

Data Protection and Encryption Management on AWS

1. Problem Statement

A financial services organization handling sensitive financial records required consistent encryption governance across storage, databases, and logs. Inconsistent encryption configurations created regulatory risk.

I implemented this framework to ensure that sensitive data stored in S3, EBS, and RDS is protected using strong encryption, centralized key management, and continuous compliance monitoring.

2. Objectives

The primary objectives of this implementation were to:

- Prevent public exposure of sensitive data
- Enforce encryption at rest across all storage services
- Centralize key management using AWS KMS
- Enable automatic and manual key rotation
- Maintain continuous compliance visibility
- Reduce the risk of data leakage

3. S3 Data Protection

3.1 Account-Level Public Access Controls

I enabled S3 Block Public Access at the account level to prevent accidental public exposure of any bucket.

The screenshot shows the AWS S3 console under 'Account and organization settings'. On the left sidebar, there are sections for Buckets, Access management and security, Storage management and insights, and Account and organization settings. Under 'Access management and security', the 'Block all public access' option is selected ('On'). A green success message at the top right says 'Block Public Access settings for this account successfully updated.' Below this, there's a section for 'AWS Organizations settings for Storage Lens'.

This ensured that all existing and newly created buckets were protected by default.

3.2 S3 Encryption

All S3 buckets were encrypted using AWS SSE-KMS with customer-managed keys.

The screenshot shows the AWS S3 console under 'Buckets' for the 'aws-cloudtrail-logs-securephere' bucket. The left sidebar includes sections for Buckets, Access management and security, Storage management and insights, and Account and organization settings. In the main area, under 'Default encryption', it shows 'Encryption type: Info' (Server-side encryption with AWS Key Management Service keys (SSE-KMS)) and 'Encryption key ARN' (arn:aws:kms:us-east-1:1879381257906:key/f9bc03c2-6857-4950-b1a0-c2f22628e590). A green success message at the top right says 'Successfully edited default encryption. Objects uploaded, modified, or copied into this bucket will inherit this encryption configuration unless otherwise specified.' There is also a note about an upcoming change to default encryption in April 2026.

This provided full control over encryption policies, auditing, and key lifecycle management.

4. EBS Encryption Management

4.1 Default Encryption

EBS encryption by default was enabled to ensure that all new volumes and snapshots are automatically encrypted.

The screenshot shows the AWS EC2 Settings page under the 'Security Manager' menu. The left sidebar includes sections for Images, Elastic Block Store, Network & Security, Load Balancing, and Auto Scaling. The main content area displays the 'EBS encryption' settings, which are currently enabled. It also shows the 'Default encryption key' set to 'arn:aws:kms:us-east-1:879381257906:key/17d94a5b-c5bb-420c-acfb-7c6db3c12a42'. Below this, there are sections for 'Block public access for AMIs' and 'Block public access for EBS snapshots', both of which have 'Public access' set to 'New public sharing blocked'. A large watermark reading 'ANSWER DAY' is overlaid across the page.

4.2 Migrating Unencrypted Volumes

An unencrypted EBS volume was identified on an existing EC2 instance.

To remediate this:

- A snapshot was created from the unencrypted volume
- The snapshot was copied and encrypted
- A new encrypted volume was created
- The instance was stopped
- The encrypted volume was attached

This process ensured encryption without data loss.

5. RDS Encryption Enforcement

All RDS databases were reviewed to confirm encryption at rest.

Encryption was enforced at creation time, as RDS encryption cannot be enabled after deployment.

Operational guidance was provided to ensure future databases follow this standard.

6. Key Management with AWS KMS

6.1 Customer-Managed Keys

A customer-managed symmetric KMS key was created for workloads.

This key was used to encrypt:

- S3 buckets
- EBS volumes
- RDS databases

6.2 Automatic Key Rotation

Automatic key rotation was enabled with a 365-day cycle.

The screenshot shows the AWS KMS console interface for managing keys. On the left, there's a navigation sidebar with 'Key Management Service (KMS)' selected. Under 'Customer-managed keys', 'securedKMSKEY' is listed. The main panel shows the 'General configuration' of the key, including its ARN, Alias, Status (Enabled), Description, Creation date (Dec 21, 2025), and Regionality (Single region). Below this, the 'Key material and rotations' tab is active, showing the 'Automatic key rotation' section. It indicates that AWS KMS automatically rotates the key based on a 365-day rotation period, with the next rotation date set for Dec 21, 2026. There's also an 'On-demand key rotation' section with a 'Rotate now' button.

This reduced the risk associated with long-lived cryptographic keys.

6.3 Manual Key Rotation Process

For keys that do not support automatic rotation, a documented manual rotation process was implemented:

Step 1: Identify the active key

Step 2: Create a new customer-managed key

Step 3: Update services to use the new key

Step 4: Monitor system behavior

Step 5: Disable the old key

Step 6: Retain the old key for recovery and compliance

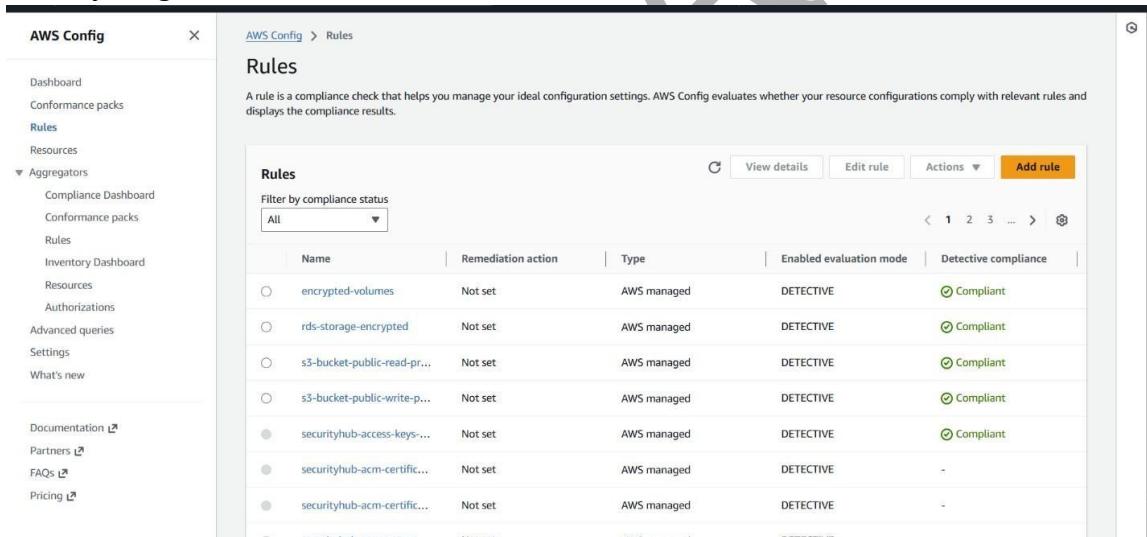
This process ensured secure rotation without service disruption.

7. Compliance Monitoring and Alerting

7.1 AWS Config Rules

AWS Config rules were created to monitor:

- S3 public access settings
- Storage encryption status
- KMS key usage



The screenshot shows the AWS Config Rules interface. On the left is a navigation sidebar with links like Dashboard, Conformance packs, Rules, Resources, Aggregators, Documentation, Partners, FAQs, and Pricing. The main content area has a header "AWS Config > Rules". Below the header is a descriptive text: "A rule is a compliance check that helps you manage your ideal configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and displays the compliance results." A large "Add rule" button is at the top right. The main table lists seven rules, each with columns for Name, Remediation action, Type, Enabled evaluation mode, and Detective compliance status (all marked as Compliant). The table includes a "Filter by compliance status" dropdown set to "All" and a pagination bar showing page 1 of 1.

Name	Remediation action	Type	Enabled evaluation mode	Detective compliance
encrypted-volumes	Not set	AWS managed	DETECTIVE	Compliant
rds-storage-encrypted	Not set	AWS managed	DETECTIVE	Compliant
s3-bucket-public-read-pr...	Not set	AWS managed	DETECTIVE	Compliant
s3-bucket-public-write-p...	Not set	AWS managed	DETECTIVE	Compliant
securityhub-access-keys-...	Not set	AWS managed	DETECTIVE	Compliant
securityhub-acm-certific...	Not set	AWS managed	DETECTIVE	-
securityhub-acm-certific...	Not set	AWS managed	DETECTIVE	-

These rules provided real-time compliance visibility.

7.2 Alerting with SNS

Amazon SNS topics were configured to deliver alerts when non-compliant resources were detected.

7.3 Validation Testing

Public access settings were temporarily modified to validate alert delivery.

Successful notifications confirmed correct configuration.

The screenshot shows a Gmail inbox with 221 messages. Four messages from 'AWS Notifications' are visible, all sent 33 minutes ago at 8:47 PM. The messages are identical, detailing an AWS API Call via CloudTrail with ID 99217fc-6084-7d6f-d39e-bb8dbe532e4, source 'aws.config', account '879381257906', and time '2020-01-06T20:53:06Z'. The fourth message includes a large amount of JSON data describing the configuration item change.

Compose

Inbox 221

AWS Notifications [version="0", "id": "99217fc-6084-7d6f-d39e-bb8dbe532e4", "detail-type": "AWS API Call via CloudTrail", "source": "aws.config", "account": "879381257906", "time": "2020-01-06T20:53:06Z"]

AWS Notifications [version="0", "id": "0e30f65e-17ff-d80d-81e3-833782c172da", "detail-type": "AWS API Call via CloudTrail", "source": "aws.config", "account": "879381257906", "time": "2020-01-06T20:53:06Z"]

AWS Notifications [version="0", "id": "8a6b9e59-51a3-6c12-0c1d-8ad6535b5bd3", "detail-type": "AWS API Call via CloudTrail", "source": "aws.config", "account": "879381257906", "time": "2020-01-06T20:53:06Z"]

AWS Notifications [version="0", "id": "502679f4-0094-8db1-070d-fa3849e6527", "detail-type": "Config Configuration Item Change", "source": "aws.config", "account": "879381257906", "time": "2020-01-06T20:53:06Z", "region": "us-east-1", "resources": []} ...

Labels +

AWS Notifications to me

8:47PM (33 minutes ago) 8:47PM (33 minutes ago) 8:48PM (32 minutes ago) 8:53PM (27 minutes ago)

Compose Reply Forward

8. Governance and Documentation

All encryption policies, key management procedures, and compliance controls were documented.

This supported:

- Security audits
- Regulatory compliance
- Incident investigations
- Operational continuity

9. Outcomes and Impact

This implementation delivered the following results:

- Eliminated public S3 exposure risk
- Enforced encryption across storage services
- Centralized cryptographic key management

- Reduced operational risk
- Improved audit readiness
- Enabled real-time compliance monitoring

10. Conclusion

I designed and implemented a secure, compliant, and centrally managed data protection framework on AWS.

Through strong encryption, effective key management, and continuous monitoring, this solution protects sensitive data and supports regulatory requirements.

ADDEDAYO