AWS Cloud Security Incident Response Automation Lab

Technical Implementation Report

Project Title

Implementation of Automated Cloud Security Monitoring and Incident Response Remediation Using AWS Config, Amazon S3, IAM, Terraform, and AWS Systems Manager

Submitted By

Name: Suraj Somkuwar

Domain: AWS Cloud Security / DevSecOps

Project Platform

Cloud Provider: Amazon Web Services (AWS)

AWS Region: Asia Pacific (Mumbai) — ap-south-1

Technologies and Services Used

Category	Technology / Service
Infrastructure as Code	Terraform
Cloud Platform	Amazon Web Services (AWS)
Storage Service	Amazon S3
Security Monitoring	AWS Config
Identity Management	AWS IAM
Automation Engine	AWS Systems Manager Automation
Command Line Interface	AWS CLI
Authentication	IAM User (terraform-admin)

Contents

Technical Implementation Report 1
Project Title 1
Submitted By 1
Project Platform 1
Technologies and Services Used 1
Executive Summary 3
1. Project Objective 3
Project Goals 3
2. Project Scope 3
Amazon S3 Security 3
Identity and Access Management 4
Security Monitoring 4
Automated Remediation 4
3. Security Problem Statement 4
Proposed Solution 4
4. Tools and Technologies Explanation 5
Terraform 5
Amazon S3 5
AWS Config 5
AWS IAM 5
AWS Systems Manager Automation 5
AWS CLI 5
5. Solution Architecture Overview 5
6. Detailed Step-by-Step Implementation 6
Phase 1: Terraform Infrastructure Deployment 6
Phase 2: Amazon S3 Secure Storage Configuration 8
Phase 3: IAM Security Configuration 9
Phase 4: AWS Config Monitoring Implementation 11
Phase 5: Automated Remediation Implementation 13
Phase 6: Security Incident Simulation and Validation 15
Phase 7: Detection, Response and Recovery Workflow 16
Phase 8: Implementation Outcome 18
7. Conclusion 18

Executive Summary

Cloud environments are highly dynamic, and security misconfigurations can occur due to accidental changes, improper permissions, or human errors. One of the most common AWS security risks is unintended public exposure of Amazon S3 buckets, which can lead to unauthorized data access and compliance violations.

This project implements an automated cloud security monitoring and incident response solution using AWS native security services combined with Infrastructure as Code (IaC).

The solution uses Terraform to provision AWS security resources, AWS Config to continuously monitor resource compliance, and AWS Systems Manager Automation to automatically remediate security violations.

A controlled security incident was simulated by introducing a temporary public-read permission on an Amazon S3 bucket. AWS Config detected the violation using the managed rule s3-bucket-public-read-prohibited. The remediation workflow used AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock to restore secure configuration.

1. Project Objective

The objective of this project was to design and implement an automated AWS cloud security monitoring and remediation framework that detects unauthorized Amazon S3 public access configurations and automatically applies corrective security controls.

Project Goals

- Secure AWS Resource Deployment using Terraform Infrastructure as Code.
- Continuous Security Monitoring using AWS Config.
- Security Violation Detection using AWS Config managed rules.
- Automated Incident Response using AWS Config Remediation and Systems Manager Automation.
- Security Validation through controlled incident simulation and remediation testing.

2. Project Scope

The project covers the following AWS security areas:

Amazon S3 Security

- Secure S3 bucket configuration
- Server-side encryption
- Bucket versioning
- Block Public Access controls
- Secure bucket policies

Identity and Access Management

- IAM roles
- IAM policies
- Dedicated Terraform administrative identity

Security Monitoring

- AWS Config Recorder
- AWS Config Delivery Channel
- AWS Config Compliance Rules

Automated Remediation

- AWS Config remediation configuration
- Systems Manager Automation execution
- Automatic security recovery

3. Security Problem Statement

Amazon S3 buckets are frequently targeted in cloud security incidents because incorrect permissions can expose sensitive data publicly. Common causes include incorrect bucket policies, misconfigured ACL permissions, accidental public access enablement, and lack of continuous monitoring.

Proposed Solution

The implemented solution introduces automated security governance:

Prevent
↓
Monitor
↓
Detect
↓
Respond
↓
Recover

Security Requirement	**AWS Service**
Resource Monitoring	AWS Config
Data Protection	Amazon S3
Identity Control	AWS IAM
Automation Response	AWS Systems Manager
Deployment Automation	Terraform

4. Tools and Technologies Explanation

Terraform

Terraform was used as Infrastructure as Code to automate AWS resource creation. It provides repeatable deployments, version-controlled infrastructure, and auditable changes.

Amazon S3

Amazon S3 was used as the storage service for AWS Config configuration history and logs. Security controls implemented include encryption, versioning, and public access protection.

AWS Config

AWS Config was used as the continuous compliance monitoring service. It provides resource configuration tracking, compliance evaluation, and security drift detection.

AWS IAM

IAM was used to control authentication and authorization. Implemented identities included terraform-admin, AWS Config service role, and remediation automation role.

AWS Systems Manager Automation

Systems Manager Automation was used for automated security remediation using AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock.

AWS CLI

AWS CLI was used to verify the active AWS identity using **aws sts get-caller-identity** and confirm Terraform CLI authentication.

5. Solution Architecture Overview

The implemented solution follows a security automation lifecycle where infrastructure deployment, continuous monitoring, threat detection, and automated remediation work together.

Terraform
↓
AWS Infrastructure Deployment
↓
AWS Config Recorder
↓
AWS Config Compliance Rule
↓
Security Violation Detection
↓
AWS Config Remediation
↓
AWS Systems Manager Automation

↓

S3 Security Configuration Restored

Component	AWS Service / Tool	Security Purpose
Infrastructure Deployment	Terraform	Creates repeatable and auditable AWS resources
Storage Security	Amazon S3	Stores AWS Config logs securely
Monitoring	AWS Config	Evaluates resource compliance continuously
Identity Management	AWS IAM	Controls authentication and authorization
Automation Response	Systems Manager Automation	Executes corrective security actions

6. Detailed Step-by-Step Implementation

Phase 1: Terraform Infrastructure Deployment

Implementation Performed: Terraform was used as Infrastructure as Code to provision AWS security resources. It provides repeatable and auditable deployment.

- **Security Benefit:** Reduces manual configuration errors.
- **Operational Benefit:** Enables consistent deployment.
- **Governance Benefit:** Allows change review before deployment.

Evidence:

Figure 1: Terraform Initialization Successful

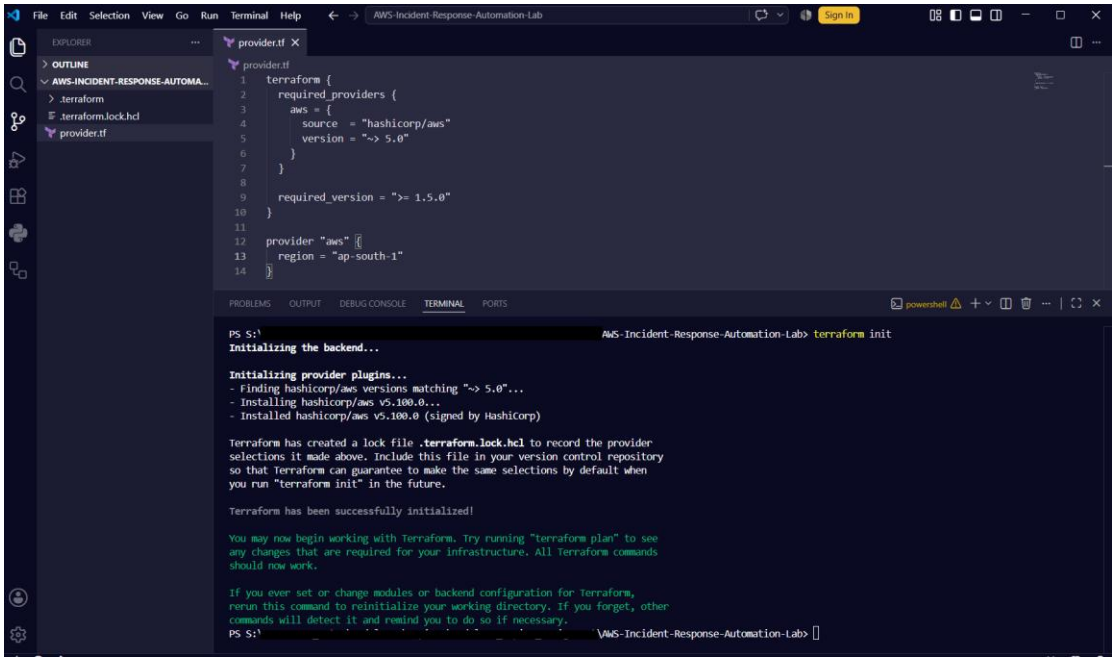


Figure 2: Terraform Configuration Validation Successful

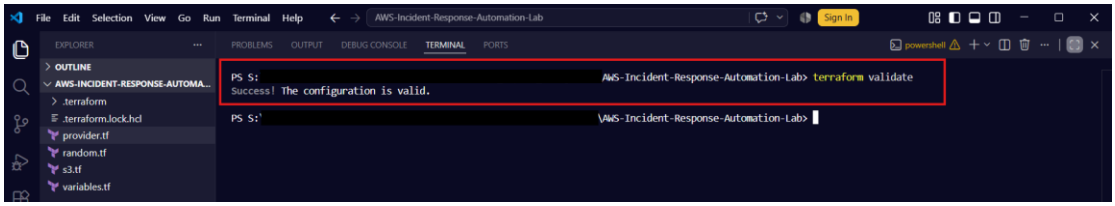


Figure 3: Terraform Execution Plan for AWS Infrastructure

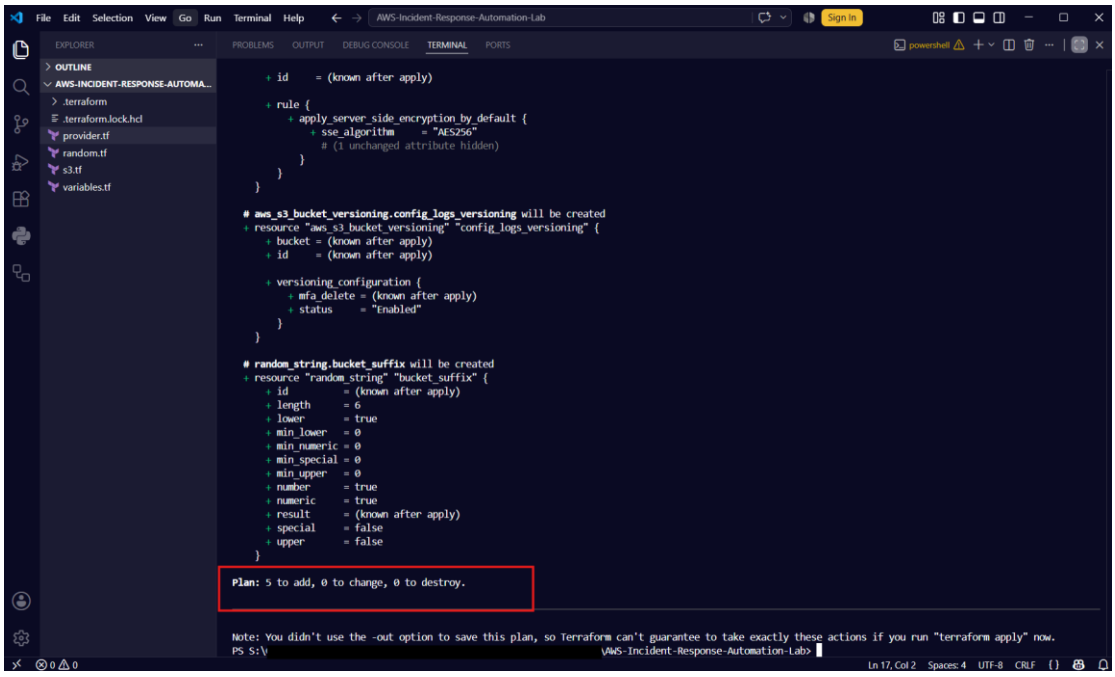
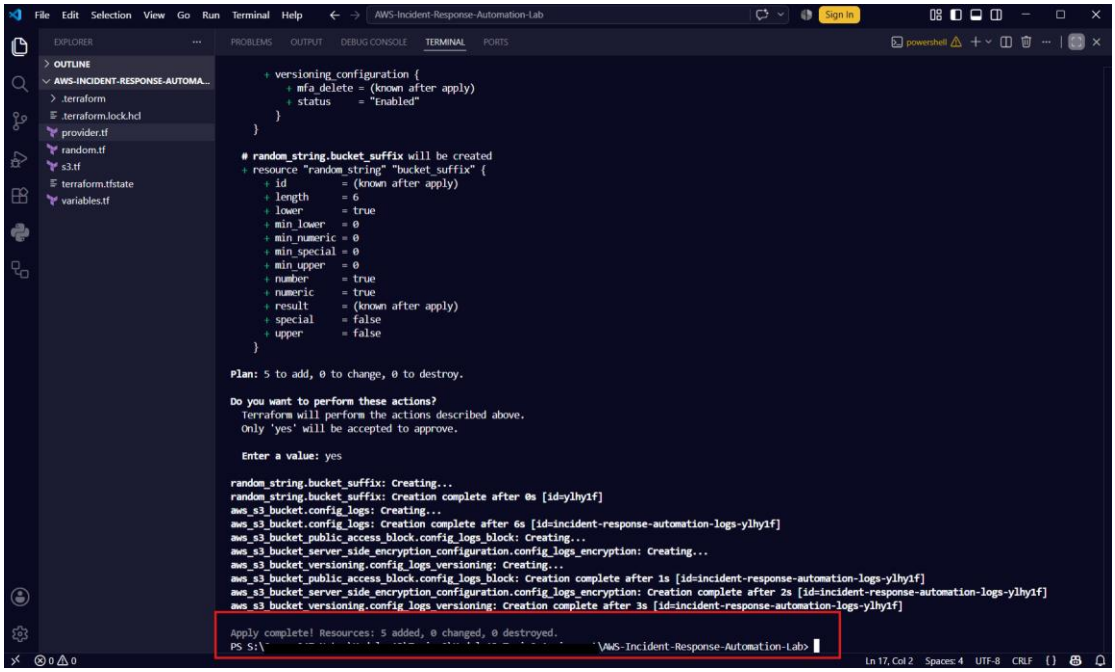


Figure 4: Terraform Infrastructure Deployment Successful



Phase 2: Amazon S3 Secure Storage Configuration

Implementation Performed: A dedicated S3 bucket was created for AWS Config logs with encryption, versioning, tagging, and public access protection.

- **Encryption:** Protects stored configuration data.
- **Versioning:** Supports recovery from accidental changes.
- **Block Public Access:** Prevents unintended exposure.

Evidence:

Figure 5: Amazon S3 Bucket Created Successfully

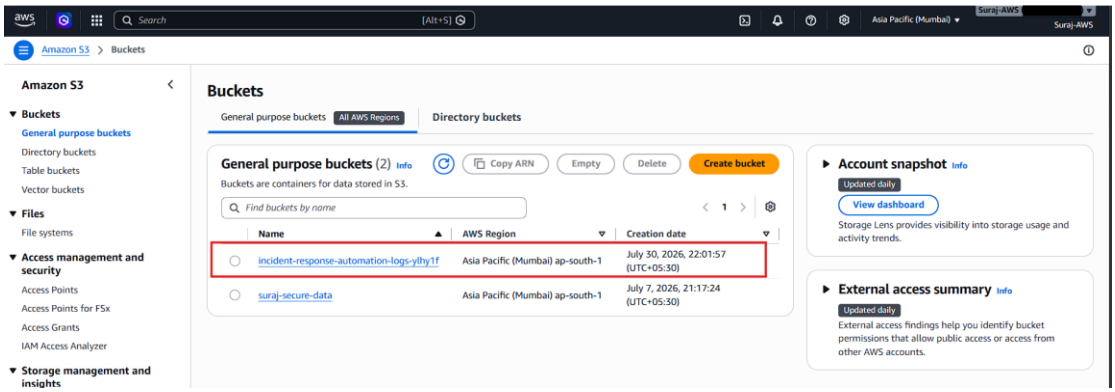


Figure 6: Amazon S3 Bucket Overview

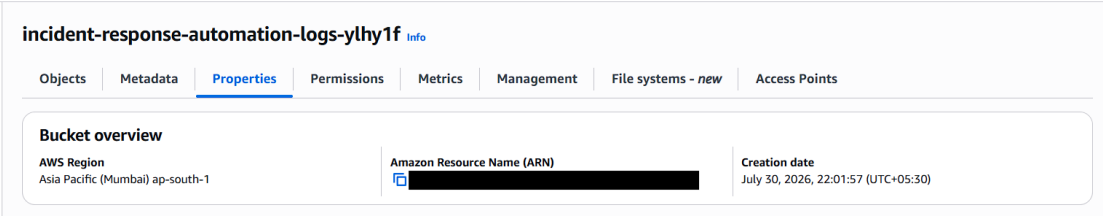


Figure 7: Amazon S3 Bucket Versioning Enabled

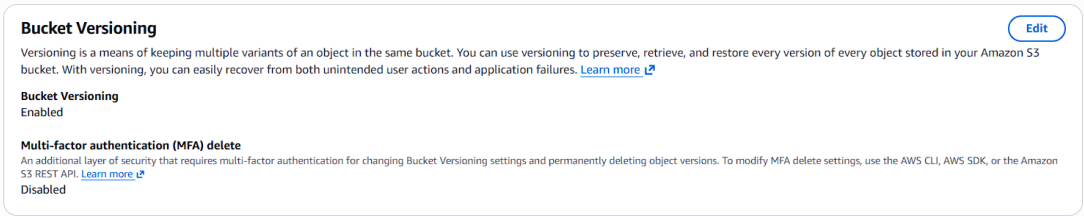


Figure 8: Amazon S3 Default Encryption Enabled

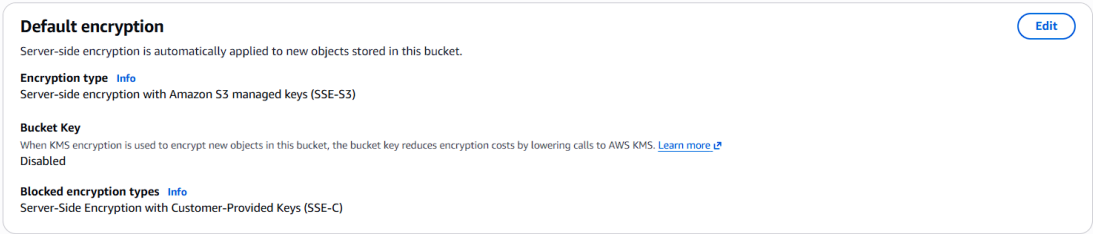


Figure 9: Amazon S3 Resource Tags Configuration

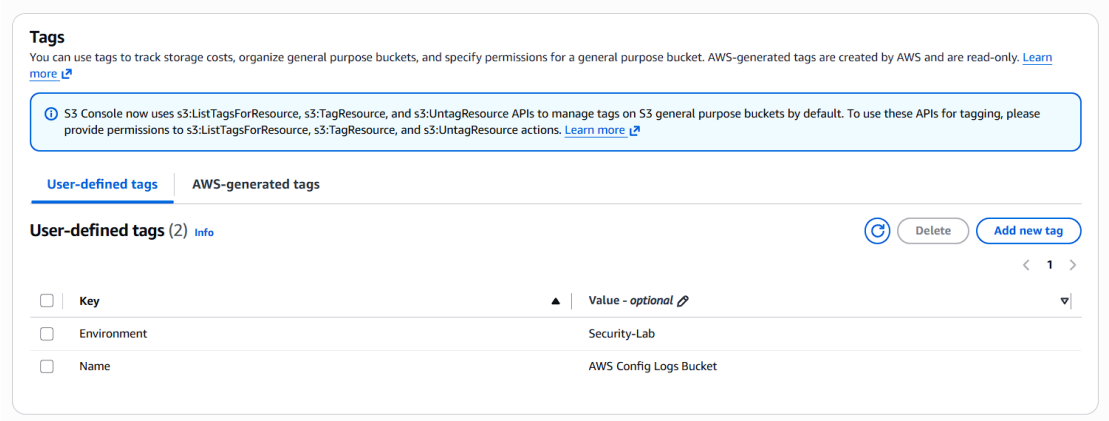
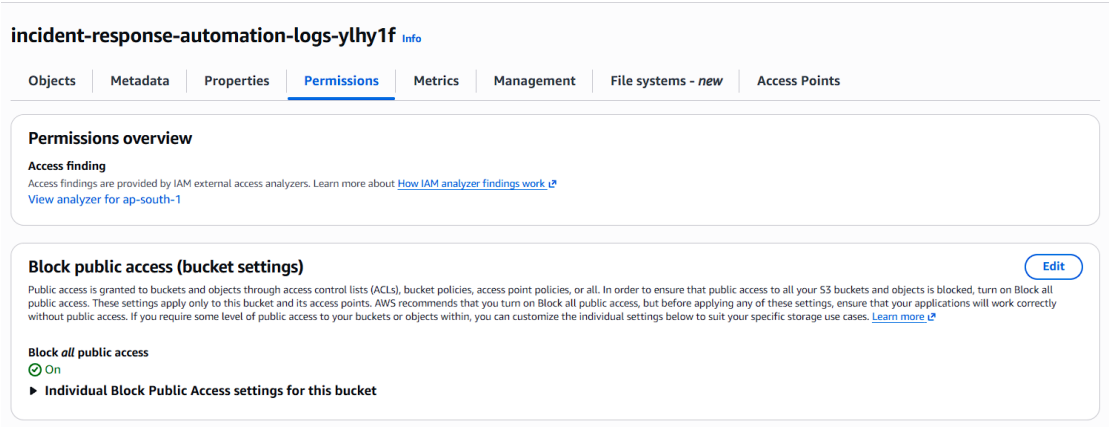


Figure 10: Amazon S3 Block Public Access Enabled



Phase 3: IAM Security Configuration

Implementation Performed: IAM roles and permissions were created for AWS Config and remediation automation. Terraform CLI used the **terraform-admin** IAM identity.

- **Least Privilege Principle:** Controls resource access.
- **Credential Separation:** Avoids dependency on root credentials.

Evidence:

Figure 11: IAM Remediation Role Validation

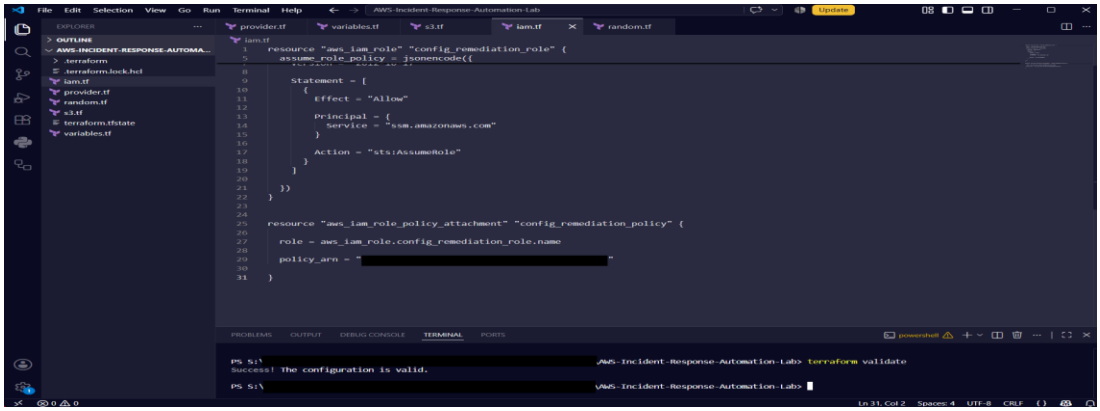


Figure 12: AWS Config Recorder and Delivery Channel Validation

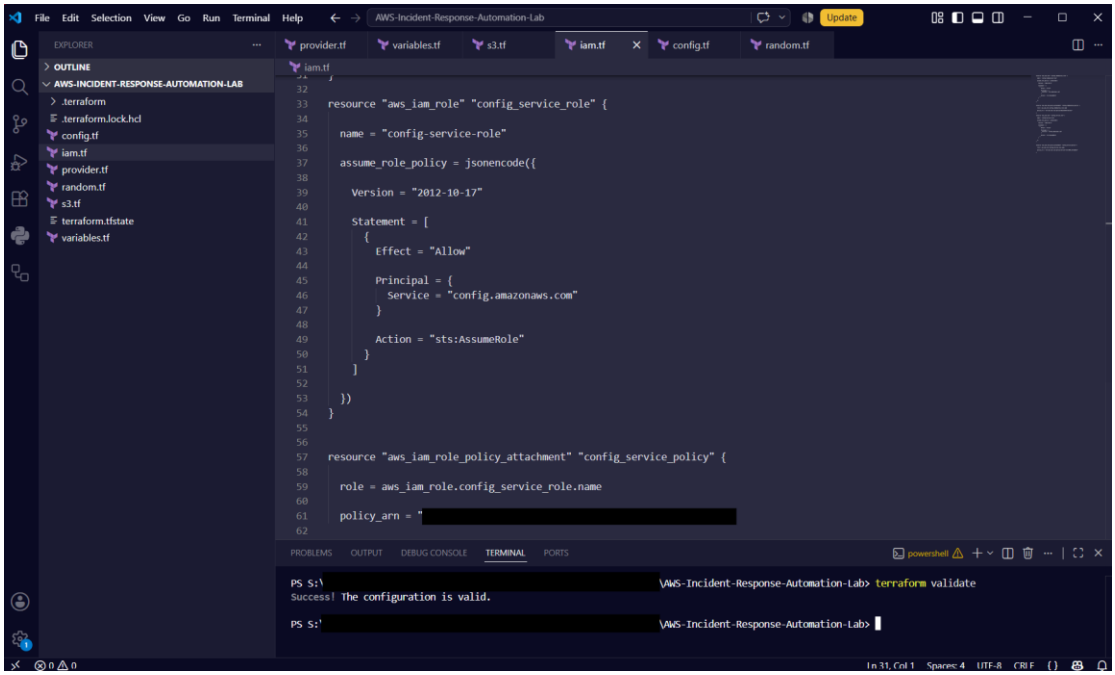


Figure 13: Final Terraform Plan Before Deployment

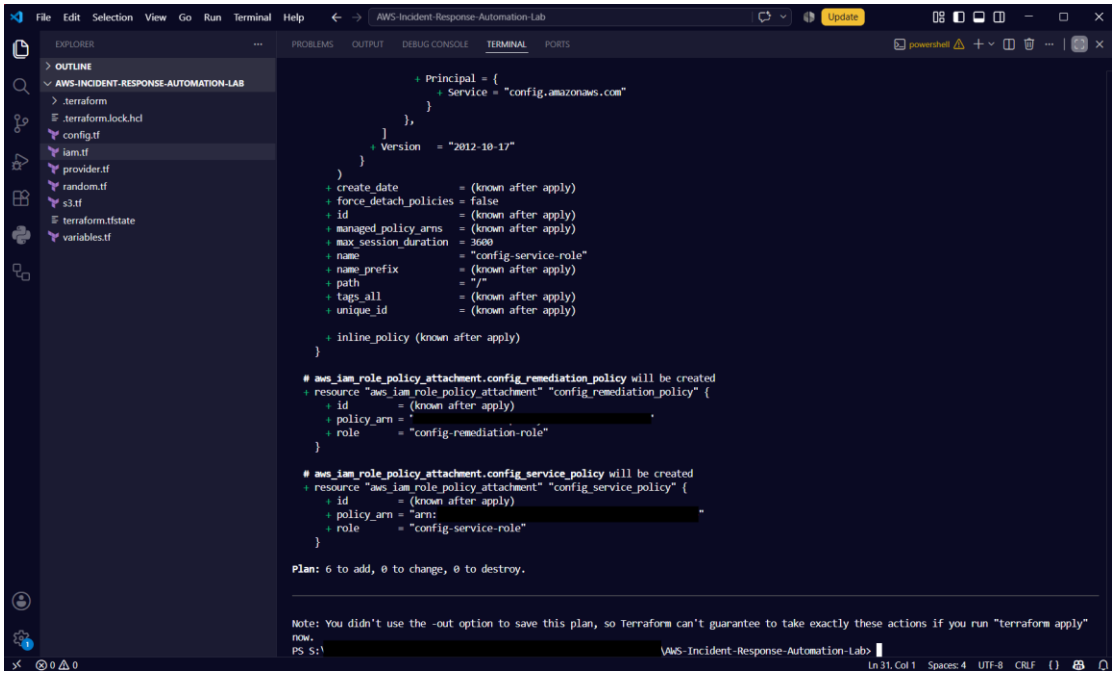
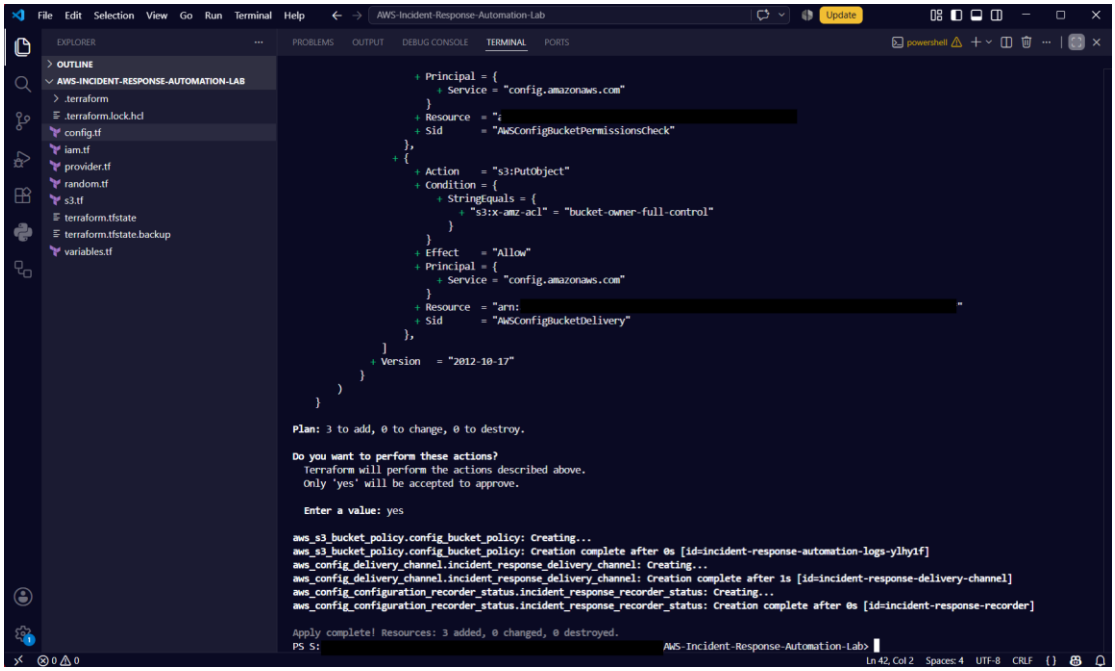


Figure 14: Terraform Apply Completed Successfully



Phase 4: AWS Config Monitoring Implementation

Implementation Performed: AWS Config Recorder, Delivery Channel, and the s3-bucket-public-read-prohibited rule were configured.

- **Continuous Monitoring:** Detects security drift.
- **Compliance Visibility:** Shows resource security status.

Evidence:

Figure 15: AWS Config Recorder Enabled

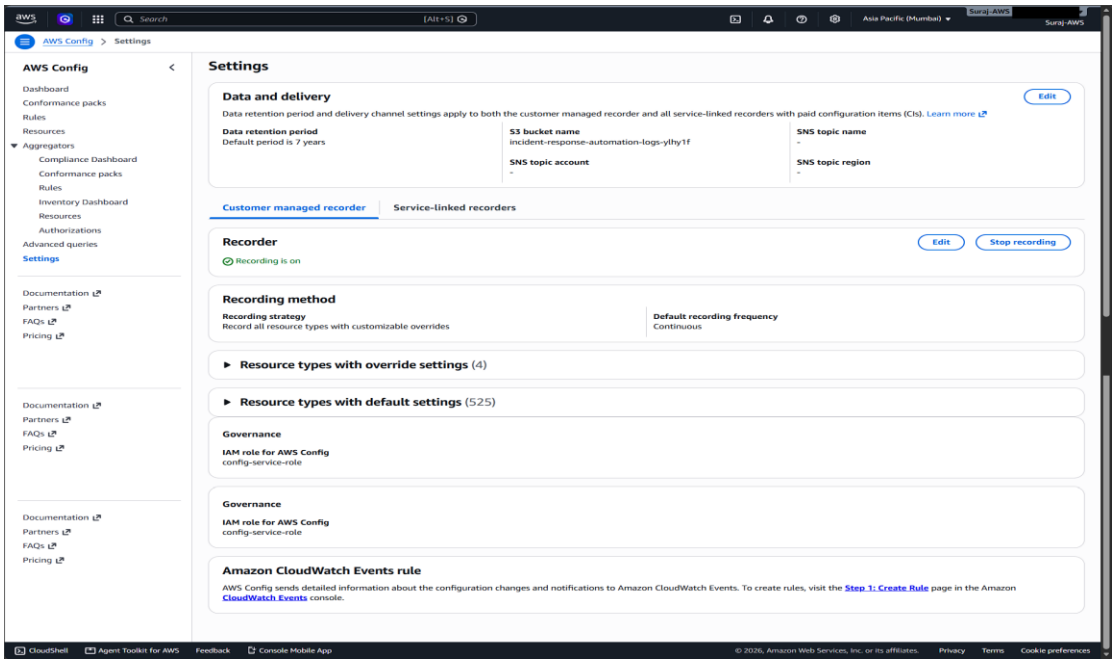


Figure 16: AWS Config Delivery Channel Configured

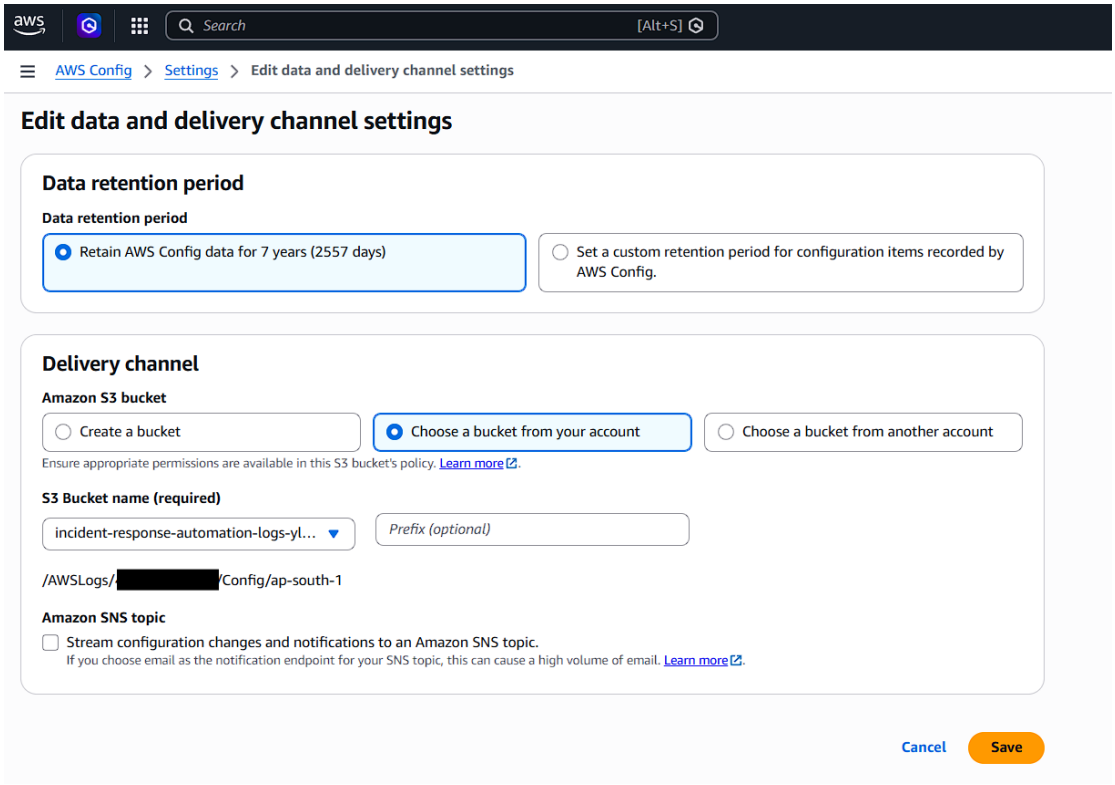


Figure 17: AWS Config Logs Delivered to Amazon S3

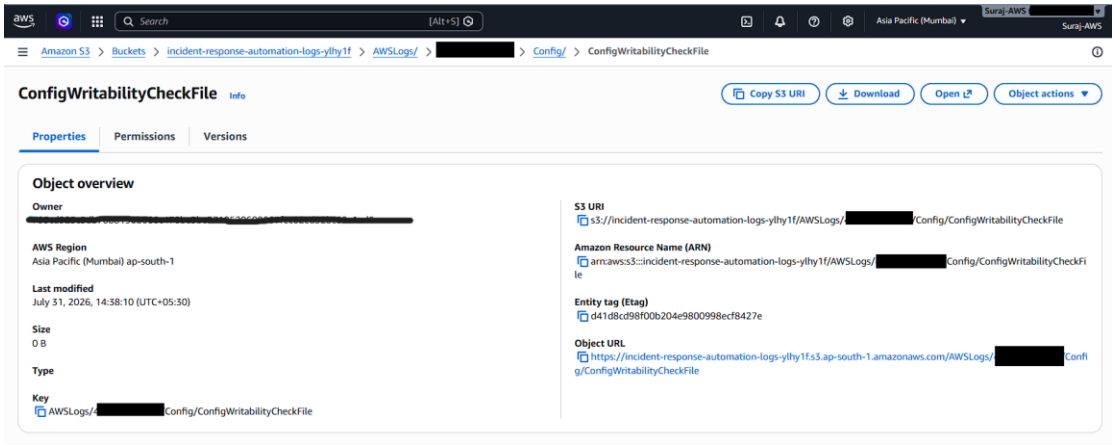


Figure 18: AWS Config Rule Review Before Creation

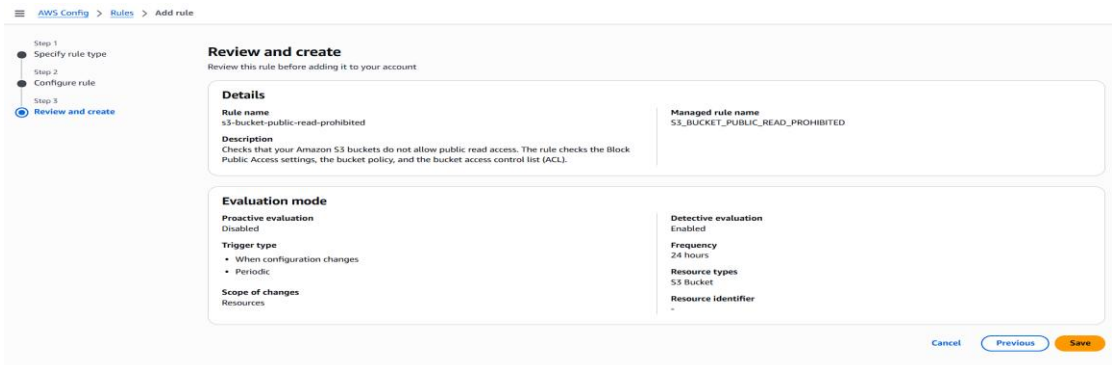
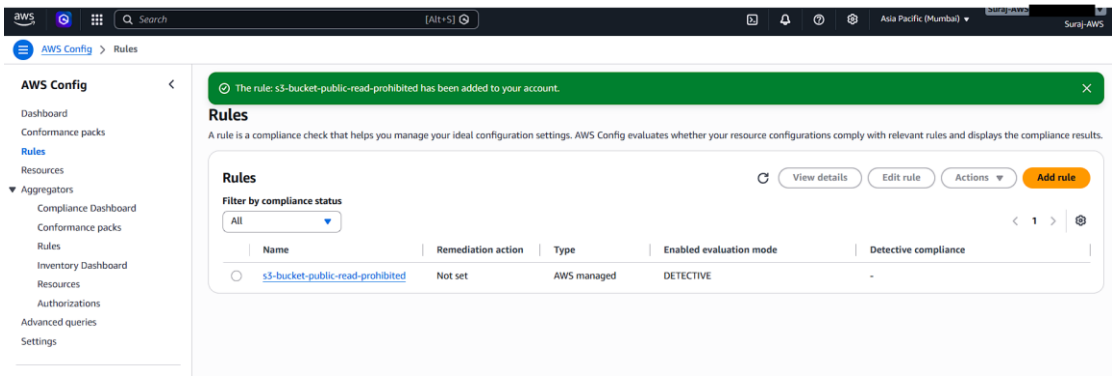


Figure 19: AWS Config Rule Created Successfully



Phase 5: Automated Remediation Implementation

Implementation Performed: AWS Config Remediation used AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock to automatically restore S3 security.

- **Faster Response:** Reduces manual effort.
- **Reduced Exposure Window:** Limits security risk duration.

Evidence:

Figure 20: Terraform Plan for Automatic Remediation

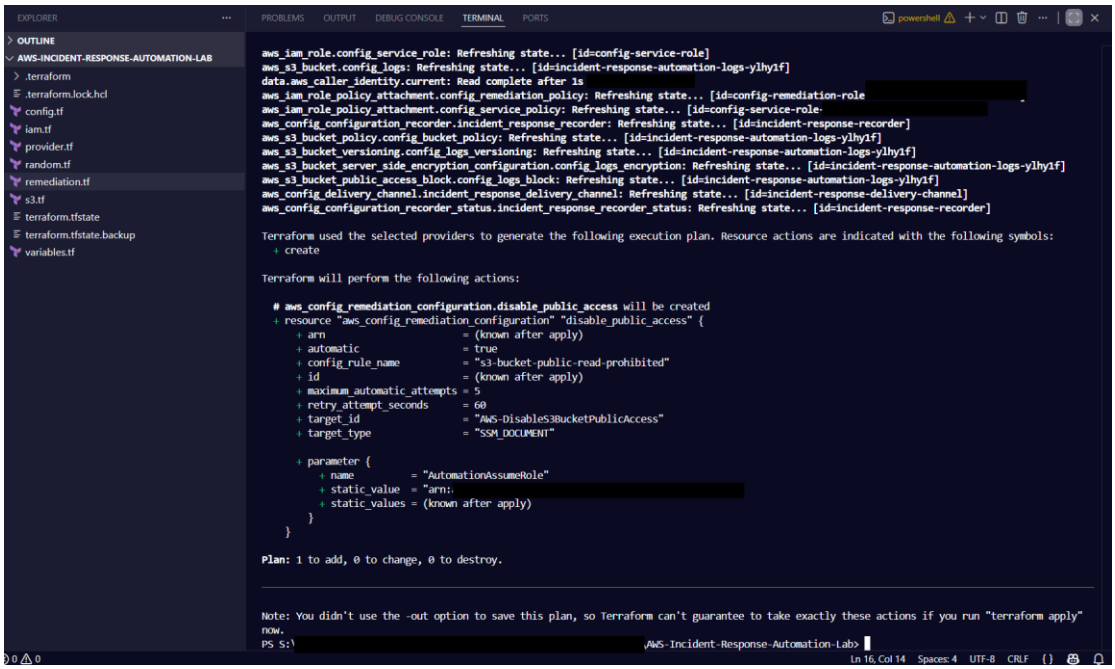


Figure 21: Final Terraform Plan for Automatic Remediation

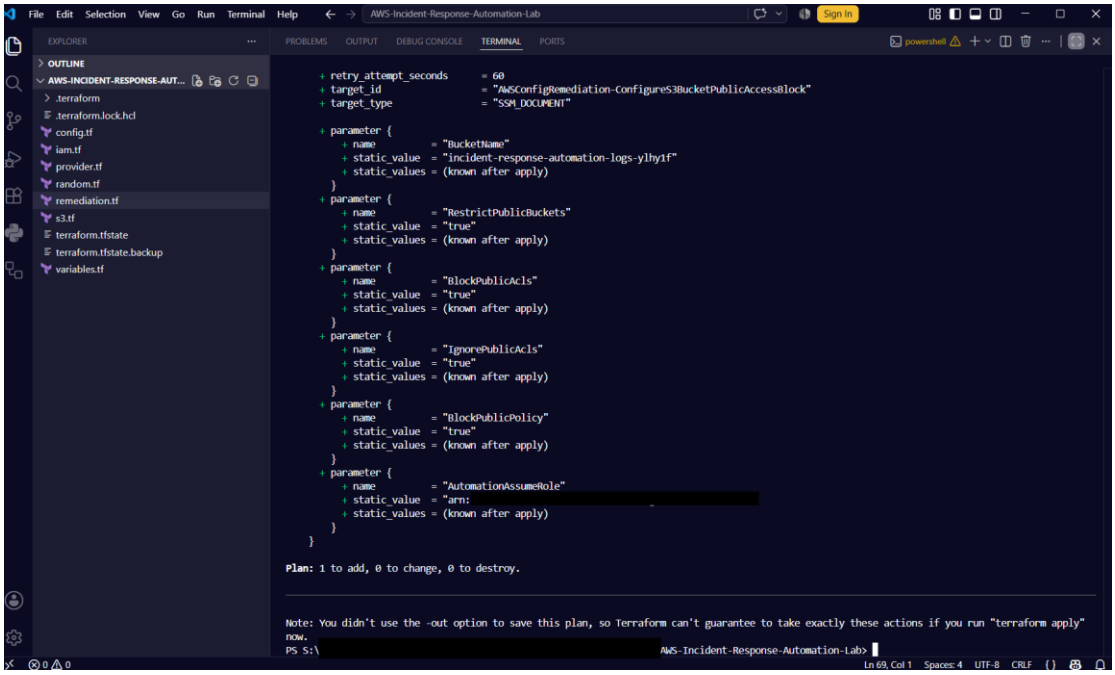


Figure 22: Automatic Remediation Successfully Deployed

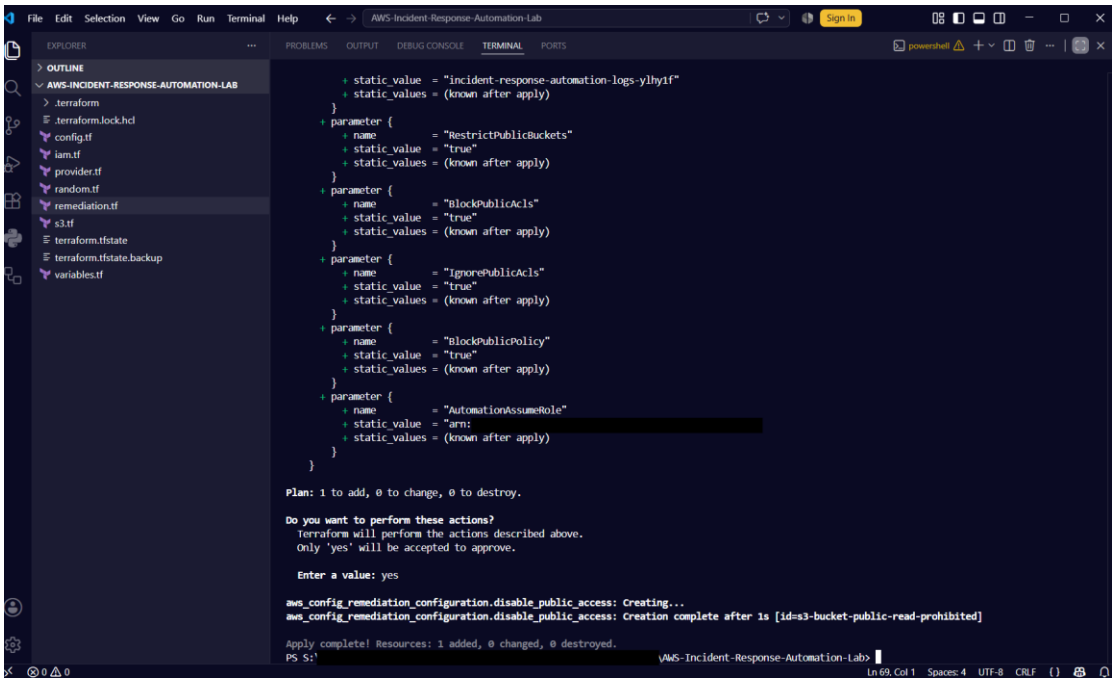


Figure 23: AWS Config Rule with Remediation Configuration

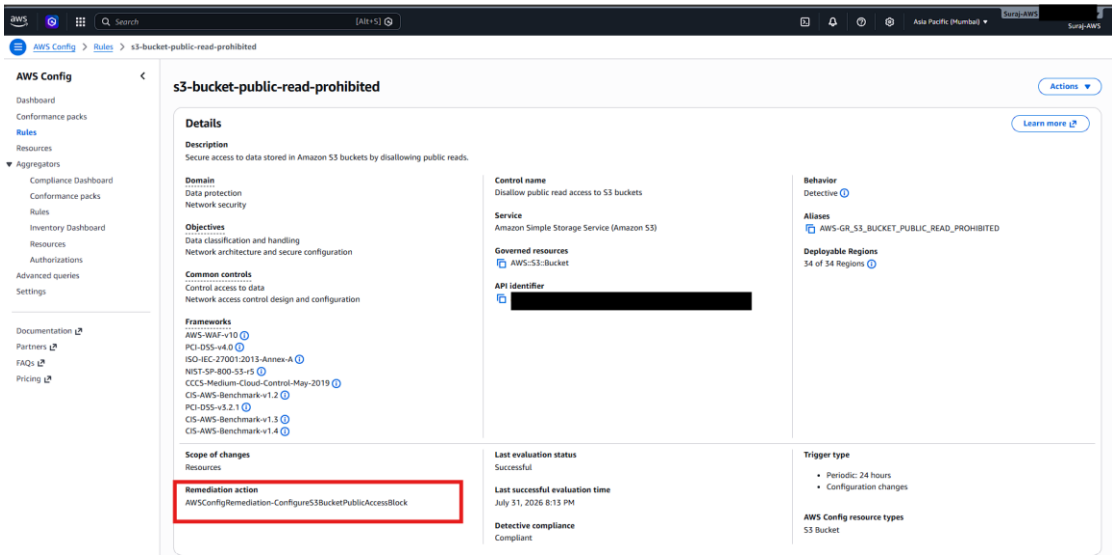
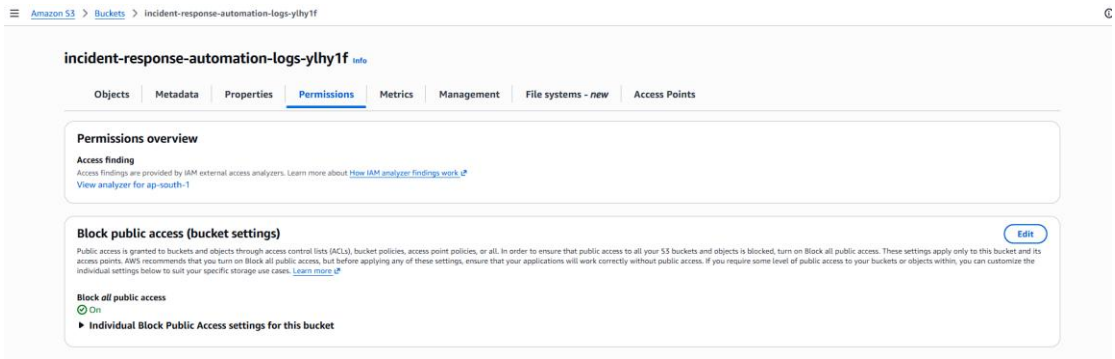


Figure 24: Initial Secure State of S3 Bucket



Phase 6: Security Incident Simulation and Validation

A controlled security incident was intentionally created to validate detection and automatic response capabilities.

A temporary S3 policy was created with:

Principal: *
Action: s3:GetObject
Effect: Allow

Evidence:

Figure 25: Public Access Violation Introduced for Testing

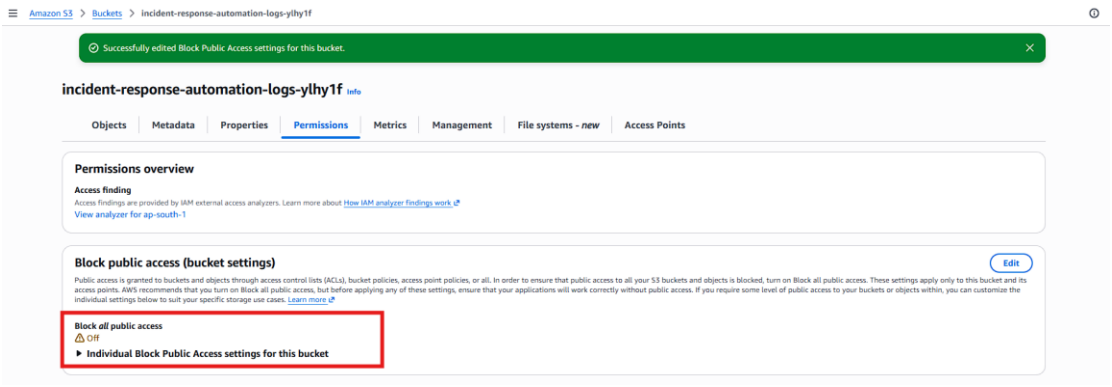
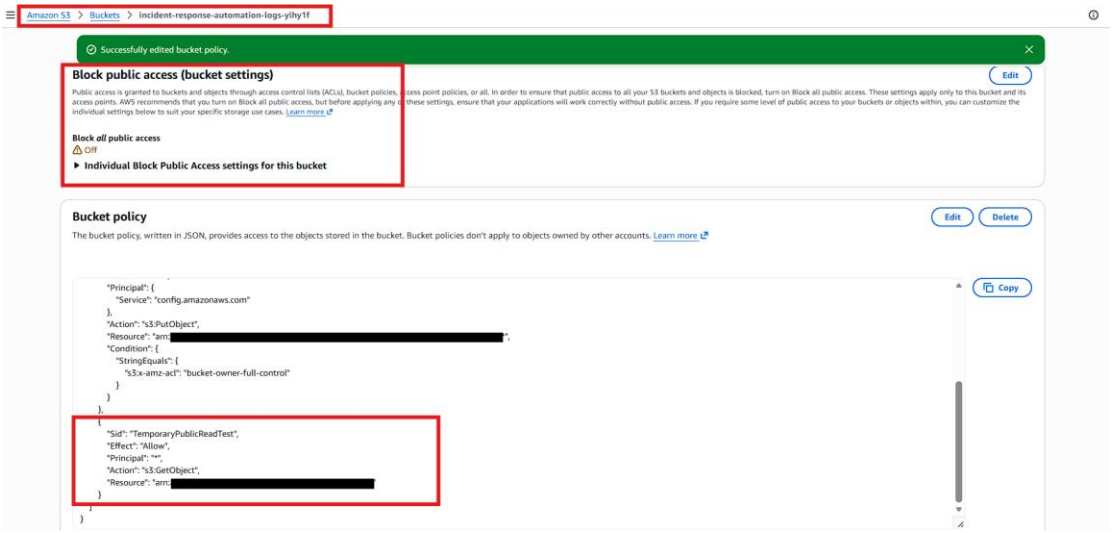


Figure 26: Temporary Public Read Bucket Policy Created



Phase 7: Detection, Response and Recovery Workflow

1. Public access misconfiguration created.
2. AWS Config detected the violation.
3. Compliance changed to Noncompliant.
4. Automatic remediation executed.
5. S3 protection was restored.
6. Compliance returned to Compliant.

Evidence:

Figure 27: AWS Config Detected Violation and Queued Automatic Remediation

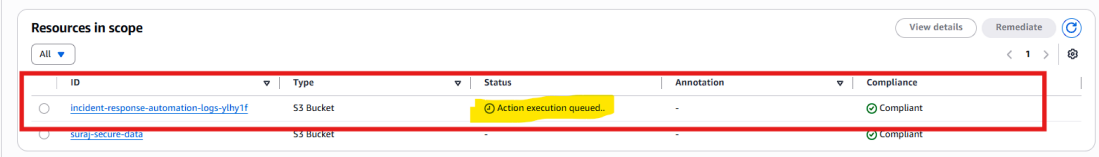


Figure 28: Automatic Remediation Executed Successfully

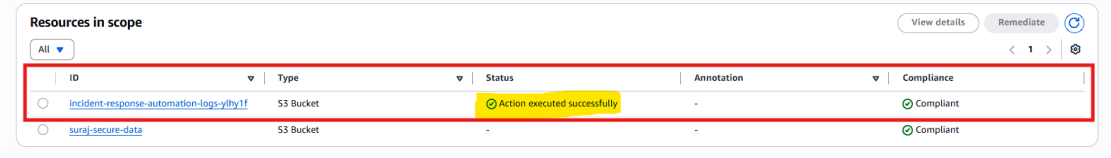


Figure 29: Amazon S3 Block Public Access Restored Automatically

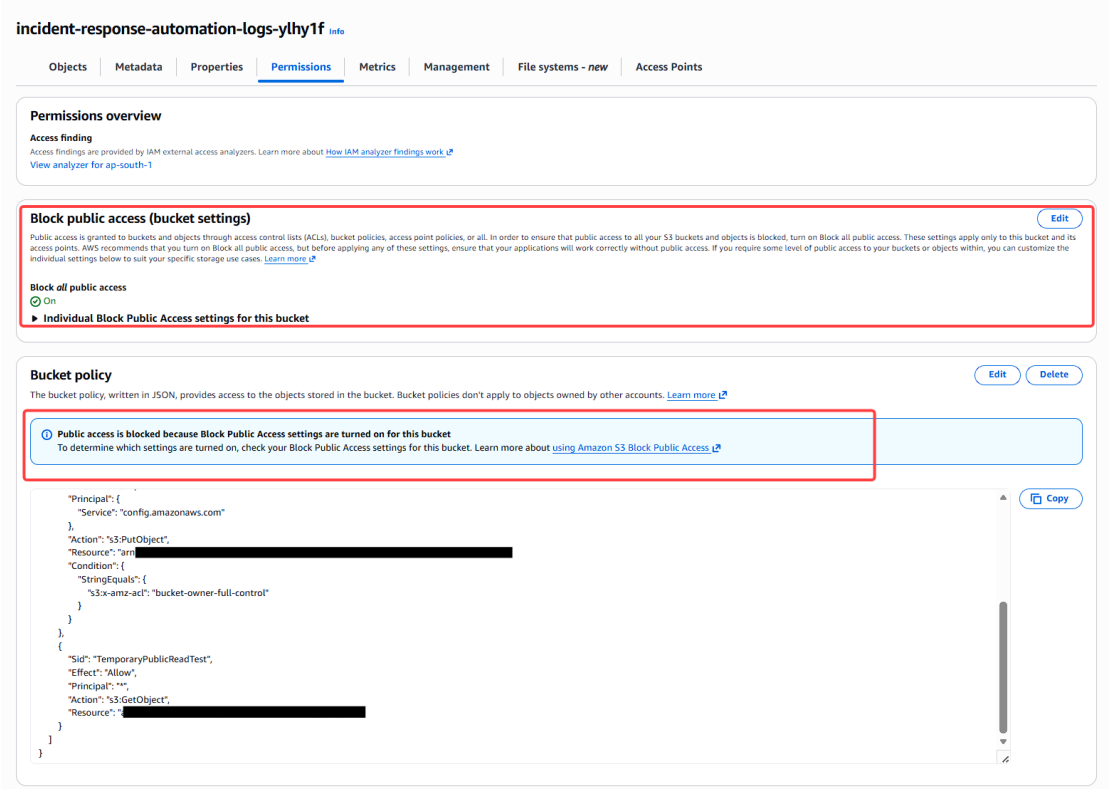
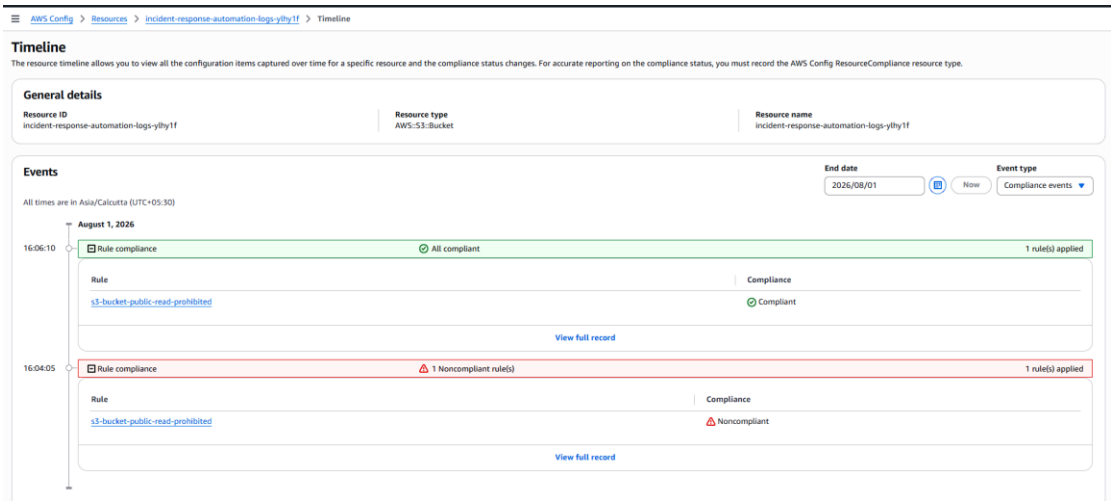


Figure 30: AWS Config Compliance History (Noncompliant → Compliant)



Phase 8: Implementation Outcome

The project successfully demonstrated an automated cloud security incident response process using AWS native security services.

- Terraform deployed security infrastructure.
- AWS Config monitored compliance.
- S3 public exposure was detected.
- Automatic remediation restored security.
- Public access validation confirmed protection.

Evidence:

Figure 31: Public Access Denied After Automatic Remediation

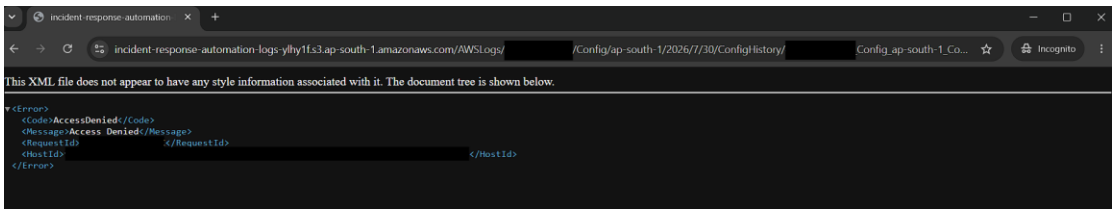
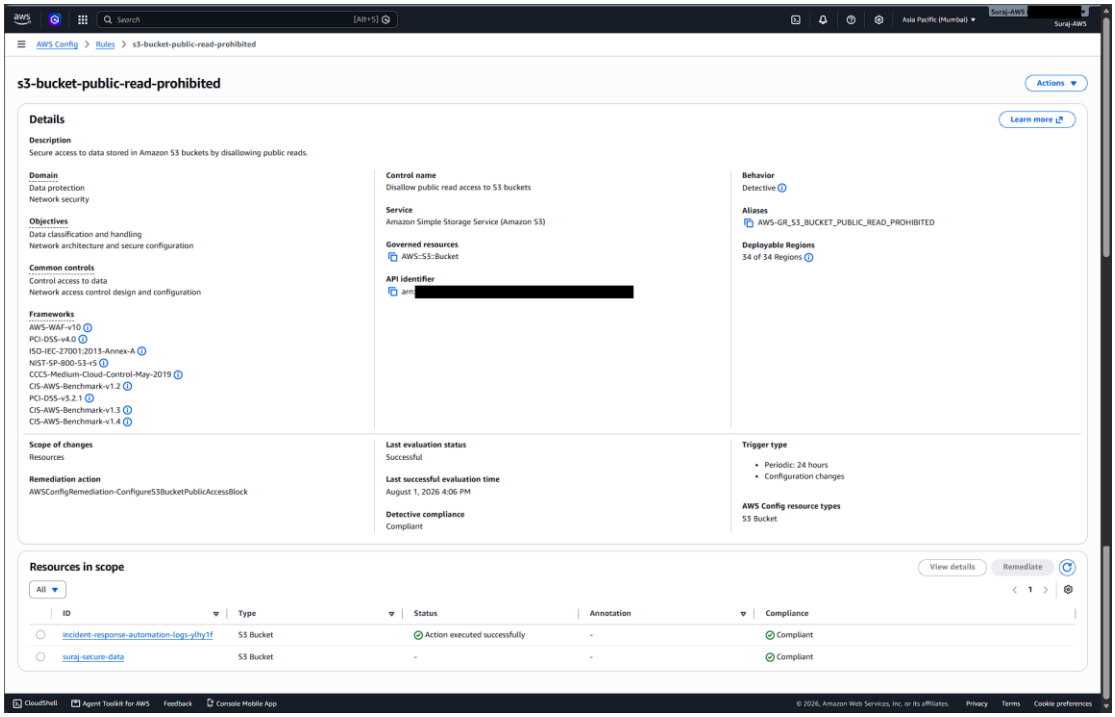


Figure 32: Final AWS Config Compliance Status Verified



7. Conclusion

This project successfully demonstrated the design and implementation of an automated AWS cloud security incident response solution using AWS native security services and Infrastructure as Code practices.

The solution combined **Terraform, Amazon S3, AWS IAM, AWS Config, and AWS Systems Manager Automation** to create a complete security monitoring and remediation workflow. Terraform enabled consistent and repeatable deployment of AWS security resources, while AWS Config continuously monitored resource configurations and evaluated compliance against security rules.

A controlled S3 public access misconfiguration was intentionally introduced to simulate a real-world cloud security incident. The AWS Config rule **s3-bucket-public-read-prohibited** successfully detected the security violation, changed the resource compliance status to **Noncompliant**, and triggered the configured automated remediation workflow. AWS Systems Manager Automation then executed **AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock** to restore secure S3 public access settings automatically.

The validation process confirmed that the automated response mechanism successfully reduced manual intervention, minimized the security exposure window, and restored the S3 bucket to a compliant security state. The project also demonstrated important AWS security practices, including least privilege access through IAM, avoiding root account dependency, secure S3 configuration, continuous compliance monitoring, and automated incident response.

Overall, this implementation highlights the importance of automation-driven security in modern cloud environments. By integrating preventive controls, continuous monitoring, automated detection, and remediation capabilities, organizations can improve their cloud security posture, reduce operational risks, and respond faster to security incidents. This project provides a practical foundation for implementing scalable Cloud Security and DevSecOps practices in real-world AWS environments.