



Topic: Automating Compliance Audits with Cloud Security Tools

Table of contents:

Project Summary.....	2
Project Scope.....	2
Implementation.....	3
Set Up Your AWS Environment.....	3
Create a new Group User	7
Review and Create User	9
Download Credentials	10
Configure AWS CLI.....	11
Create Managed rules.....	13
Setup AWS Config.....	14
Create AWS Config Rules.....	17
Enable AWS Security Hub	18
Create a Lambda Function.....	19
Line-by-Line Explanation of the Function:.....	21
Deploy the Function.....	26

Set Up AWS CloudWatch Events (EventBridge).....	27
Navigate to EventBridge:.....	27
Create a New Rule.....	27
Define the Event Source.....	28
Add Event Pattern.....	29
Configuration of the Event pattern:.....	30
Select Target:.....	31
Create Rule.....	31
Test the Setup.....	34
Trigger a Compliance Violation.....	34
Monitor the Event and Lambda Function.....	37
Verify Lambda Execution.....	38
Review and Validate Compliance Reports.....	41

Project Summary

- The primary goal of this project is to enhance and streamline the compliance audit process by implementing automated solutions using advanced cloud security tools. The rapid growth of cloud adoption necessitates robust, scalable, and efficient mechanisms to maintain compliance with ever-evolving regulatory standards such as GDPR, HIPAA, and PCI-DSS.
- Traditional manual audit processes are time-consuming, prone to errors, and often unable to keep pace with the dynamic nature of cloud environments. By leveraging automated cloud security tools, this project aims to transform the compliance landscape.
- Automated tools will continuously monitor cloud infrastructure, detect non-compliance in real time, and provide actionable insights, thereby reducing the risk of regulatory breaches and associated penalties.
- These tools will be configured to conduct routine compliance checks, generate comprehensive audit reports, and integrate seamlessly with existing Security Information and Event Management (SIEM) systems. This ensures that compliance is not a periodic activity but a continuous process embedded within the organizational workflow.
- Additionally, the project will foster a culture of proactive security, where compliance is maintained not just to meet regulatory requirements but to enhance overall security posture. By automating the compliance audit process, organizations can achieve greater operational efficiency, reduce manual workload, and ensure that compliance standards are consistently met across all cloud environments.

Project Scope

The project encompasses developing, deploying, and integrating automated auditing systems across various cloud environments. Key activities include:

1. Assessment and Selection of Tools:

- Usage of existing cloud security tools suitable for compliance audits.
Here we have used Amazon Web Service.

2. Implementation:

- Configure and deploy chosen tools across cloud environments.
- Develop custom scripts and policies for compliance checks.
- Integrate automated systems with existing security information and event management (SIEM) solutions.

3. Automation and Continuous Monitoring:

- Automate routine compliance checks and reporting.
- Establish continuous monitoring to detect and alert on compliance deviations in real-time.
- Implement periodic reviews and updates of compliance policies to adapt to regulatory changes.

4. Training and Documentation:

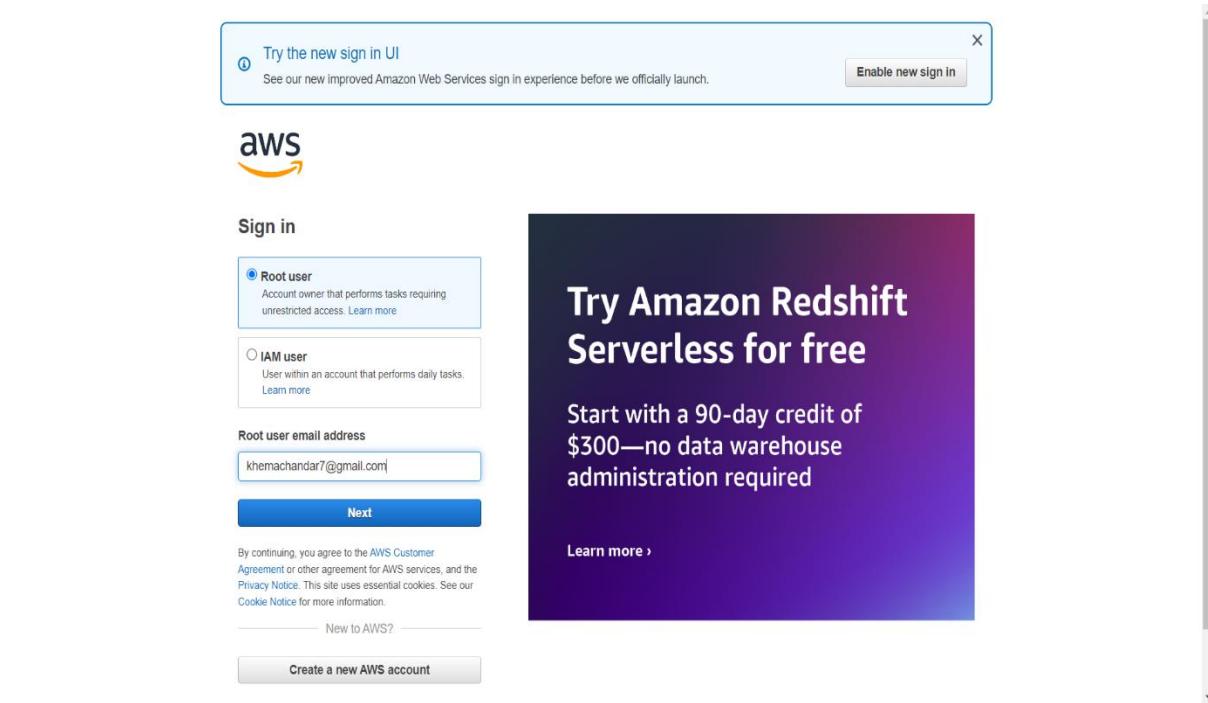
- Train IT and security teams on using the new automated compliance tools.
- Develop comprehensive documentation detailing the tools' setup, usage, and maintenance.

Implementation:

Set Up Your AWS Environment

Create an AWS Account

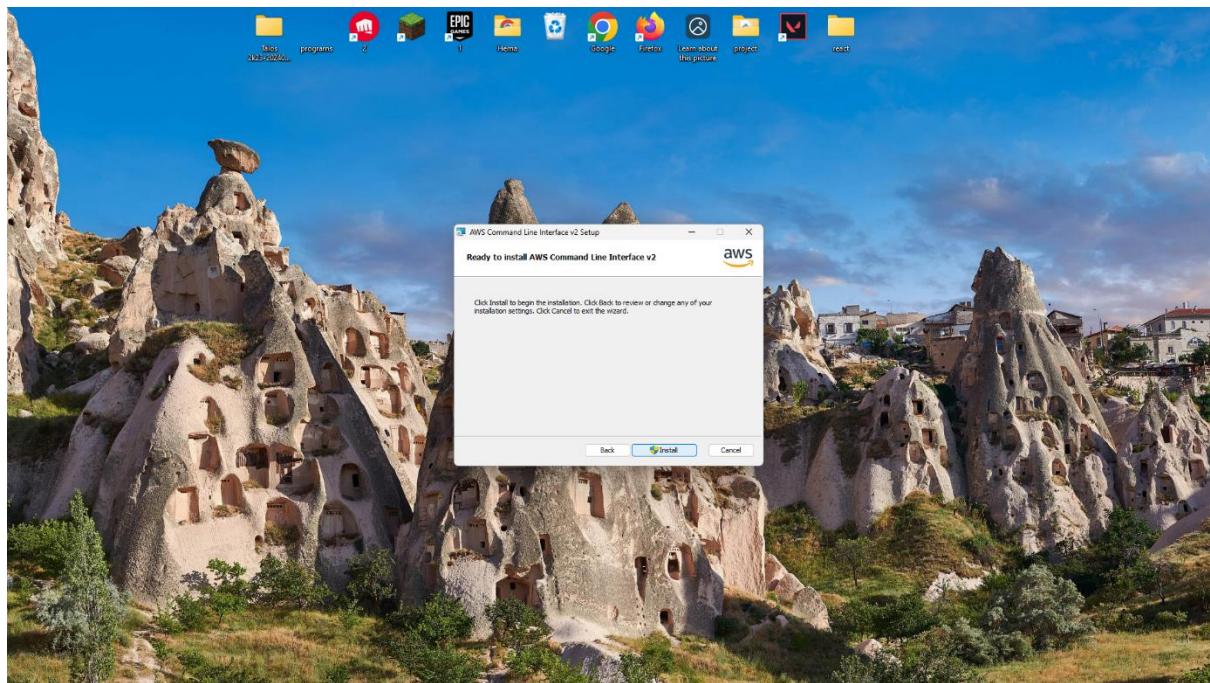
1. Go to the AWS Management Console.
2. Sign up for an account if you don't already have one.
3. Complete the sign-up process, including providing billing information.

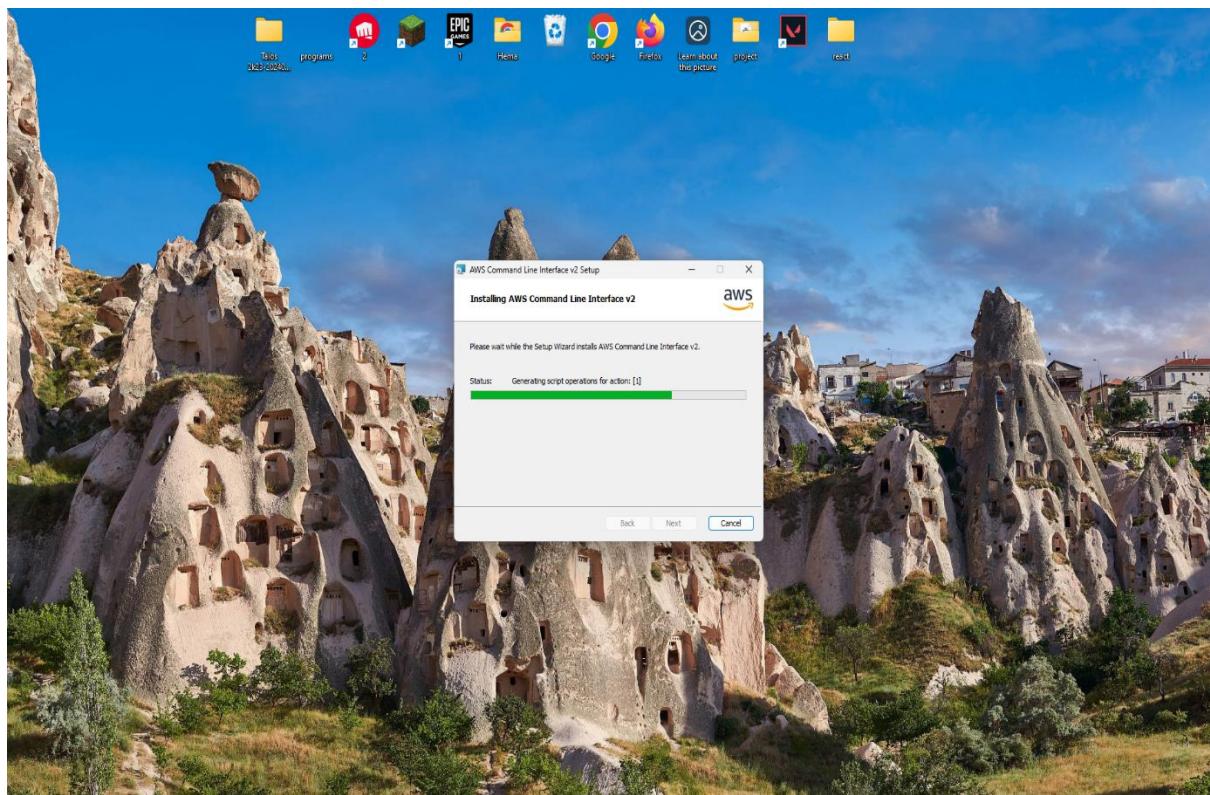


Install AWS CLI

1. Download and install the AWS CLI from the official AWS CLI page.
2. Go to the AWS CLI installation page and download the Windows installer.
3. Open the downloaded .msi file and follow the on-screen instructions to complete the installation.

Screenshot of the AWS Command Line Interface User Guide for Version 2. The page shows the 'Install or update the AWS CLI' section for Windows. It includes instructions for downloading the MSI installer from <https://awscli.amazonaws.com/AWSCLIV2.msi> and running it with msieexec.exe. A callout box for 'Introducing Amazon Q' is visible.



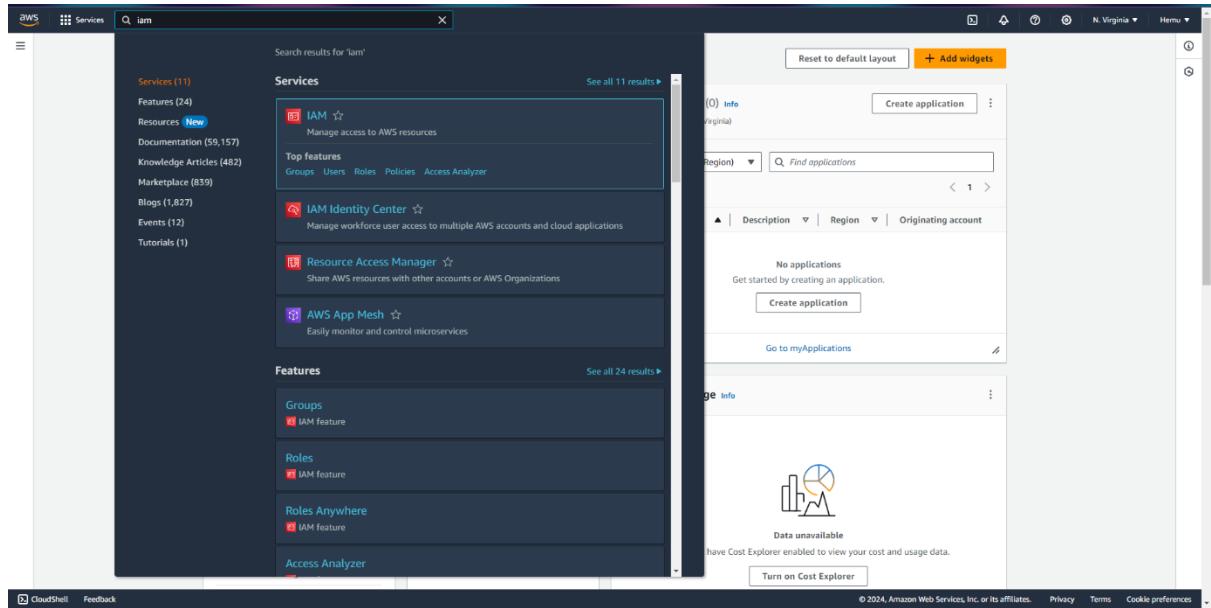


1. Open the Command Prompt and run the following command to verify the installation:

```
Command Prompt
Microsoft Windows [Version 10.0.22621.3958]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91814>aws --version
aws-cli/2.17.19 Python/3.11.9 Windows/10 exe/AMD64
C:\Users\91814>
```

2. Navigate to IAM (Identity and Access Management):



Create a New Group User:

- In the IAM dashboard, click on "User Groups" from the left-hand menu, and then click the "Create Group" button.

The screenshot shows the AWS IAM User groups page. A green banner at the top says 'User group deleted.' Below it, the 'User groups (0)' section is displayed with a table header: 'Group name' (sorted by creation time), 'Users', and 'Permissions'. A note below the table says 'No resources to display'. On the right, there are 'Create group' and 'Delete' buttons. The left sidebar includes sections for Dashboard, Access management (User groups, Roles, Policies, Identity providers, Account settings), Access reports (Access Analyzer, External access, Unused access, Analyzer settings, Credential report, Organization activity, Service control policies), and Related consoles (IAM Identity Center, AWS Organizations).

1. Set Group Details:

The screenshot shows the 'Create user group' page. It has three main sections: 'Name the group' (with a 'User group name' input field containing 'project1'), 'Add users to the group - Optional (0)', and 'Attach permissions policies - Optional (1/944)'. The 'Attach permissions policies' section lists three policies: 'AdministratorAccess' (selected), 'AdministratorAccess-Amplify', and 'AdministratorAccess-AWSCElastiCacheForRedis'. The bottom right corner shows copyright information: '© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences'.

2. Select the Administrator Access:

The screenshot shows the AWS IAM User Groups page. On the left, there's a sidebar with navigation links like Dashboard, Access management (User groups, Roles, Policies, Identity providers, Account settings), Access reports (Access Analyzer, External access, Unused access, Analyzer settings), Credential report, Organization activity, Service control policies, and Related consoles (CloudShell, Feedback). The main content area displays a table of managed policies:

	Policy Name	Description	Condition	Actions
<input checked="" type="checkbox"/>	AdministratorAccess	AWS managed - job function	None	Provides full access to AWS services an...
<input type="checkbox"/>	AdministratorAccess-Amplify	AWS managed	None	Grants account administrative permis...
<input type="checkbox"/>	AdministratorAccess-AWSElasticBeanstalk	AWS managed	None	Grants account administrative permis...
<input type="checkbox"/>	AlexaForBusinessDeviceSetup	AWS managed	None	Provide device setup access to AlexaFo...
<input type="checkbox"/>	AlexaForBusinessFullAccess	AWS managed	None	Grants full access to AlexaForBusiness ...
<input type="checkbox"/>	AlexaForBusinessGatewayExecution	AWS managed	None	Provide gateway execution access to A...
<input type="checkbox"/>	AlexaForBusinessLifesizeDelegatedAccess...	AWS managed	None	Provide access to Lifesize AVS devices
<input type="checkbox"/>	AlexaForBusinessPolyDelegatedAccessPolicy	AWS managed	None	Provide access to Poly AVS devices
<input type="checkbox"/>	AlexaForBusinessReadOnlyAccess	AWS managed	None	Provide read only access to AlexaForB...
<input type="checkbox"/>	AmazonAPIGatewayAdministrator	AWS managed	None	Provides full access to create/edit/delete...
<input type="checkbox"/>	AmazonAPIGatewayInvokeFullAccess	AWS managed	None	Provides full access to invoke APIs in A...
<input type="checkbox"/>	AmazonAPIGatewayPushToCloudWatchLogs	AWS managed	None	Allows API Gateway to push logs to us...
<input type="checkbox"/>	AmazonAppFlowFullAccess	AWS managed	None	Provides full access to Amazon AppFlo...
<input type="checkbox"/>	AmazonAppFlowReadOnlyAccess	AWS managed	None	Provides read only access to Amazon A...
<input type="checkbox"/>	AmazonAppStreamFullAccess	AWS managed	None	Provides full access to Amazon AppStre...
<input type="checkbox"/>	AmazonAppStreamPCAAccess	AWS managed	None	Amazon AppStream 2.0 access to AWS...
<input type="checkbox"/>	AmazonAppStreamReadOnlyAccess	AWS managed	None	Provides read only access to Amazon A...
<input type="checkbox"/>	AmazonAppStreamServiceAccess	AWS managed	None	Default policy for Amazon AppStream ...
<input type="checkbox"/>	AmazonAthenaFullAccess	AWS managed	None	Provide full access to Amazon Athena ...
<input type="checkbox"/>	AmazonAugmentedAIFullAccess	AWS managed	None	Provides access to perform all operati...

At the bottom right, there are 'Cancel' and 'Create user group' buttons.

3. Specify User Details:

The screenshot shows the 'Create user' wizard, Step 1: Specify user details. The left sidebar shows navigation: IAM > Users > Create user, with tabs for Step 1 (Specify user details), Step 2 (Set permissions), and Step 3 (Review and create). The main content area has a title 'Specify user details' and a 'User details' section with a 'User name' input field. Below it is a note about character restrictions and a checkbox for 'Provide user access to the AWS Management Console - optional'. A note at the bottom explains how to generate programmatic access keys. At the bottom right are 'Cancel' and 'Next Step' buttons.

4. Set Permissions: Click on the “Add user to group” Option

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job function. [Learn more](#)

Permissions options

Add user to group Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

Copy permissions Copy all group memberships, attached managed policies, and inline policies from an existing user.

Attach policies directly Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

User groups (1/1)

Group name	Users	Attached policies	Created
project1	0	AdministratorAccess	2024-07-30 (Now)

Set permissions boundary - optional

Review and Create User:

- Review the user details and permissions.
- Click the "Create user" button.

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name user1	Console password type None	Require password reset No
--------------------	-------------------------------	------------------------------

Permissions summary

Name	Type	Used as
project1	Group	Permissions group

Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

Add new tag

You can add up to 50 more tags.

Download Credentials:

- After creating the user, you will see a success screen that displays the user's Access Key ID and Secret Access Key. Click the "Download .csv" button to save these credentials securely. This is the only time you will be able to view or download the Secret Access Key, so make sure to save it in a secure location.

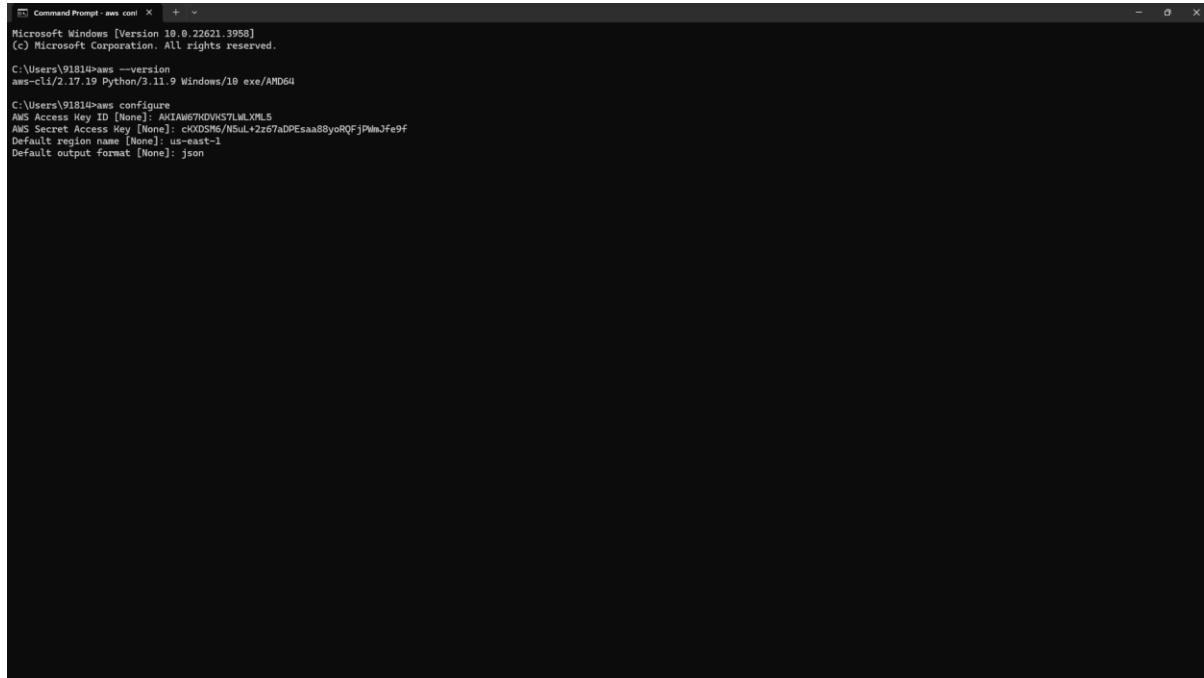
The screenshot shows the AWS IAM 'Access key created' page. The URL is [IAM > Users > user1 > Create access key](#). The page title is 'Access key created'. It states: 'This is the only time that the secret access key can be viewed or downloaded. You cannot recover it later. However, you can create a new access key any time.' Below this, there are three steps: Step 1 (Access key best practices & alternatives), Step 2 (optional Set description tag), and Step 3 (Retrieve access keys). The 'Access key' section shows 'Access key' AKIAW67KDVKS7LWLXMLS and 'Secret access key' (redacted). The 'Access key best practices' section lists: Never store your access key in plain text, in a code repository, or in code.; Disable or delete access key when no longer needed.; Enable least-privilege permissions.; Rotate access keys regularly. At the bottom are 'Download .csv file' and 'Done' buttons.

The screenshot shows an Excel spreadsheet titled 'user1_accessKeys.csv'. The file is in 'Excel (Product Activation Failed)' format. The data is as follows:

A1	B1	C1	D1	E1	F1	G1	H1	I1	J1	K1	L1	M1	N1	O1	P1	Q1	R1	S1	T1	U1	V1	W1	X1	Y1	Z1	AA1	AB1	AC1
1	Access key	Secret access key																										
2	AKIAW67KDVKS7LWLXMLS	ckDD5M6/N5uL+2267aDPesa8ByoRQfjPWhufe#f																										
3																												
4																												
5																												
6																												
7																												
8																												
9																												
10																												
11																												
12																												
13																												
14																												
15																												
16																												
17																												
18																												
19																												
20																												
21																												
22																												
23																												
24																												
25																												
26																												
27																												
28																												
29																												
30																												
31																												
32																												
33																												
34																												
35																												
36																												
37																												
38																												
39																												

Configure AWS CLI:

- Run the following command in your terminal or command prompt and follow the prompts to input your credentials and default region:



The screenshot shows a Windows Command Prompt window titled "Command Prompt - aws config". The window displays the following text:

```
Microsoft Windows [Version 10.0.22621.3958]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91814>aws --version
aws-cli/2.17.19 Python/3.11.9 Windows/10 exe/AMD64
AWS Access Key ID [None]: AVIAW67KDVHS7LWLXNL5
AWS Secret Access Key [None]: cxDXM61/NsUL+2z67aDPEsa88yoRQFjPmJFe9f
Default region name [None]: us-east-1
Default output format [None]: json
```

When prompted, enter your AWS Access Key ID, Secret Access Key, default region name (e.g., us-west-2), and default output format (e.g., json).

After you input these details, there won't be a confirmation message, but your credentials and configuration will be saved. To verify that everything is set up correctly, you can run a simple AWS CLI command, such as listing your S3 buckets:

```

Command Prompt
Microsoft Windows [Version 10.0 22621.3958]
(c) Microsoft Corporation. All rights reserved.

C:\Users\91814\ans --version
ans@cli/2.17.19 Python/3.11.9 Windows/10 exe/AMD64

C:\Users\91814\ans configure
AWS Access Key ID [None]: AKIAW67HDV57LwXNL5
AWS Secret Access Key [None]: cIXDS56/W5uL+2z67aDPEsa88yoRQfjPmJFe9f
Default region name [None]: us-east-1
Default output format [None]: json

C:\Users\91814\ans s3 ls
2024-07-29 23:03:52 config-bucket-U78842956453

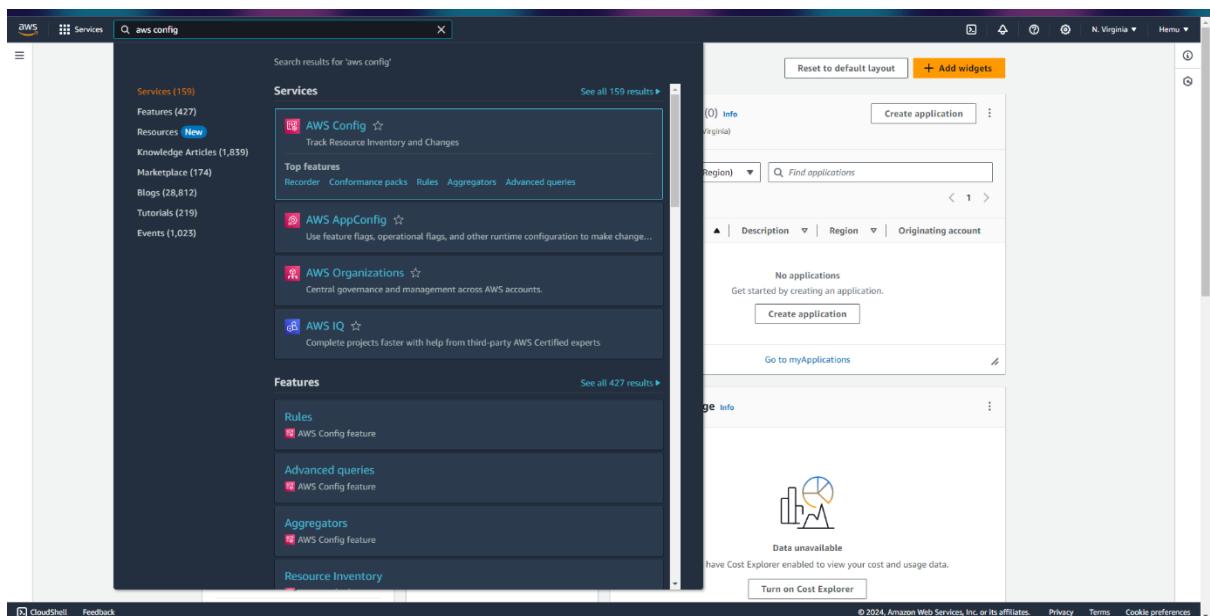
C:\Users\91814\

```

Create Managed Rules

Go to the AWS Config Console:

Open your AWS Management Console and sign in. In the console's services menu, type "Config" and select AWS Config. AWS Config is the primary service for tracking the configurations and changes of your AWS resources, ensuring compliance with your policies.



Now we can able to see the AWS config Interface:

The screenshot shows the AWS Config landing page. At the top, there's a banner with the title "AWS Config" and the subtitle "Record and evaluate configurations of your AWS resources". Below the banner, a section titled "How it works" contains a flowchart illustrating the process: "Configuration changes in your AWS resources" → "AWS Config records and evaluates changes into a consistent format" → "AWS Config automatically evaluates the recorded configurations against the configurations you specify" → "Access change history and compliance reports from the console or API, CloudWatch Events, or AWS Lambda when changes occur. Select configuration items and snapshot files to your S3 bucket for analysis." To the right of this diagram is a "Set up AWS Config" box with the subtext "A summarized view of AWS and non-AWS resources and the compliance status of the rules and the resources in each AWS Region." It features two buttons: "Get started" and "1-click setup". Further down, there are sections for "Pricing" (listing AWS Config, AWS Config rules, AWS GovCloud (US), and AWS Pricing Calculator) and "Learn more" (Documentation, FAQs, Partners, and What's new).

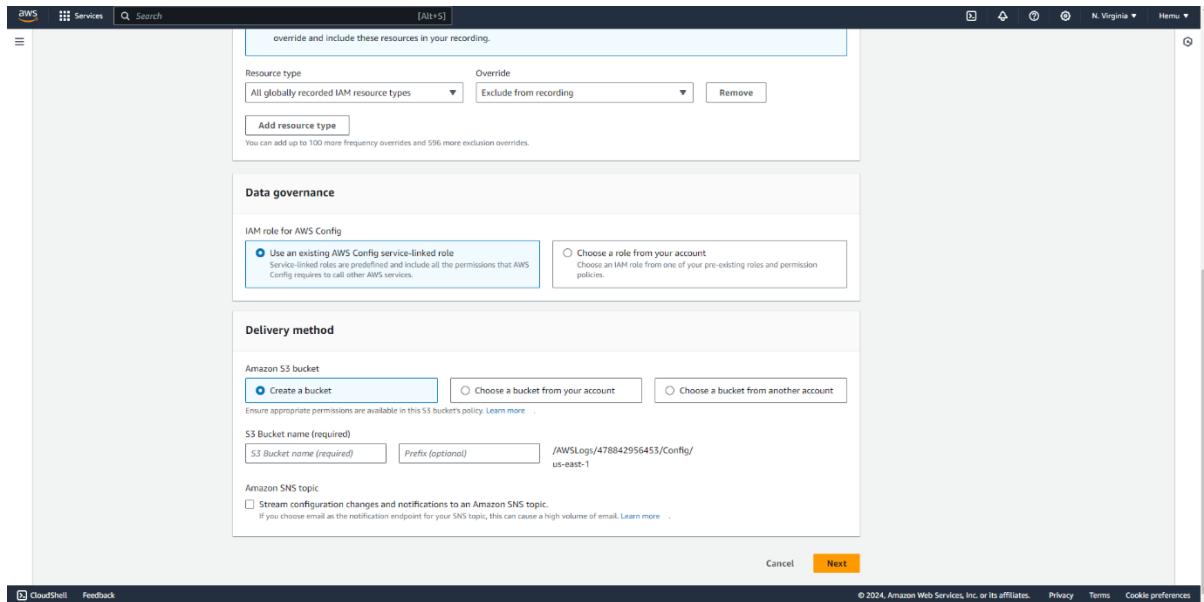
Set Up AWS Config:

1. Click on the **Get Started** button.
2. Start with the default settings to record all resources.

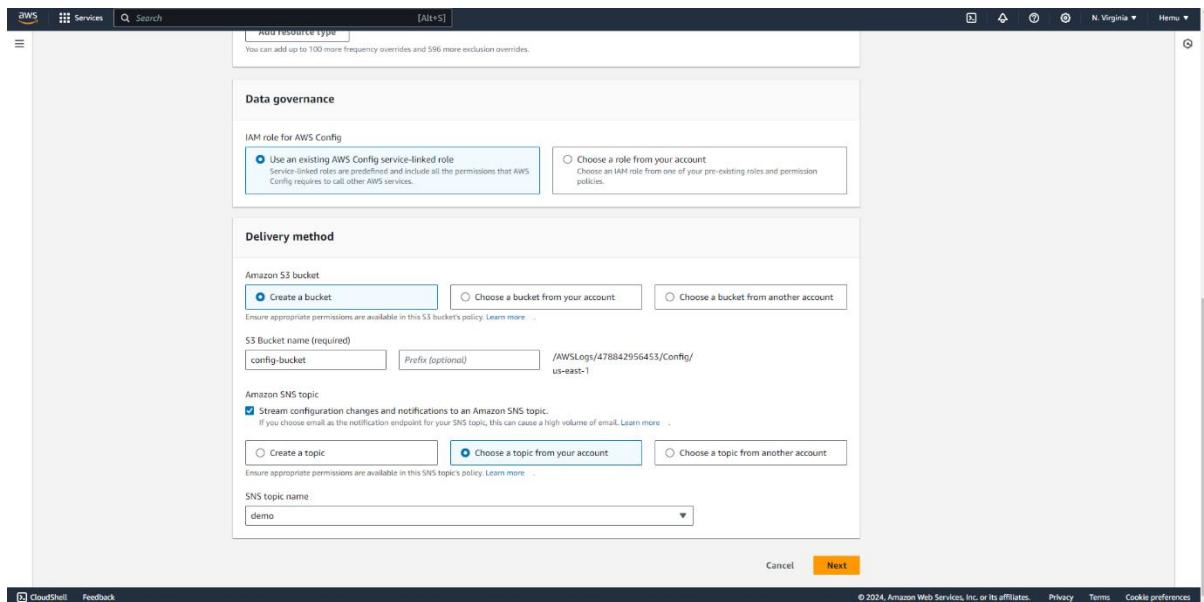
The screenshot shows the "Step 1 Settings" screen of the AWS Config setup wizard. The left sidebar lists "Step 1 Settings", "Step 2 Rules", and "Step 3 Review". The main area is titled "Settings" and contains several configuration sections:

- Recording method**: A radio button group for "Recording strategy" with "All resource types with customizable overrides" selected. A note states: "Customize AWS Config to record configuration changes for all supported resource types, or for only the supported resource types that are relevant to you. Globally recorded resources (RDS global clusters and IAM users, groups, roles, and customer managed policies) may be recorded in more than this Region. Learn more" and "You are charged based on the number of configuration items recorded. Pricing details".
 - Default settings**: "Recording frequency" with "Continuous recording" selected. A note says: "Configure the default recording frequency for all current and future supported resource types. It impacts the cost to your bill. Pricing details".
 - Override settings**: "Resource types to override" with a note: "Override the recording frequency for specific resource types, or exclude specific resource types from recording. If you change the recording frequency for a resource type or stop recording a resource type, the configuration items that were already recorded will remain unchanged." A note at the bottom says: "All globally recorded IAM resource types" are initially excluded from recording to help you reduce costs. Choose Remove to remove the override and include these resources in your recording."

1. Choose an S3 bucket for storing configuration history and snapshots. You can create a new bucket or use an existing one.



2. Choose an SNS topic for notifications. You can create a new topic or use an existing one.



3. Select an existing rule, and click on **Next** to review the settings:

<input type="checkbox"/>	cloudwatch-log-group-encrypted	CloudWatch, KMS	DETECTIVE
<input type="checkbox"/>	dax-encryption-enabled	DAX, encryption	DETECTIVE
<input type="checkbox"/>	dynamodb-table-encrypted-kms	DynamoDB, Kms	DETECTIVE
<input type="checkbox"/>	dynamodb-table-encryption-enabled	DynamoDb	DETECTIVE
<input checked="" type="checkbox"/>	ec2-ebs-encryption-by-default	EC2, EBS	DETECTIVE
<input type="checkbox"/>	efs-encrypted-check	EFS, KMS	DETECTIVE
<input type="checkbox"/>	eks-secrets-encrypted	EKS	DETECTIVE
<input type="checkbox"/>	elasticsearch-encrypted-at-rest	ElasticSearch, Encryption	DETECTIVE

Cancel Previous Next

4. Now, review your settings and click on **Confirm**:

Recording method	
Recording strategy Record all resource types with customizable overrides	Default recording frequency Continuous
► Resource types with override settings (4) ► Resource types with default settings (402)	
Delivery method	
S3 bucket name config-bucket SNS topic name demo	
▼ AWS Config rules (2) iam-user-mfa-enabled ec2-ebs-encryption-by-default	

Cancel Previous Confirm

5. Now we can able to see the AWS dashboard on the screen:

The screenshot shows the AWS Config Dashboard. On the left, there's a sidebar with navigation links like Dashboard, Conformance packs, Rules, Resources, Aggregators, Compliance Dashboard, Conformance packs, Rules, Inventory Dashboard, Resources, Authorizations, Advanced queries, Settings, and What's new. Below that are links for Documentation, Partners, FAQs, and Pricing. At the bottom of the sidebar are CloudShell and Feedback buttons.

The main area has several sections: 'Conformance Packs by Compliance Score' (No conformance packs deployed), 'Compliance status' (Rules: 0 Noncompliant rule(s), 0 Compliant rule(s); Resources: 0 Noncompliant resource(s), 0 Compliant resource(s)), 'Noncompliant rules by noncompliant resource count' (No noncompliant rules), and a 'Resource inventory (43)' section. To the right, there are three line charts under 'AWS Config usage metrics': 'Configuration Items Recorded' (Count: 32, 17, 2), 'Configuration Recorder Insufficient Permission...' (Count: 1), and 'AWS Config success metrics' (Change Notifications Deliv..., Config History Export Failed, Config Snapshot Export Fail...).

Create AWS Config Rules:

In the AWS Config console, go to **Rules** and click on **Add Rule**.

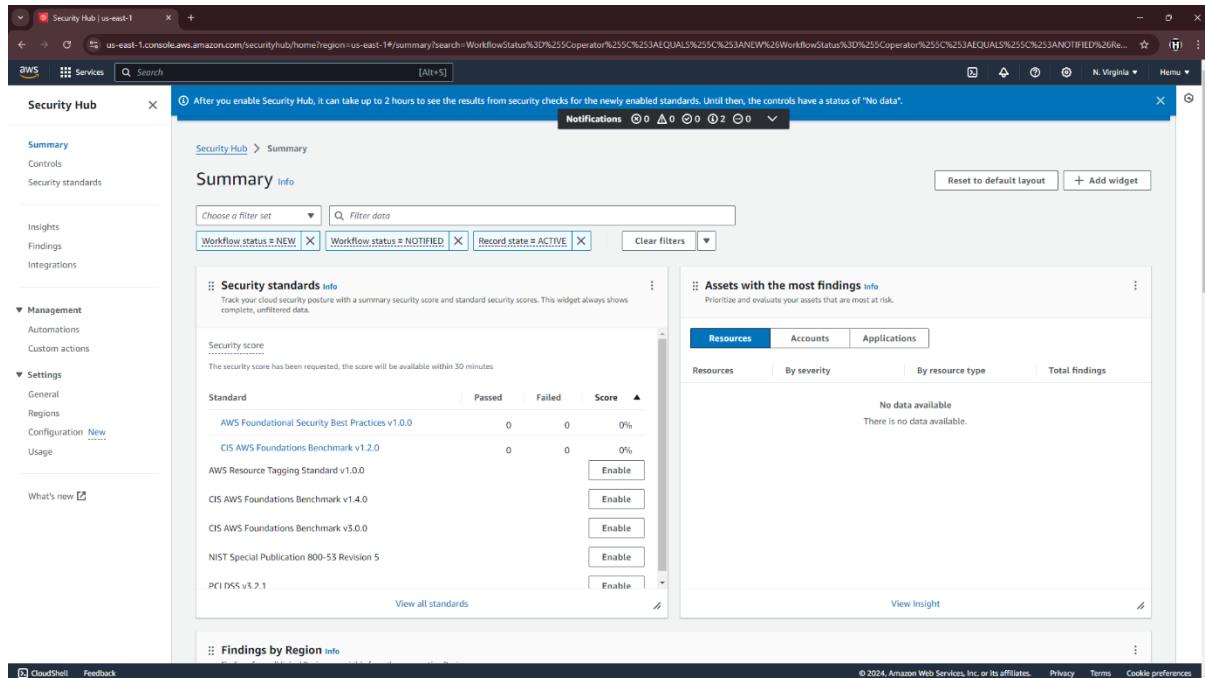
The screenshot shows the 'Rules' page in the AWS Config console. The sidebar is identical to the one in the previous screenshot. The main area has a 'Rules' heading with a sub-section 'A rule is a compliance check that helps you manage your ideal configuration settings. AWS Config evaluates whether your resource configurations comply with relevant rules and displays the compliance results.' Below this is a table titled 'Rules' with columns: Name, Remediation action, Type, Enabled evaluation mode, and Detective compliance. Two rules are listed: 'ex2-ebs-encryption-by-default' (Not set, AWS managed, DETECTIVE) and 'iam-user-mfa-enabled' (Not set, AWS managed, DETECTIVE). At the top right of the table are buttons for View details, Edit rule, Actions, and Add rule. The URL at the bottom of the page is <https://us-east-1.console.aws.amazon.com/config/home?region=us-east-1#rules>.

Enable AWS Security Hub:

Set Up Security Hub

Click on Go to Security Hub

- After navigating to the Security Hub service, you will see an overview page.
- Click on the button or link that says "Go to Security Hub" to proceed with the setup.



Click Enable Security Hub

- On the Security Hub main page, you will see a button labeled "Enable Security Hub."
- Click on this button to start the enabling process.

AWS Security Hub Will Automatically Begin Collecting and Analyzing Data

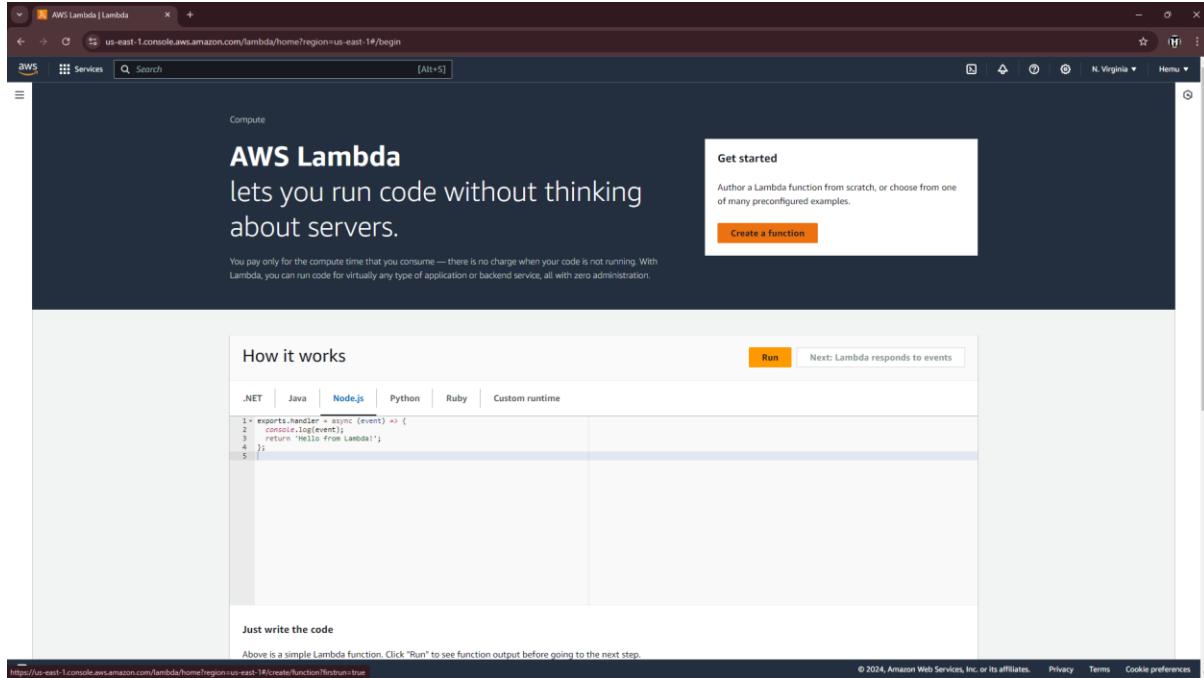
- Once you click "Enable Security Hub," the service will start to collect and analyze security-related data from your AWS account.
- Security Hub aggregates findings from various AWS services and third-party tools to provide a comprehensive view of your security posture.

The initial setup might take a few minutes, and after that, you will start seeing security findings and insights in the Security Hub dashboard.

Automate Remediation with AWS Systems Manager

Create a Remediation Lambda Function

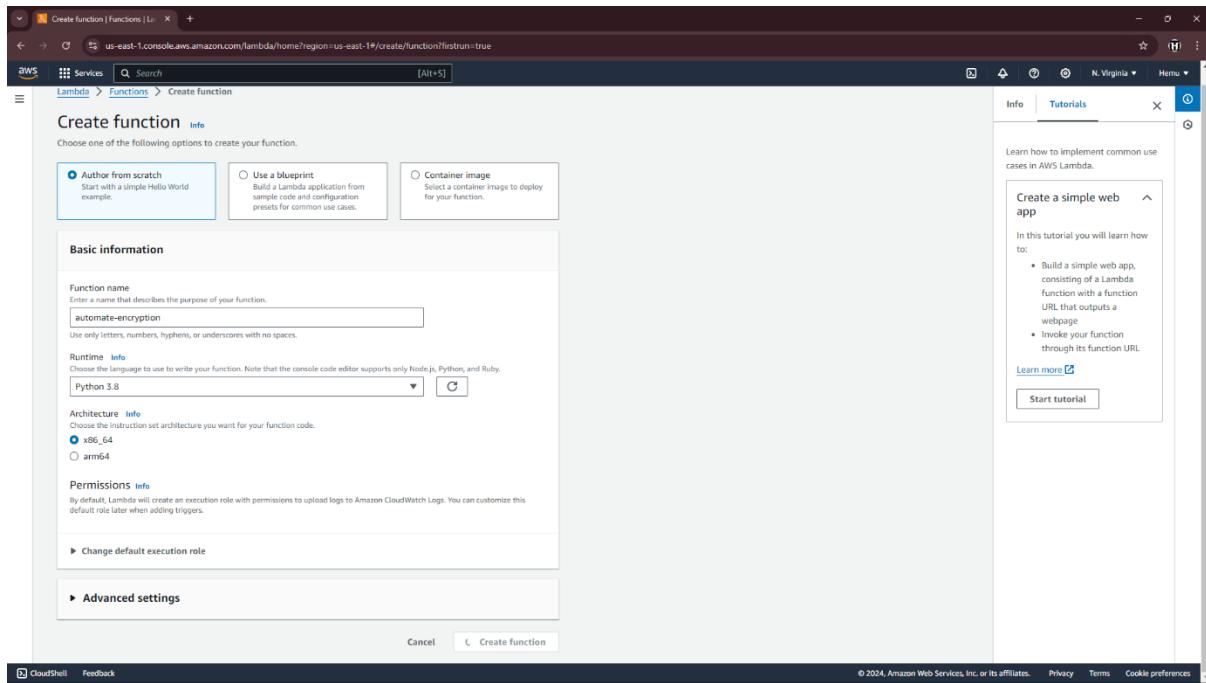
Navigate to the Lambda Service



In the top search bar, type "Lambda" and select "Lambda" from the dropdown list.

Create a Lambda Function

- Click the "Create function" button.
- Choose "Author from scratch."
- Provide a name for your function, such as "automate-encryption".
- Choose a runtime (e.g., Python 3.8).
- Click "Create function."



Write Your Function Code:

In the function code editor, input the following Python code to remediate non-compliant resources, specifically to encrypt an unencrypted EBS volume:

```
import boto3

def lambda_handler(event, context):
    ec2 = boto3.client('ec2')

    try:
        # Validate the event contains the necessary details
        if 'detail' not in event:
            raise ValueError("Event does not contain 'detail'")
        if 'resourceId' not in event['detail']:
            raise ValueError("Event detail does not contain 'resourceId'")
        if 'resourceInstanceId' not in event['detail']:
            raise ValueError("Event detail does not contain 'resourceInstanceId'")

        volume_id = event['detail']['resourceId']
        instance_id = event['detail']['resourceInstanceId']
        region = 'us-east-1' # Replace with your region
        availability_zone = 'us-east-1a' # Replace with your AZ

        # Create snapshot of the unencrypted volume
        snapshot = ec2.create_snapshot(VolumeId=volume_id, Description="Snapshot for volume encryption")
        snapshot_id = snapshot['SnapshotId']

        # Wait for the snapshot to complete
        waiter = ec2.get_waiter('snapshot_completed')
        waiter.wait(SnapshotIds=[snapshot_id])

    except Exception as e:
        print(f"An error occurred: {e}")

    finally:
        context.done = True
```

Line by Line Explanation of the Function:

```
import boto3
```

boto3: The AWS SDK for Python, which allows Python developers to write software that makes use of services like Amazon EC2 and S3.

```
def lambda_handler(event, context):
```

```
    ec2 = boto3.client('ec2')
```

- **lambda_handler:** The main function that AWS Lambda calls when the function is triggered.
- **event:** The event data that triggered the Lambda function. In this context, it contains details about the non-compliant resource.
- **context:** Provides runtime information to the handler.
- **ec2:** A client object to interact with the EC2 service.

```
try:
```

```
    if 'detail' not in event:
```

```
        raise ValueError("Event does not contain 'detail'")
```

```
    if 'resourceId' not in event['detail']:
```

```
        raise ValueError("Event detail does not contain 'resourceId'")
```

```
    if 'resourceInstanceId' not in event['detail']:
```

```
        raise ValueError("Event detail does not contain 'resourceInstanceId'")
```

- Checks if the event contains necessary details:
- **detail:** Main content of the event.
- **resourceId:** The ID of the EBS volume.
- **resourceInstanceId:** The ID of the instance the volume is attached to.
- Raises a ValueError if any of these are missing.

Extracting Details and Setting Up

```
volume_id = event['detail']['resourceId']
instance_id = event['detail']['resourceInstanceId']
region = 'us-east-1' # Replace with your region
availability_zone = 'us-east-1a' # Replace with your AZ
```

- Extracts the volume ID and instance ID from the event details.
- Sets the AWS region and availability zone (replace these with your specific values).

Creating and Copying Snapshots:

```
# Create snapshot of the unencrypted volume
snapshot = ec2.create_snapshot(VolumeId=volume_id,
Description="Snapshot for volume encryption")
snapshot_id = snapshot['SnapshotId']

# Wait for the snapshot to complete
waiter = ec2.get_waiter('snapshot_completed')
waiter.wait(SnapshotIds=[snapshot_id])

# Copy the snapshot with encryption
encrypted_snapshot = ec2.copy_snapshot(
    SourceSnapshotId=snapshot_id,
    SourceRegion=region,
    Encrypted=True,
    Description="Encrypted copy of snapshot"
)
encrypted_snapshot_id = encrypted_snapshot['SnapshotId']
```

```
# Wait for the encrypted snapshot to complete
waiter.wait(SnapshotIds=[encrypted_snapshot_id])
```

- **Create Snapshot:** Creates a snapshot of the unencrypted volume.
- **Wait for Snapshot Completion:** Uses a waiter to ensure the snapshot operation is complete.
- **Copy Snapshot with Encryption:** Copies the snapshot and enables encryption.
- **Wait for Encrypted Snapshot Completion:** Ensures the encrypted snapshot operation is complete.

```
# Copy the snapshot with encryption
encrypted_snapshot = ec2.copy_snapshot(
    SourceSnapshotId=snapshot_id,
    SourceRegion=region,
    Encrypted=True,
    Description="Encrypted copy of snapshot"
)
encrypted_snapshot_id = encrypted_snapshot['SnapshotId']

# Wait for the encrypted snapshot to complete
waiter.wait(SnapshotIds=[encrypted_snapshot_id])

# Create a new encrypted volume from the encrypted snapshot
encrypted_volume = ec2.create_volume(
    SnapshotId=encrypted_snapshot_id,
    AvailabilityZone=availability_zone
)
encrypted_volume_id = encrypted_volume['VolumeId']

# Wait for the volume to become available
waiter = ec2.get_waiter('volume_available')
waiter.wait(VolumeIds=[encrypted_volume_id])
```

```
# Create a new encrypted volume from the encrypted snapshot
encrypted_volume = ec2.create_volume(
    SnapshotId=encrypted_snapshot_id,
    AvailabilityZone=availability_zone
```

```
)  
encrypted_volume_id = encrypted_volume['VolumeId']
```

Wait for the volume to become available

```
waiter = ec2.get_waiter('volume_available')
```

```
waiter.wait(VolumeIds=[encrypted_volume_id])
```

- **Create Encrypted Volume:** Creates a new EBS volume from the encrypted snapshot.
- **Wait for Volume Availability:** Ensures the new encrypted volume is ready to use.

```
# Attach the encrypted volume to the instance  
ec2.attach_volume(  
    VolumeId=encrypted_volume_id,  
    InstanceId=instance_id,  
    Device='/dev/sdf'  
)  
  
# Delete the unencrypted snapshot  
ec2.delete_snapshot(SnapshotId=snapshot_id)  
  
return {  
    'statusCode': 200,  
    'body': 'Volume encrypted and attached successfully.'  
}  
  
except ValueError as ve:  
    return {  
        'statusCode': 400,  
        'body': str(ve)  
    }  
except Exception as e:  
    return {  
        'statusCode': 500,  
        'body': f'Error: {str(e)}'  
}
```

```

# Attach the encrypted volume to the instance
ec2.attach_volume(
    VolumeId=encrypted_volume_id,
    InstanceId=instance_id,
    Device='/dev/sdf'
)

# Delete the unencrypted snapshot
ec2.delete_snapshot(SnapshotId=snapshot_id)



- Attach Encrypted Volume: Attaches the new encrypted volume to the specified EC2 instance.
- Delete Unencrypted Snapshot: Cleans up by deleting the original unencrypted snapshot.



return {
    'statusCode': 200,
    'body': 'Volume encrypted and attached successfully.'
}

except ValueError as ve:
    return {
        'statusCode': 400,
        'body': str(ve)
    }

except Exception as e:
    return {
        'statusCode': 500,
        'body': f'Error: {str(e)}'
}

```

```
}
```

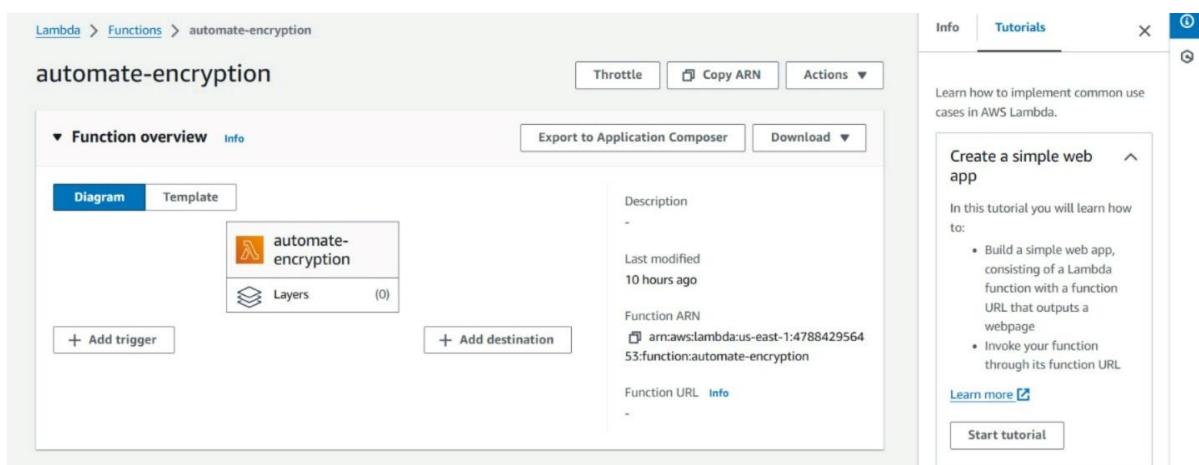
- **Return Success:** Returns a success message if everything works correctly.
- **Handle Value Errors:** Catches and returns validation errors with a 400 status code.
- **Handle General Exceptions:** Catches and returns any other errors with a 500 status code.

Summary of the code:

- This Lambda function automates the encryption of non-compliant EBS volumes through a series of well-defined steps. First, it creates a snapshot of the unencrypted volume to ensure there is a backup of the current state. Then, it creates an encrypted copy of this snapshot, effectively converting the backup into an encrypted format. Using this encrypted snapshot, the function proceeds to create a new encrypted EBS volume.
- Once the new encrypted volume is ready, it attaches this volume to the same EC2 instance from which the original unencrypted volume was taken, maintaining the continuity of data access. After the new volume is successfully attached, the function cleans up by deleting the original unencrypted snapshot to avoid any unnecessary storage costs and potential security risks. Throughout the process, the function includes detailed error handling and logging to ensure smooth operation and provide insights into any issues that may arise, making the entire workflow efficient and reliable.

Deploy the Function

- Click "Deploy" to save and deploy your Lambda function.

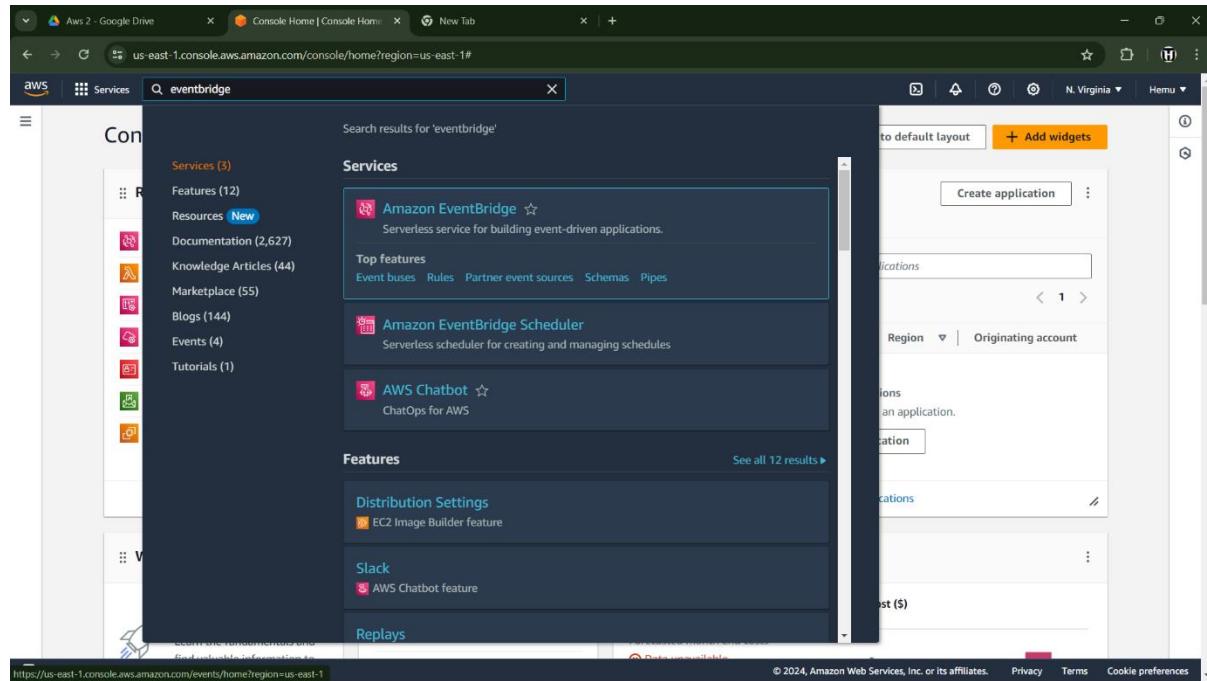


Set Up AWS CloudWatch Events (EventBridge)

- Setting up AWS CloudWatch Events (now known as EventBridge) involves creating a rule to trigger specific actions when certain conditions are met.

Navigate to EventBridge:

- Go to the AWS Management Console and navigate to Amazon EventBridge. EventBridge is a serverless event bus that allows you to create rules to react to changes in your AWS resources or other events.



Create a New Rule:

- Click "Create rule." Provide a name (e.g., EBSVolumeComplianceRemediation) to identify the rule easily. Optionally, add a description to explain its purpose. This rule will help automate responses to specific events.

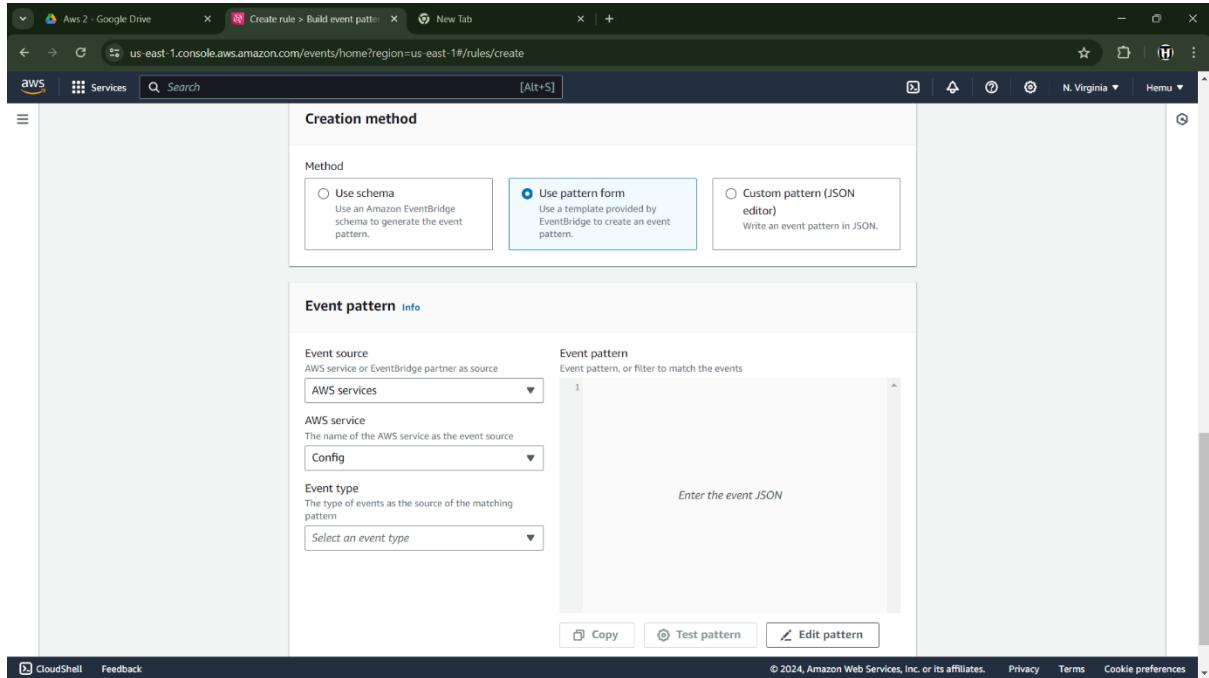
The screenshot shows the Amazon EventBridge console homepage. On the left, there is a navigation sidebar with links for Dashboard, Developer resources (Learn, Sandbox, Quick starts), Buses (Event buses, Rules, Global endpoints, Archives, Replays), Pipes (Pipes), Scheduler (Schedules, Schedule groups), and Integration (Partner event sources). The main content area features the title "Amazon EventBridge" and the subtitle "A serverless service for building event-driven applications". It includes a brief description of what EventBridge is and how it works, followed by a "How it works" section with a video thumbnail titled "Serverless 101: Amazon EventBridge". To the right, there are sections for "Get started" (with options for EventBridge Rule, EventBridge Pipes, EventBridge Schedule, and EventBridge Schema registry) and "Pricing". At the bottom, there is a copyright notice and links for CloudShell, Feedback, Privacy, Terms, and Cookie preferences.

Define the Event Source:

- Under Event Source, select AWS Events or EventBridge partner events.

The screenshot shows the "Create rule > Build event pattern" step in the Amazon EventBridge console. On the left, there is a sidebar with steps: Step 1 (Define rule detail), Step 2 (Build event pattern), Step 3 (Select target(s)), Step 4 - optional (Configure tags), and Step 5 (Review and create). The main content area is titled "Build event pattern" and contains a "Event source" section. In this section, there is an "Event source" dropdown with three options: "AWS events or EventBridge partner events" (selected), "Other" (Custom events or events sent from more than one source, e.g. events from AWS services and partners.), and "All events" (All events sent to your account.). Below this is a "Sample event - optional" section with a note about referencing a sample event for testing. At the bottom, there is a "in a sample event" link and a copyright notice at the very bottom.

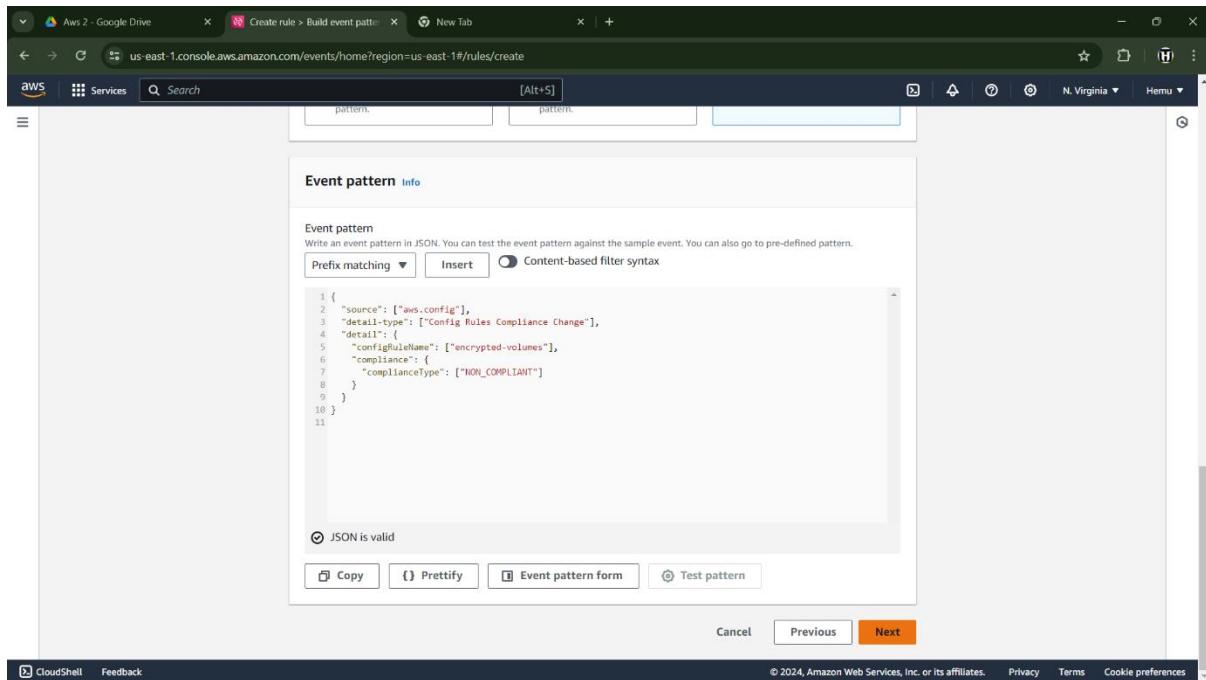
- Choose Service Name as Config, and select Config Rules Compliance Change for Event Type.



- This sets up the rule to listen for changes in the compliance status of AWS Config rules.

Add Event Pattern:

- We have used the following event pattern in JSON to capture non-compliant resources.



Configuration of the Event pattern:

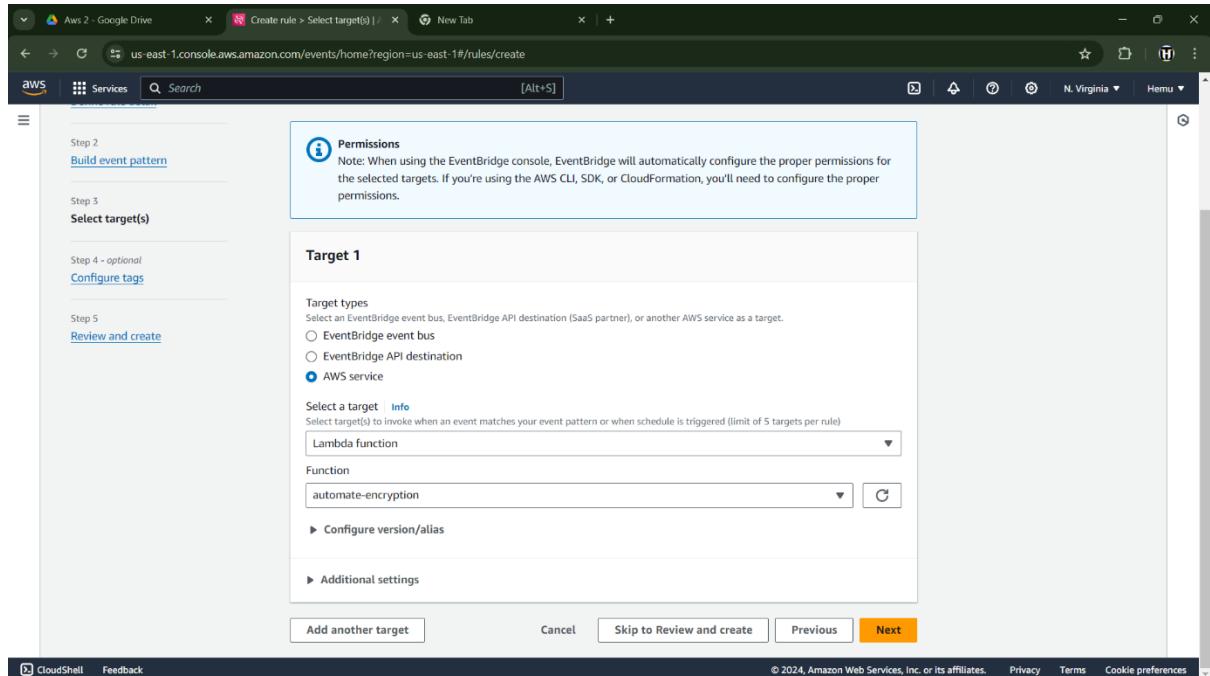
- The event pattern is configured to capture specific non-compliant events from AWS Config.

```
{
  "source": ["aws.config"],
  "detail-type": ["Config Rules Compliance Change"],
  "detail": {
    "configRuleName": ["encrypted-volumes"],
    "compliance": {
      "complianceType": ["NON_COMPLIANT"]
    }
  }
}
```

- The event pattern is configured to capture changes in compliance status for AWS Config rules. It specifically targets events from **aws.config** where the detail type is **Config Rules Compliance Change**. Within these events, it filters for a Config rule named encrypted-volumes with a compliance status of NON_COMPLIANT. This ensures that only non-compliant encrypted volumes trigger the event.
- Click on the Next button to select the targets.

Select Target:

- Under Select Targets, choose the **Lambda function**. Select the Lambda function we created for remediation.



- This target will handle the non-compliant events by executing predefined actions.
- Click on the Next Button to review the rule.

Create Rule:

- Reviewing our settings and click Create Rule.

A screenshot of the AWS EventBridge 'Create rule > Review and create' wizard. The page shows the 'Review and create' step with the following details:

- Step 1: Define rule detail** (selected)
- Step 2: Build event pattern**
- Step 3: Select target(s)**
- Step 4 - optional: Configure tags**
- Step 5: Review and create**

Step 1: Define rule detail

Rule name	Status	Event bus
balee1	Enabled	default
Description	Rule type	
	Standard rule	

Step 2: Build event pattern

Event pattern [info](#)

```

1 {
2   "source": ["aws.config"],
3   "detail-type": ["Config Rules Compliance Change"],
4   "detail": {
5     "configRuleName": ["encrypted-volumes"],
6     "compliance": {
7       "complianceType": ["NON_COMPLIANT"]
8     }
9   }
10 }

```

[Copy](#)

Step 3: Select target(s)

Targets

Details	Target Name	Type	Arn	Input	Role
automate...	Lambda	arn:aws:lambda:us-east-1:...	Matched		

A screenshot of the AWS EventBridge 'Create rule > Review and create' wizard. The page shows the 'Review and create' step with the following details:

- Step 1: Define rule detail** (selected)
- Step 2: Build event pattern**
- Step 3: Select target(s)**
- Step 4 - optional: Configure tags**
- Step 5: Review and create**

Step 1: Define rule detail

Rule name	Status	Event bus
balee1	Enabled	default
Description	Rule type	
	Standard rule	

Step 2: Build event pattern

Event pattern [info](#)

```

1 {
2   "source": ["aws.config"],
3   "detail-type": ["Config Rules Compliance Change"],
4   "detail": {
5     "configRuleName": ["encrypted-volumes"],
6     "compliance": {
7       "complianceType": ["NON_COMPLIANT"]
8     }
9   }
10 }

```

[Copy](#)

Step 3: Select target(s)

Targets

Details	Target Name	Type	Arn	Input	Role
automate...	Lambda	arn:aws:lambda:us-east-1:...	Matched		

The screenshot shows the AWS EventBridge rule creation process. In Step 3: Select target(s), a Lambda function named 'automate-encryption' is selected as the target. In Step 4: Configure tag(s), no tags are currently associated with the resource.

- This finalizes the rule, enabling automated detection and remediation of non-compliant EBS volumes, ensuring they comply with our security policies.

By setting up this EventBridge rule, we're automating the monitoring and remediation process for non-compliant resources, ensuring our AWS environment remains secure and compliant with our policies.

Test the Setup

Trigger a Compliance Violation

Create a Non-Compliant Resource: To test whether the setup correctly identifies and remediates non-compliant resources, you need to create a resource that violates the compliance rule.

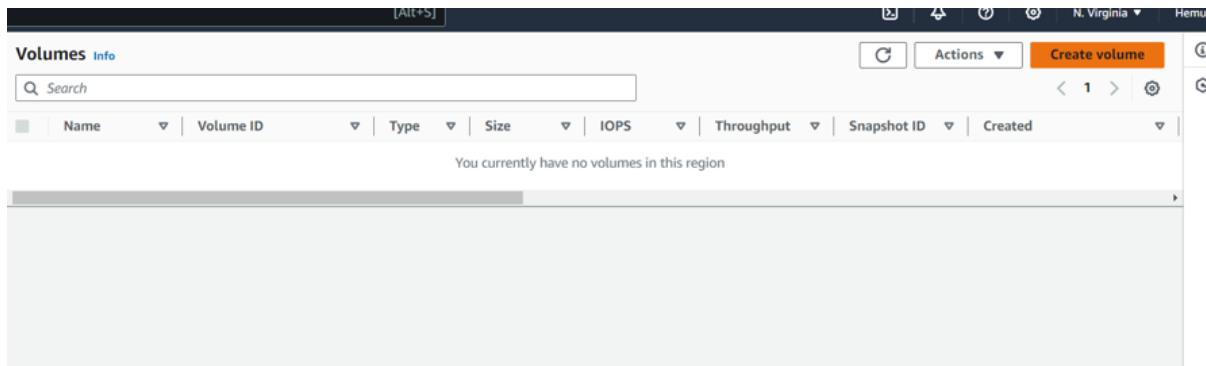
Go to the EC2 Dashboard in the AWS Management Console:

- Log in to your AWS Management Console.
- In the top left corner, click on the services menu and select "EC2" under "Compute."

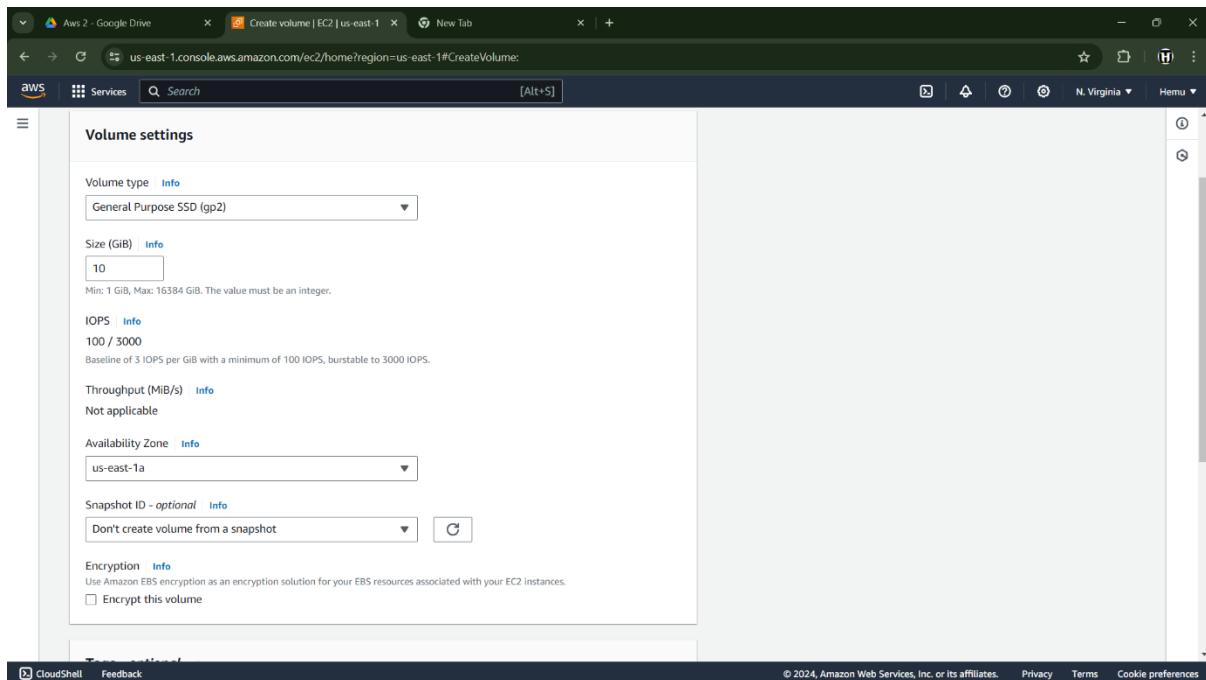
Create a New EBS Volume Without Encryption:

- In the EC2 Dashboard, navigate to **Volumes** under the "Elastic Block Store" section.

- Click on the "Create Volume" button in the top right corner.



- Choose your desired volume type, size, and availability zone.



- Ensure the "Encrypt this volume" option is **not** checked. This will create an unencrypted volume, simulating a compliance violation.
- Click "Create Volume" to finish.

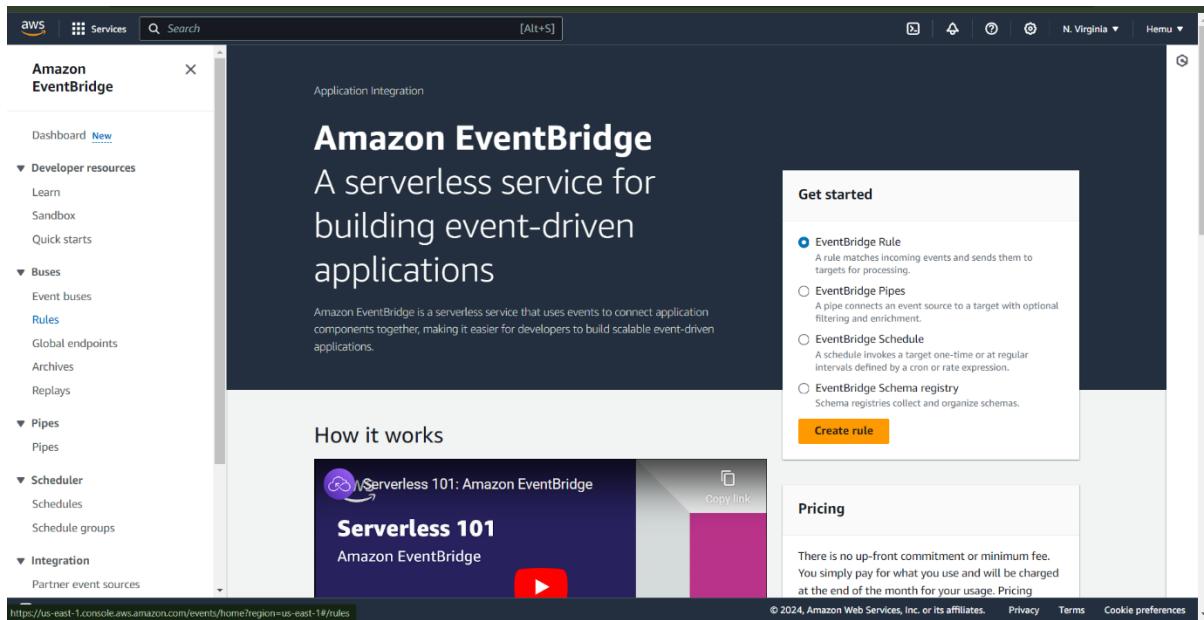
By creating an unencrypted EBS volume, you simulate a non-compliant resource according to the encrypted-volumes Config rule.

Monitor the Event and Lambda Function

Check EventBridge: After creating the non-compliant resource, you need to verify that EventBridge captures the event.

1. Go to the EventBridge Console:

- o Navigate to the Amazon EventBridge service in the AWS Management Console.



2. Check the Rules Section:

- o In the EventBridge console, go to the "Rules" section.
- o Ensure that the rule you created (e.g., EBSVolumeComplianceRemediation) is active. The rule should be "Enabled".

The screenshot shows the AWS EventBridge Rules page. On the left, there's a navigation sidebar with options like Dashboard, Developer resources, Buses, Pipes, Scheduler, and Integration. The main area is titled 'Rules' and contains a section for 'Select event bus' with a dropdown set to 'default'. Below this is a table titled 'Rules (2)' showing two entries:

Name	Status	Type	ARN
balee1	Enabled	Standard	arn:aws:events:us-east-1:478842956453:rule/balee1
fugi	Enabled	Standard	arn:aws:events:us-east-1:478842956453:rule/fugi

3. Verify Event Capture:

- In the EventBridge console, check the "Metrics" tab or related sections to see if any events matching your pattern have been captured. This will confirm that the event pattern correctly identifies the compliance change event.

Verify Lambda Execution

- Once you confirm that EventBridge captures the event, the next step is to ensure that the Lambda function is triggered and performs the remediation.

1. Go to the Lambda Console:

- Navigate to the AWS Lambda service in the AWS Management Console.

The screenshot shows the AWS Lambda console homepage. At the top, there's a 'Get started' box with the text: 'Author a Lambda function from scratch, or choose from one of many preconfigured examples.' Below it is a 'Create a function' button. The main area features a 'How it works' section with tabs for '.NET', 'Java', 'Node.js', 'Python', 'Ruby', and 'Custom runtime'. The 'Node.js' tab is selected, showing the following code:

```

1 exports.handler = async (event) => {
2   console.log(event);
3   return 'Hello from Lambda!';
4 }
5

```

Below the code editor, there's a note: 'Just write the code' and 'Above is a simple Lambda function. Click "Run" to see function output before going to the next step.'

At the bottom of the page, the URL is https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1&createFunction=true, and the footer includes links for Privacy, Terms, and Cookie preferences.

2. Check the Monitoring Tab of Your Lambda Function:

- Find the Lambda function you created for remediation.

The screenshot shows the Amazon EventBridge console. On the left, there's a sidebar with options like Dashboard, Developer resources, Buses, Rules, Pipes, Scheduler, and Integration. The main area is titled 'balee1' and shows 'Rule details' for a rule named 'balee1'. The 'Monitoring' tab is selected, displaying the CloudWatch Metrics configuration. The rule details table includes:

Rule name	Status	Event bus name	Type
balee1	Enabled	default	Standard

The 'Event pattern' section shows the following JSON configuration:

```

1 {
2   "source": ["aws.config"],
3   "detail-type": ["Config Rules Compliance Change"],
4   "detail": {
5     "configRuleName": ["encrypted-volumes"],
6     "compliance": {
7       "complianceType": ["NON_COMPLIANT"]
8     }
9   }
10 }

```

At the bottom, there's a 'Copy' button. The footer of the browser window shows the URL https://us-east-1.console.aws.amazon.com/events/home?region=us-east-1#eventbus/default/rules/balee1 and the standard AWS footer with links for Privacy, Terms, and Cookie preferences.

- Click on the function to open its details page.

The screenshot shows the AWS Lambda function configuration page for 'automate-encryption'. The 'Targets' tab is selected, displaying a table with one row. The row details are as follows:

Details	Target Name	Type	Arn	Input	Role
automate-encryption	automate-encryption	Lambda function	arn:aws:lambda:us-east-1:478842956453:function:automate-encryption	Matched event	-

Input to target: Matched event
Additional parameters: --
Dead-letter queue (DLQ): -

- Go to the "Monitoring" tab.

The screenshot shows the AWS EC2 Dashboard. The 'Monitoring' tab is selected, displaying the 'Service health' section. It shows the following status:

Region: US East (N. Virginia)
Status: This service is operating normally.

- Here, you should see invocations, duration, errors, and other metrics. Check if there has been an invocation corresponding to the event you created by violating compliance.

3. Review Logs in CloudWatch Logs:

- In the Lambda function details, there is a link to "View logs in CloudWatch." Click this link to open the CloudWatch Logs console.
- In CloudWatch Logs, find the log group associated with your Lambda function.
- Review the logs to verify that the Lambda function executed correctly. The logs should show details of the event and the actions taken by the function. Look for any error messages or debug information to ensure the function behaves as expected.

The screenshot shows the AWS CloudWatch Log Streams interface. On the left, a sidebar menu is visible with sections like 'Logs' (selected), 'Log groups', 'Metrics', 'X-Ray traces', 'Events', 'Application Signals', 'Network monitoring', and 'Insights'. The main area displays a table titled 'Log streams (13)'. The table has two columns: 'Log stream' (containing hyperlinks to log details) and 'Last event time' (containing UTC timestamps). The log streams listed are:

Last event time	Log stream
2024-07-31 16:53:07 (UTC)	2024/07/31/[SLATEST]3433532f6f8f4d189d6a850eef22fd
2024-07-31 14:14:59 (UTC)	2024/07/31/[SLATEST]f2d571d0a8c04b22a3ad473575d4ea26
2024-07-31 14:04:13 (UTC)	2024/07/31/[SLATEST]f56e8ff557c4a9c808a2c94ef3b38f3
2024-07-31 13:34:50 (UTC)	2024/07/31/[SLATEST]ef7b49e3bd16404aa877149ac5e19002
2024-07-31 13:21:23 (UTC)	2024/07/31/[SLATEST]bda007e7b3a94fb8857b07126c5be4e3
2024-07-31 13:16:06 (UTC)	2024/07/31/[SLATEST]f0b1d2aa2f104d1abfda4e9db77581df
2024-07-31 13:04:38 (UTC)	2024/07/31/[SLATEST]a1b9cc5418764b7495af463e6e5e3e14
2024-07-31 12:40:14 (UTC)	2024/07/31/[SLATEST]7f119cea2fd649539bba6162a8831fe6
2024-07-31 12:34:19 (UTC)	2024/07/31/[SLATEST]af1f1961bb5e14a19b7a740bc16f431e3
2024-07-31 07:54:41 (UTC)	2024/07/31/[SLATEST]c37ab303bed9484ba5af6f2808075cda
2024-07-31 07:50:12 (UTC)	2024/07/31/[SLATEST]4b9bb055fa864d1ea3736e837bef01ee
2024-07-31 07:48:15 (UTC)	2024/07/31/[SLATEST]0c940995fd8545d3a2b3d2976f1c5f23
2024-07-31 04:39:03 (UTC)	2024/07/31/[SLATEST]0ab6580cdb45456c90b8637ba6fd21a9

By following these steps, we validate that your EventBridge rule and Lambda function correctly identify and remediate non-compliant resources.

Review and Validate Compliance Reports

1. AWS Config Console:

- Go to the AWS Config console and check the compliance status of your resources.

The screenshot shows the AWS Config homepage. At the top, there's a banner with the text "AWS Config | us-east-1" and "aws.amazon.com/config/home?region=us-east-1#home". Below the banner, the title "AWS Config" is displayed with the subtitle "Record and evaluate configurations of your AWS resources". A subtext explains that AWS Config provides a detailed view of resources and their configurations over time. To the right, there's a "Set up AWS Config" box with "Get started" and "1-click setup" buttons, and a "Pricing" section listing prices for AWS Config, rules, GovCloud, and the Pricing Calculator. On the left, a "How it works" diagram illustrates the process: configuration changes in AWS resources lead to AWS Config recording and evaluating them into a consistent format, which is then used to compare against specified configurations. Below the diagram, there's a "Benefits" section.

- Ensure that the encrypted-volume rule is being evaluated and that violations are being captured.

The screenshot shows the AWS Config Dashboard. The left sidebar includes links for Conformance packs, Rules, Resources, Aggregators, Compliance Dashboard, Inventory Dashboard, Resources, Authorizations, Advanced queries, Settings, What's new, Documentation, Partners, FAQs, and Pricing. The main dashboard features three main sections: "Conformance Packs by Compliance Score" (which shows no deployed packs), "Compliance status" (listing 30 non-compliant rules and 14 compliant rules), and "Noncompliant rules by noncompliant resource count" (a table with columns for Name and Compliance). To the right, there's a "AWS Config usage metrics" section with a chart titled "Configuration Items Recorded" and another titled "Configuration Recorder Insuff...". The bottom of the screen shows standard AWS navigation links for CloudShell, Feedback, and Copyright information.

2. AWS Security Hub:

- If you have AWS Security Hub enabled, review the compliance findings and ensure that all identified issues are being remediated.

Project Done by:

Balakrishnan S