

# 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 `AWSSystemsManager-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

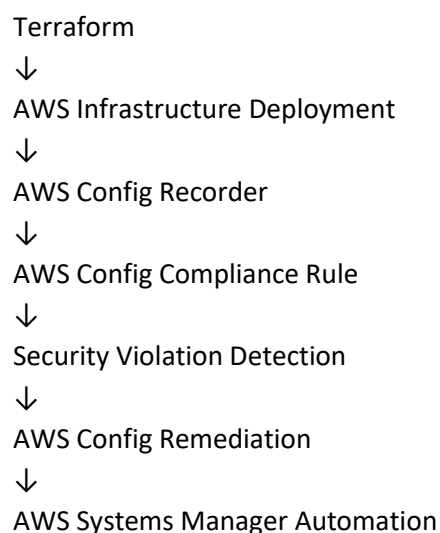
Systems Manager Automation was used for automated security remediation using AWSConfigRemediation-ConfigureS3BucketPublicAccessBlock.

### 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.**





## 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

```
provider.tf
1 terraform {
2   required_providers {
3     aws = {
4       source = "hashicorp/aws"
5       version = "~> 5.0"
6     }
7   }
8   required_version = ">= 1.5.0"
9 }
10
11 provider "aws" {
12   region = "ap-south-1"
13 }
14

PS S:\ AWS-Incident-Response-Automation-Lab> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding hashicorp/aws versions matching "~> 5.0"...
- Installing hashicorp/aws v5.100.0...
- Installed hashicorp/aws v5.100.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

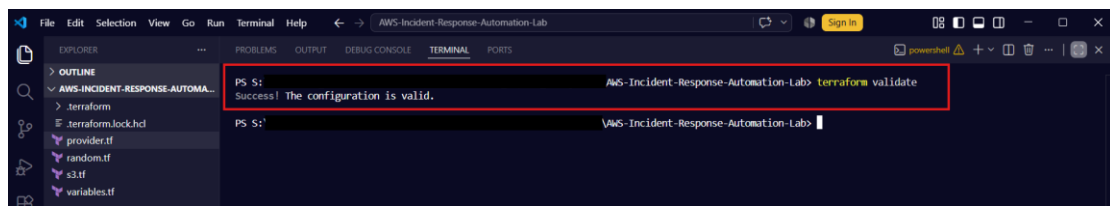
Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.

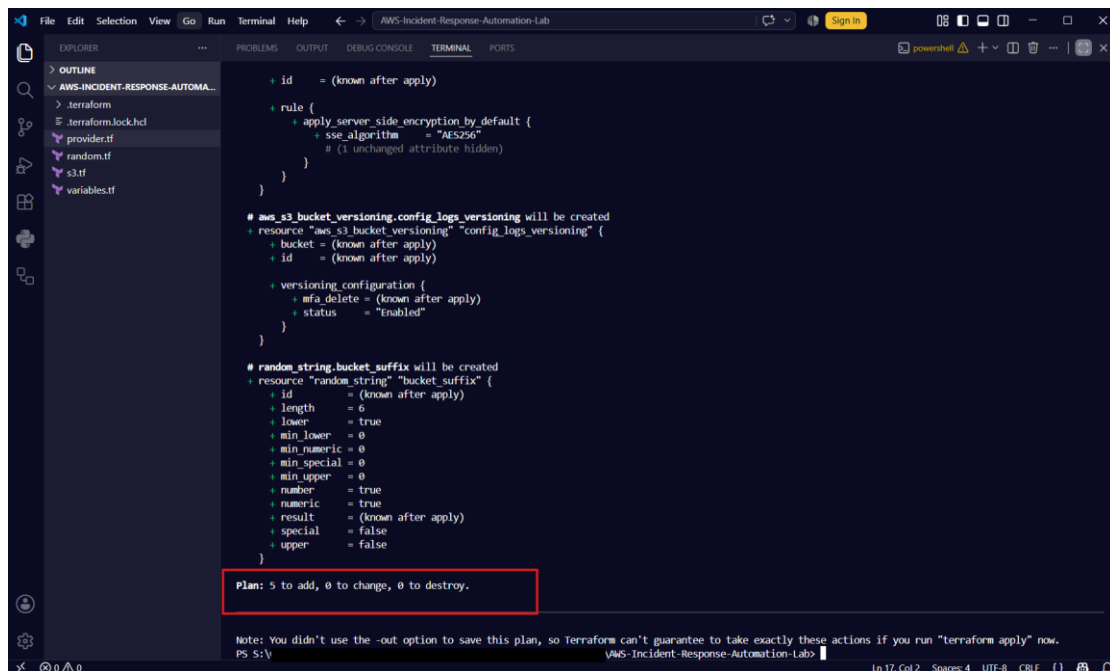
PS S:\ AWS-Incident-Response-Automation-Lab>
```

Figure 2: Terraform Configuration Validation Successful



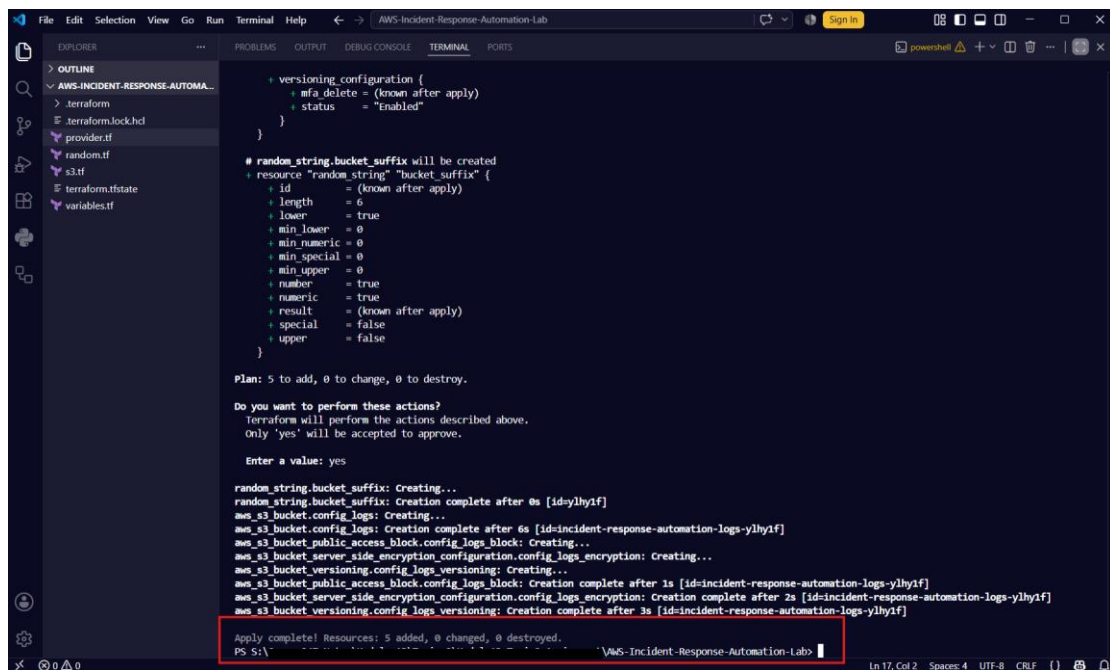
The screenshot shows a PowerShell terminal window in VS Code. The command `terraform validate` has been executed, and the output is "Success! The configuration is valid." The terminal title bar indicates the current directory is `AWS-Incident-Response-Automation-Lab`.

Figure 3: Terraform Execution Plan for AWS Infrastructure



The screenshot shows the output of `terraform plan` in a PowerShell terminal. The plan lists several resources to be added, including `aws_s3_bucket_versioning.config_logs_versioning` and `random_string.bucket_suffix`. A summary line states: "Plan: 5 to add, 0 to change, 0 to destroy." The terminal also shows a note about the `-out` option and the current directory.

Figure 4: Terraform Infrastructure Deployment Successful



The screenshot shows the output of `terraform apply` in a PowerShell terminal. The plan is approved, and the resources are created. The output shows the creation of `random_string.bucket_suffix`, `aws_s3_bucket.config_logs`, `aws_s3_bucket_public_access_block.config_logs_block`, `aws_s3_bucket_server_side_encryption_configuration.config_logs_encryption`, and `aws_s3_bucket_versioning.config_logs_versioning`. A summary line states: "Apply complete! Resources: 5 added, 0 changed, 0 destroyed." The terminal also shows the current directory and the status of the resources.

## 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

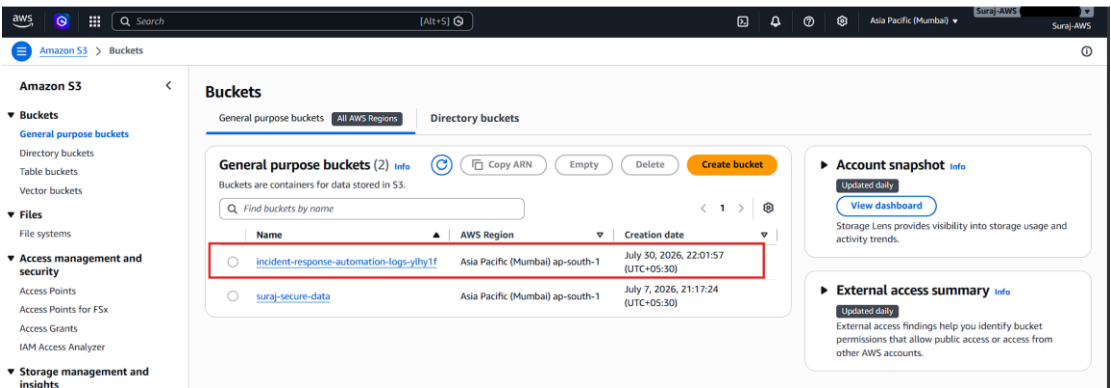


Figure 6: Amazon S3 Bucket Overview

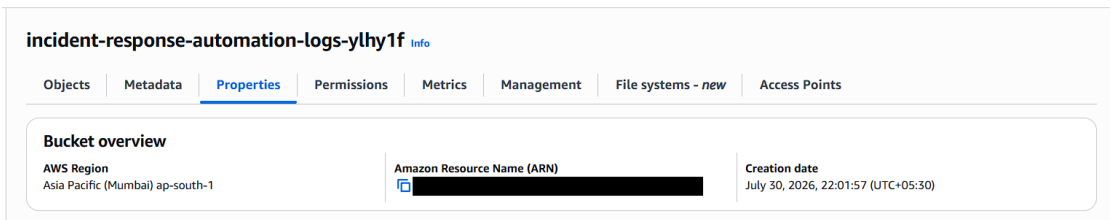


Figure 7: Amazon S3 Bucket Versioning Enabled

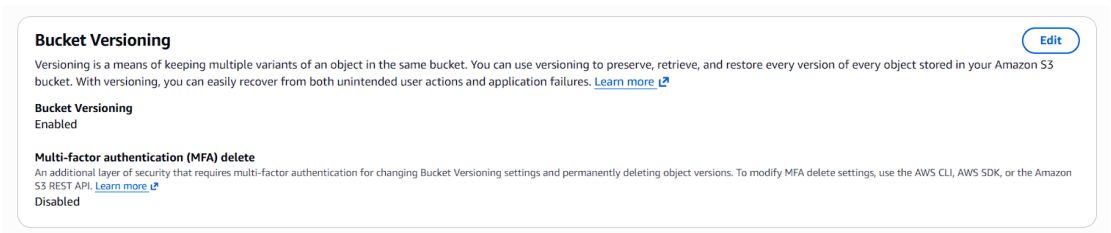


Figure 8: Amazon S3 Default Encryption Enabled

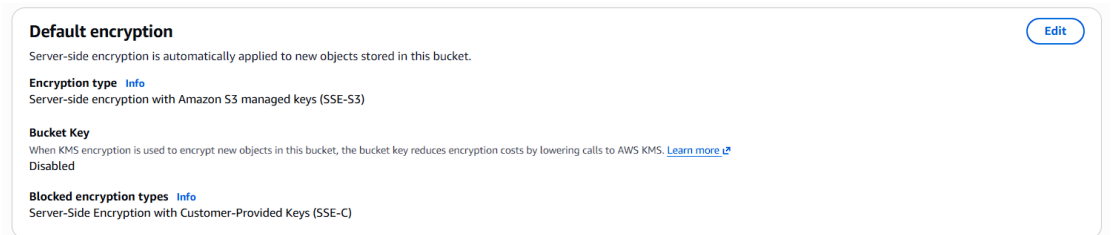


Figure 9: Amazon S3 Resource Tags Configuration

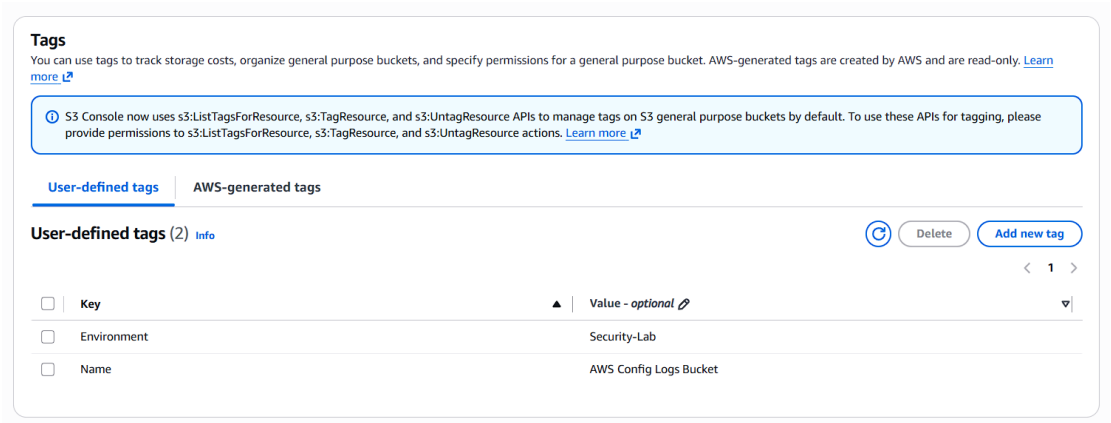
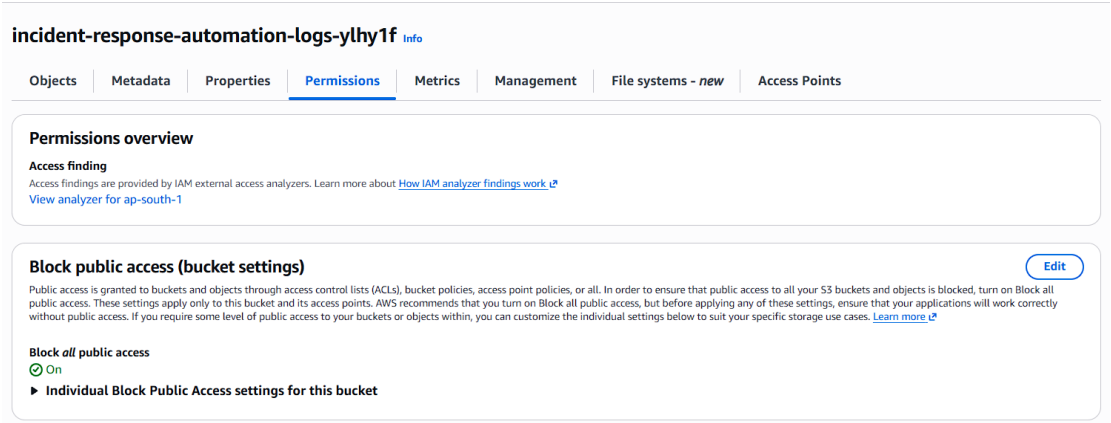


Figure 10: Amazon S3 Block Public Access Enabled



### 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

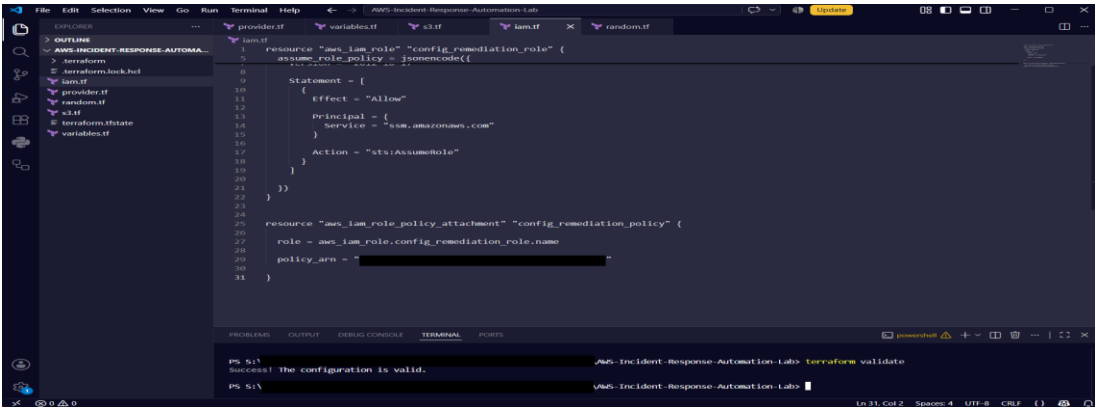
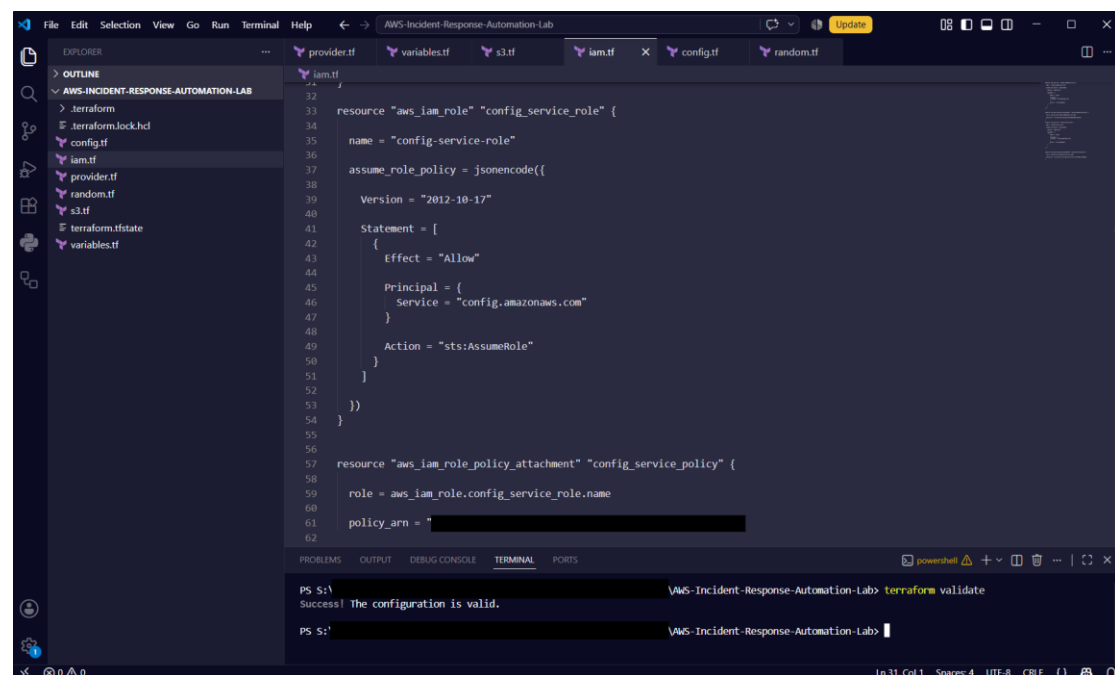


Figure 12: AWS Config Recorder and Delivery Channel Validation



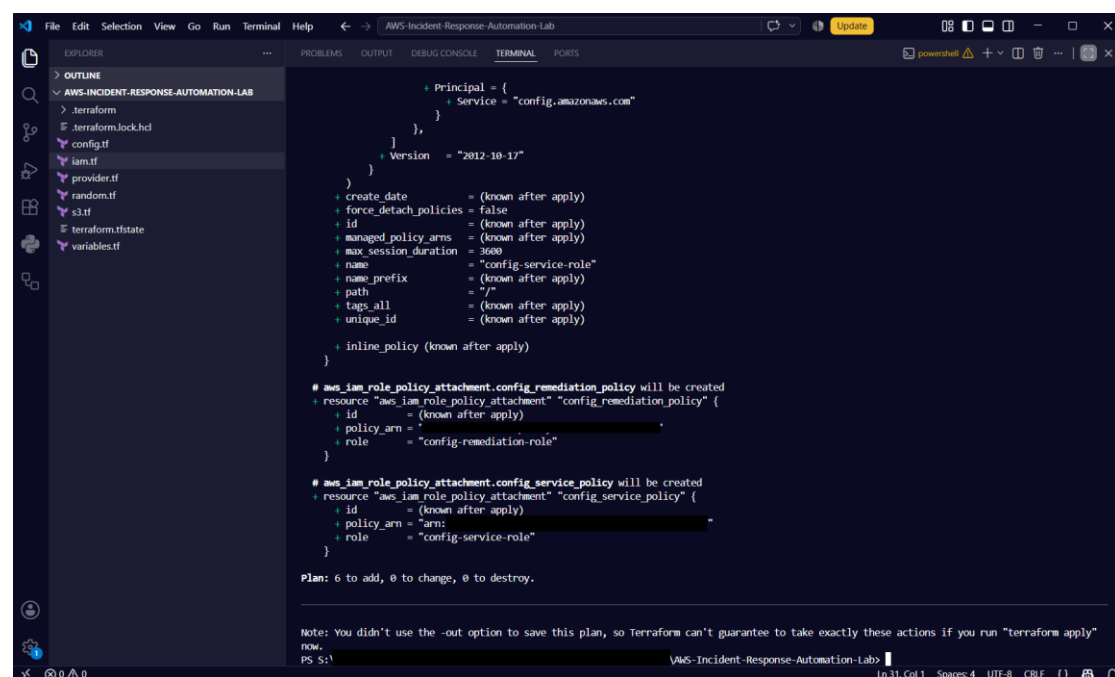
The screenshot shows the Visual Studio Code interface with the Terraform configuration file `iam.tf` open. The configuration defines an `aws_iam_role` resource named `config_service_role` and an `aws_iam_role_policy_attachment` resource named `config_service_policy`. The terminal at the bottom shows the command `terraform validate` being executed, resulting in a success message: "Success! The configuration is valid."

```
resource "aws_iam_role" "config_service_role" {
  name = "config-service-role"
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Effect = "Allow"
        Principal = {
          Service = "config.amazonaws.com"
        }
        Action = "sts:AssumeRole"
      }
    ]
  })
}

resource "aws_iam_role_policy_attachment" "config_service_policy" {
  role = aws_iam_role.config_service_role.name
  policy_arn = "arn:aws:iam::aws:policy/config/ConfigServiceRolePolicy"
}
```

```
PS S:\AWS-Incident-Response-Automation-Lab> terraform validate
Success! The configuration is valid.
PS S:\AWS-Incident-Response-Automation-Lab>
```

Figure 13: Final Terraform Plan Before Deployment



The screenshot shows the Visual Studio Code interface with the Terraform plan output displayed in the terminal. The plan shows that the `aws_iam_role.config_service_role` and `aws_iam_role_policy_attachment.config_service_policy` resources will be created. The plan also shows the attributes of these resources, such as `name`, `path`, `tags_all`, and `unique_id`.

```
+ Principal = {
+   Service = "config.amazonaws.com"
+ }
+ Version = "2012-10-17"
+ create_date = (known after apply)
+ force_detach_policies = false
+ id = (known after apply)
+ managed_policy_arns = (known after apply)
+ max_session_duration = 3600
+ name = "config-service-role"
+ name_prefix = (known after apply)
+ path = "/"
+ tags_all = (known after apply)
+ unique_id = (known after apply)
+ inline_policy (known after apply)
}

# aws_iam_role_policy_attachment.config_remediation_policy will be created
+ resource "aws_iam_role_policy_attachment" "config_remediation_policy" {
+   id = (known after apply)
+   policy_arn = "arn:aws:iam::aws:policy/config/ConfigRemediationRolePolicy"
+   role = "config-remediation-role"
+ }

# aws_iam_role_policy_attachment.config_service_policy will be created
+ resource "aws_iam_role_policy_attachment" "config_service_policy" {
+   id = (known after apply)
+   policy_arn = "arn:aws:iam::aws:policy/config/ConfigServiceRolePolicy"
+   role = "config-service-role"
+ }

Plan: 6 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
PS S:\AWS-Incident-Response-Automation-Lab>
```

Figure 14: Terraform Apply Completed Successfully

```

+ Principal = {
+   Service = "config.amazonaws.com"
+ }
+ Resource = "arn:aws:config:us-east-1:123456789012:config/recorder"
+ Sid = "AWSConfigBucketPermissionsCheck"
+ },
+ {
+   Action = "s3:PutObject"
+   Condition = {
+     StringEquals = {
+       "s3:x-amz-acl" = "bucket-owner-full-control"
+     }
+   }
+   Effect = "Allow"
+   Principal = {
+     Service = "config.amazonaws.com"
+   }
+   Resource = "arn:aws:s3:::incident-response-automation-logs-yihy1f"
+   Sid = "AWSConfigBucketDelivery"
+ },
+ Version = "2012-10-17"
+ }
+ }
+ }

Plan: 3 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
Terraform will perform the actions described above.
Only 'yes' will be accepted to approve.

Enter a value: yes

aws_s3_bucket_policy.config_bucket_policy: Creating...
aws_s3_bucket_policy.config_bucket_policy: Creation complete after 0s [id=incident-response-automation-logs-yihy1f]
aws_config_delivery_channel.incident_response_delivery_channel: Creating...
aws_config_delivery_channel.incident_response_delivery_channel: Creation complete after 1s [id=incident-response-delivery-channel]
aws_config_configuration_recorder_status.incident_response_recorder_status: Creating...
aws_config_configuration_recorder_status.incident_response_recorder_status: Creation complete after 0s [id=incident-response-recorder]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
PS S:\
AWS-incident-response-automation-lab

```

## 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

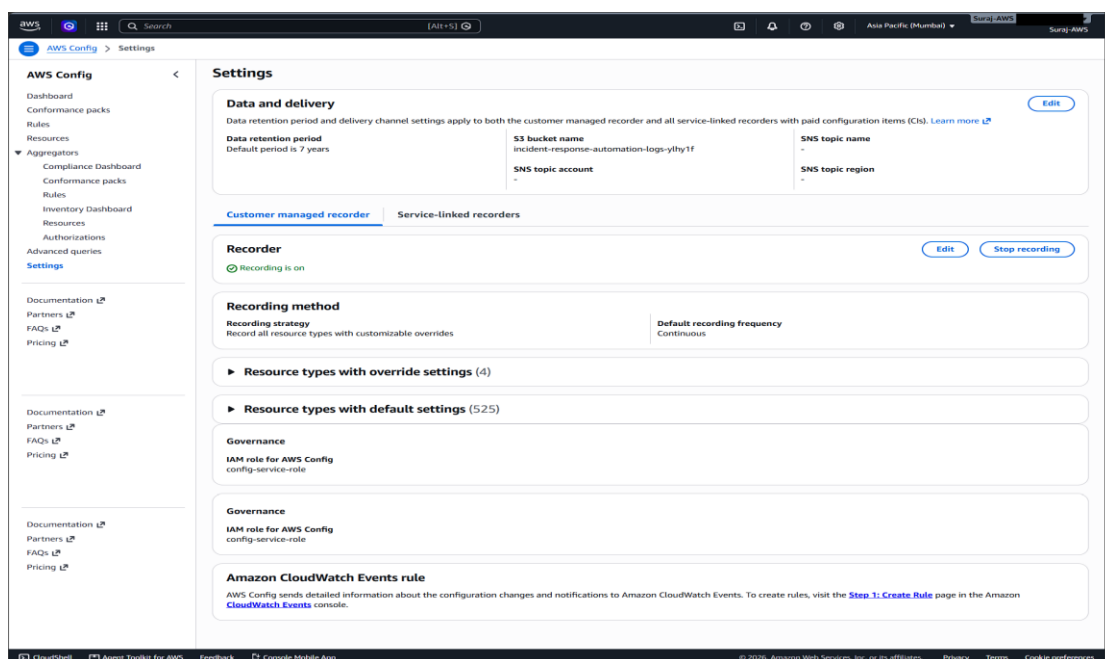


Figure 16: AWS Config Delivery Channel Configured

aws [Search] [Alt+S]

AWS Config > Settings > Edit data and delivery channel settings

### Edit data and delivery channel settings

#### Data retention period

Data retention period

☒ Retain AWS Config data for 7 years (2557 days)

☐ Set a custom retention period for configuration items recorded by AWS Config.

#### Delivery channel

**Amazon S3 bucket**

☐ Create a bucket ☒ Choose a bucket from your account ☐ Choose a bucket from another account

Ensure appropriate permissions are available in this S3 bucket's policy. [Learn more](#)

**S3 Bucket name (required)**

incident-response-automation-logs-yl...

/AWSLogs/[redacted]Config/ap-south-1

**Amazon SNS topic**

☐ Stream configuration changes and notifications to an Amazon SNS topic.  
If you choose email as the notification endpoint for your SNS topic, this can cause a high volume of email. [Learn more](#)

Cancel Save

Figure 17: AWS Config Logs Delivered to Amazon S3

aws [Search] [Alt+S]

Amazon S3 > Buckets > incident-response-automation-logs-y1fy1f > AWSLogs/[redacted] > Config/ > ConfigWritabilityCheckFile

### ConfigWritabilityCheckFile

Copy S3 URI Download Open Object actions

Properties Permissions Versions

#### Object overview

**Owner**  
[redacted]

**AWS Region**  
Asia Pacific (Mumbai) ap-south-1

**Last modified**  
July 31, 2026, 14:38:10 (UTC+05:30)

**Size**  
0 B

**Type**  
[redacted]

**Key**  
AWSLogs/[redacted]Config/ConfigWritabilityCheckFile

**S3 URI**  
s3://incident-response-automation-logs-y1fy1f/AWSLogs/[redacted]Config/ConfigWritabilityCheckFile

**Amazon Resource Name (ARN)**  
arn:aws:s3:::incident-response-automation-logs-y1fy1f/AWSLogs/[redacted]Config/ConfigWritabilityCheckFile

**Entity tag (ETag)**  
d41d8cd98f00b204e9800998ecf8427e

**Object URL**  
[https://incident-response-automation-logs-y1fy1f.s3.ap-south-1.amazonaws.com/AWSLogs/\[redacted\]Config/ConfigWritabilityCheckFile](https://incident-response-automation-logs-y1fy1f.s3.ap-south-1.amazonaws.com/AWSLogs/[redacted]Config/ConfigWritabilityCheckFile)

Figure 18: AWS Config Rule Review Before Creation

AWS Config > Rules > Add rule

Step 1: Specify rule type  
Step 2: Configure rule  
Step 3: Review and create

### Review and create

Review this rule before adding it to your account

#### Details

**Rule name**  
s3-bucket-public-read-prohibited

**Description**  
Checks that your Amazon S3 buckets do not allow public read access. The rule checks the Block Public Access settings, the bucket policy, and the bucket access control list (ACL).

**Managed rule name**  
S3\_BUCKET\_PUBLIC\_READ\_PROHIBITED

#### Evaluation mode

**Proactive evaluation**  
Disabled

**Trigger type**  
• When configuration changes  
• Periodic

**Scope of changes**  
Resources

**Detective evaluation**  
Enabled

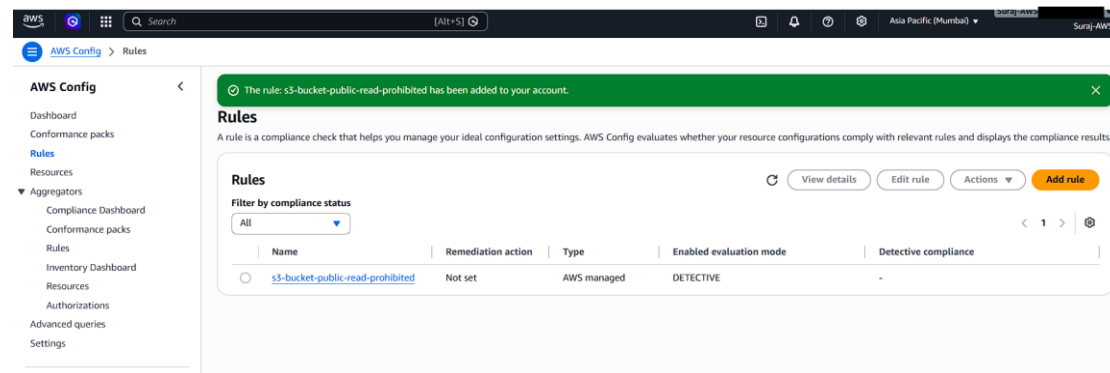
**Frequency**  
24 hours

**Resource types**  
S3 Bucket

**Resource identifier**  
-

Cancel Previous Save

Figure 19: AWS Config Rule Created Successfully



## Phase 5: Automated Remediation Implementation

**Implementation Performed:** AWS Config Remediation used `AWSSConfigRemediation-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

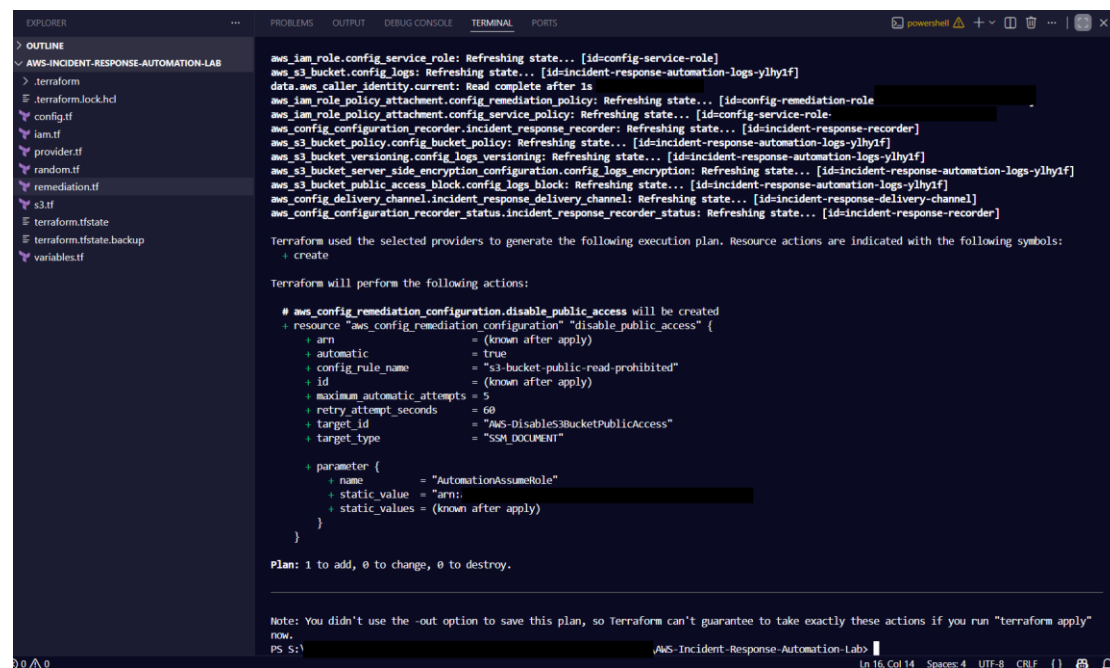


Figure 21: Final Terraform Plan for Automatic Remediation

```
+ retry_attempt_seconds = 60
+ target_id              = "AWSConfigRemediation-Configures3BucketPublicAccessBlock"
+ target_type            = "SSM_DOCUMENT"

+ parameter {
+   name      = "BucketName"
+   static_value = "incident-response-automation-logs-ylhy1f"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "RestrictPublicBuckets"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "BlockPublicAcls"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "IgnorePublicAcls"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "BlockPublicPolicy"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "AutomationAssumeRole"
+   static_value = "arn:"
+   static_values = (known after apply)
+ }
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.

PS S:\ AWS-Incident-Response-Automation-Lab

Figure 22: Automatic Remediation Successfully Deployed

```
+ static_value = "incident-response-automation-logs-ylhy1f"
+ static_values = (known after apply)
+ }

+ parameter {
+   name      = "RestrictPublicBuckets"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "BlockPublicAcls"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "IgnorePublicAcls"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "BlockPublicPolicy"
+   static_value = "true"
+   static_values = (known after apply)
+ }

+ parameter {
+   name      = "AutomationAssumeRole"
+   static_value = "arn:"
+   static_values = (known after apply)
+ }
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

aws\_config\_remediation\_configuration.disable\_public\_access: Creating...

aws\_config\_remediation\_configuration.disable\_public\_access: Creation complete after 1s [id=s3-bucket-public-read-prohibited]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

PS S:\ AWS-Incident-Response-Automation-Lab

Figure 23: AWS Config Rule with Remediation Configuration

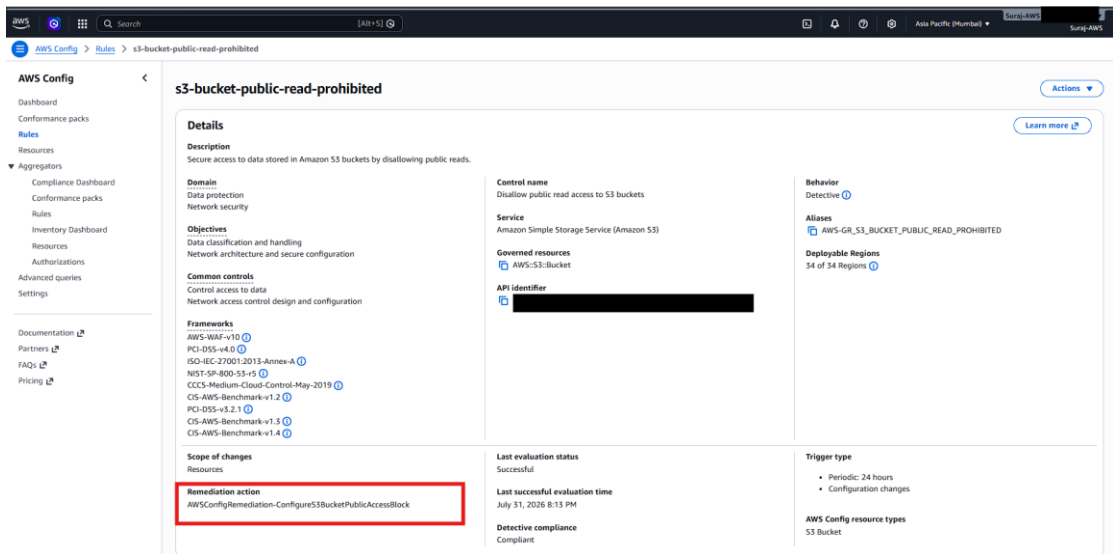
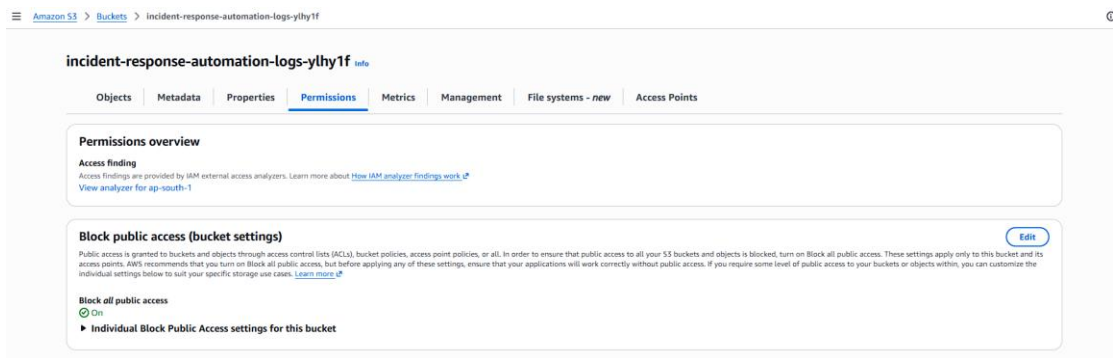


Figure 24: Initial Secure State of S3 Bucket



## 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

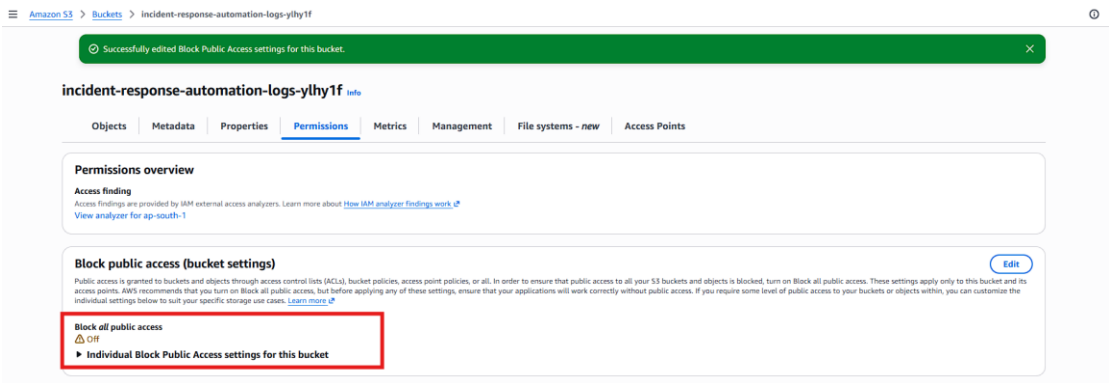
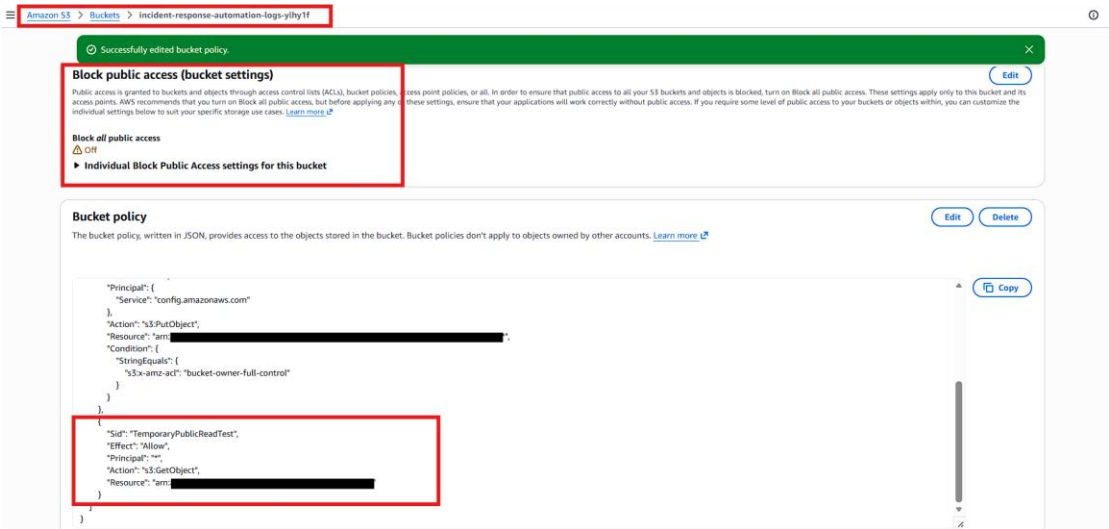


Figure 26: Temporary Public Read Bucket Policy Created



## 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

| Resources in scope    |                                          |           |                          |            |            |  |
|-----------------------|------------------------------------------|-----------|--------------------------|------------|------------|--|
| All                   |                                          |           |                          |            |            |  |
|                       | ID                                       | Type      | Status                   | Annotation | Compliance |  |
| <input type="radio"/> | incident-response-automation-logs-ylyh1f | S3 Bucket | Action execution queued. | -          | Compliant  |  |
| <input type="radio"/> | suraj-secure-data                        | S3 Bucket | -                        | -          | Compliant  |  |

Figure 28: Automatic Remediation Executed Successfully

| ID                                       | Type      | Status                       | Annotation | Compliance |
|------------------------------------------|-----------|------------------------------|------------|------------|
| incident-response-automation-logs-ylyh1f | S3 Bucket | Action executed successfully | -          | Compliant  |
| suraj-secure-data                        | S3 Bucket | -                            | -          | Compliant  |

Figure 29: Amazon S3 Block Public Access Restored Automatically

**Permissions overview**

**Block public access (bucket settings)**

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

**Block all public access**

On

► Individual Block Public Access settings for this bucket

**Bucket policy**

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Public access is blocked because Block Public Access settings are turned on for this bucket  
To determine which settings are turned on, check your Block Public Access settings for this bucket. [Learn more about using Amazon S3 Block Public Access](#)

```
{
  "Principal": {
    "Service": "config.amazonaws.com"
  },
  "Action": "s3:PutObject",
  "Resource": "arn:aws:s3:::[redacted]",
  "Condition": {
    "StringEquals": {
      "s3:x-amz-acl": "bucket-owner-full-control"
    }
  },
  "Effect": "Allow",
  "Principal": "*",
  "Action": "s3:GetObject",
  "Resource": "[redacted]"
}
```

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

**Timeline**

The resource timeline allows you to view all the configuration items captured over time for a specific resource and the compliance status changes. For accurate reporting on the compliance status, you must record the AWS Config ResourceCompliance resource type.

**General details**

Resource ID: incident-response-automation-logs-ylyh1f | Resource type: AWS::S3::Bucket | Resource name: incident-response-automation-logs-ylyh1f

**Events**

All times are in Asia/Calcutta (UTC+05:30)

End date: 2026/08/01 | Event type: Compliance events

**August 1, 2026**

16:06:10 Rule compliance All compliant 1 rule(s) applied

Rule: s3-bucket-public-read-prohibited | Compliance: Compliant

16:04:05 Rule compliance 1 Noncompliant rule(s) 1 rule(s) applied

Rule: s3-bucket-public-read-prohibited | Compliance: Noncompliant

## 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

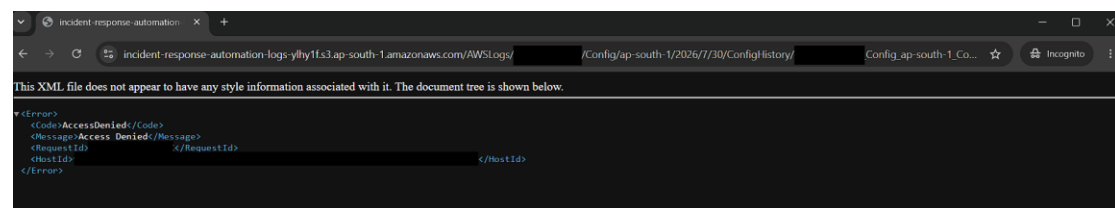
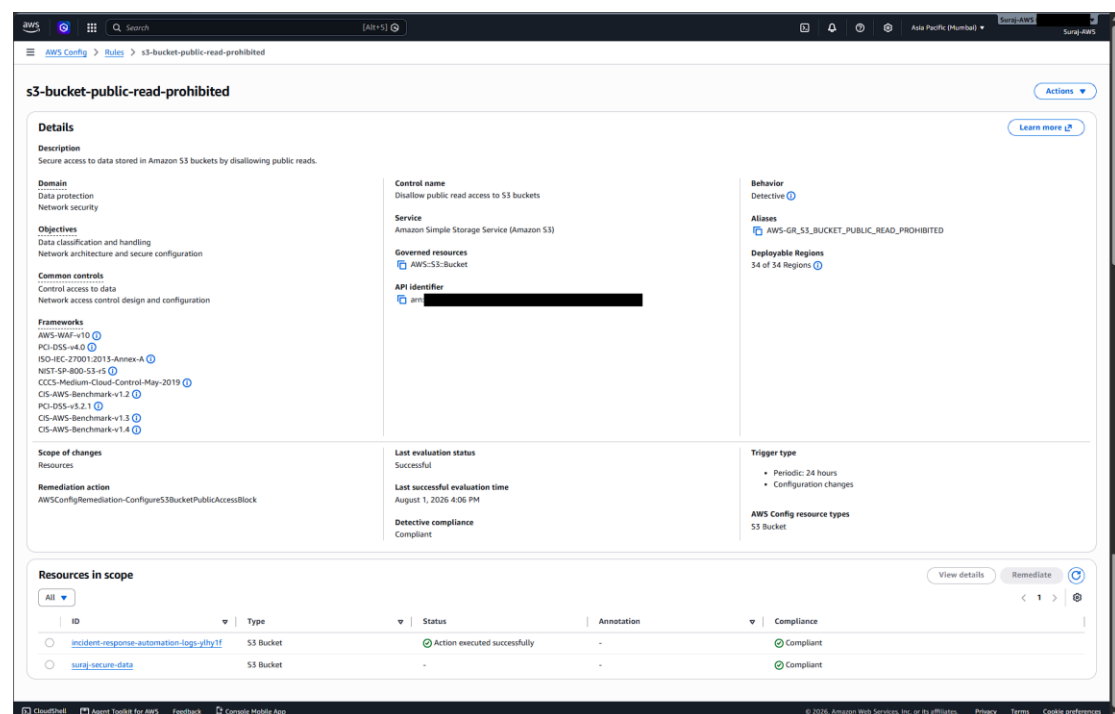


Figure 32: Final AWS Config Compliance Status Verified



## 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.