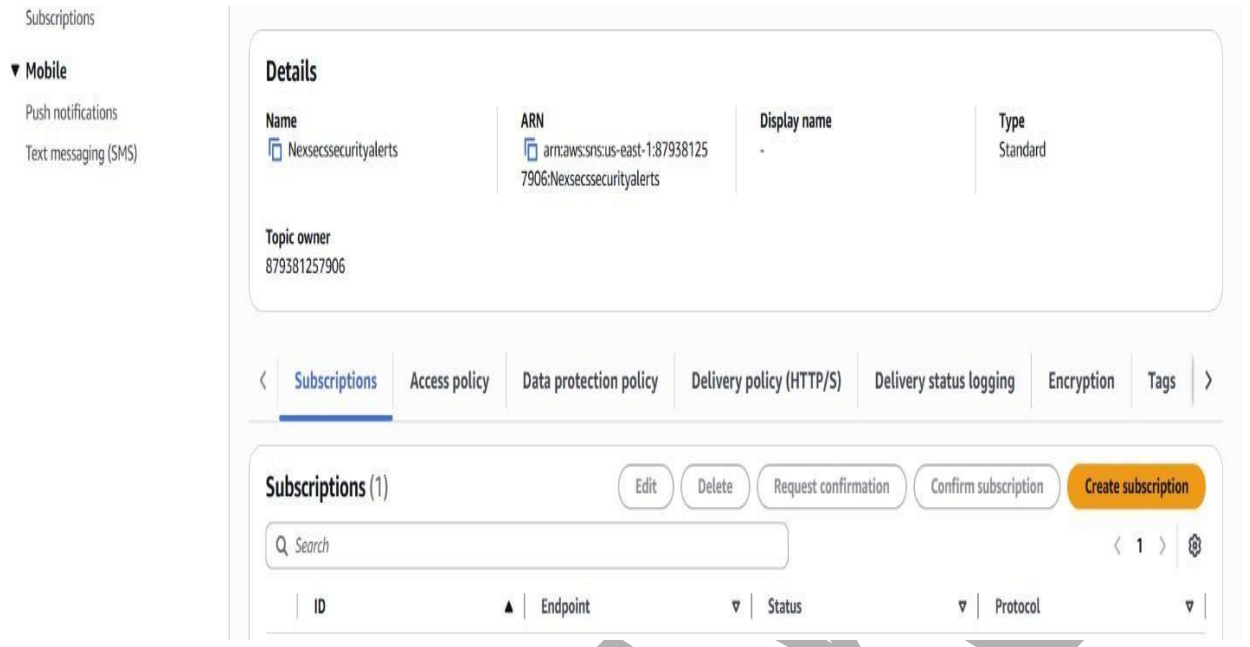
---

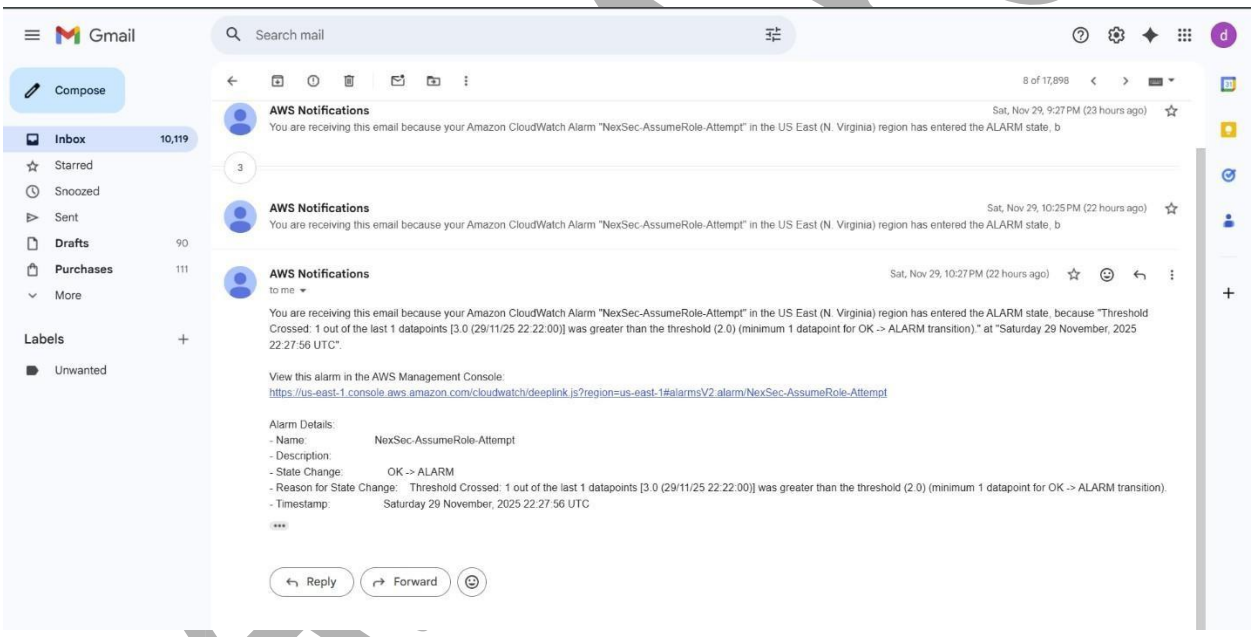
7. Alerting and Encryption Management

7.1 Alert Configuration

I configured CloudWatch alarms for each high-risk IAM event.

SNS was used to distribute security alerts to the operations team.





7.2 KMS Encryption Resolution

During testing, alerts were triggered but not delivered.

Root cause analysis showed that encrypted SNS topics lacked proper KMS permissions for CloudWatch.

I updated the KMS key policy to grant publishing rights.

After remediation, alert delivery was restored and verified.

8. Advanced Security and Compliance Controls

8.1 Threat Detection

I enabled AWS GuardDuty to provide continuous monitoring for:

- Credential compromise
- Suspicious API activity
- Network anomalies

Findings were integrated into incident response workflows.

9.2 Configuration and Compliance Monitoring I implemented

AWS Config to track:

- Root account MFA status
- IAM password policies
- Public access settings
- Unused roles
- Over-permissioned policies

Configuration changes were logged and reviewed for compliance.

---

9. Documentation, Governance, and Audit Support

All IAM roles, policies, workflows, and architectural decisions were documented.

This documentation supported:

- Internal reviews
- Security audits
- Incident investigations
- Knowledge transfer

Audit logs, credential reports, and access reviews were retained according to governance requirements.

---

10. Outcomes and Impact

This implementation delivered measurable improvements:

- Established enterprise-grade IAM governance
- Enforced strong authentication standards
- Reduced credential-related risk
- Improved visibility into identity activity
- Centralized monitoring and alerting
- Enhanced audit readiness
- Strengthened operational resilience

---

11. Conclusion

I designed and implemented a secure, scalable, and auditable IAM framework that supports enterprise security and compliance requirements.

Through automation, least-privilege enforcement, continuous monitoring, and strong governance, this environment enables secure operations and long-term growth.

This implementation reflects hands-on experience in building and operating cloud identity systems in production-like enterprise environments.