

Kyndryl AWS Bootcamp Training – 2024 Batch - 2

Task Assignment 2: Implementing Expiry Key Rotation over 90
Days Using AWS CloudTrail with SES Notifications

Presented by:

Name: KARTHIKEYAN J

Dept & Class: CSE A

Register Number: 111921CS01065

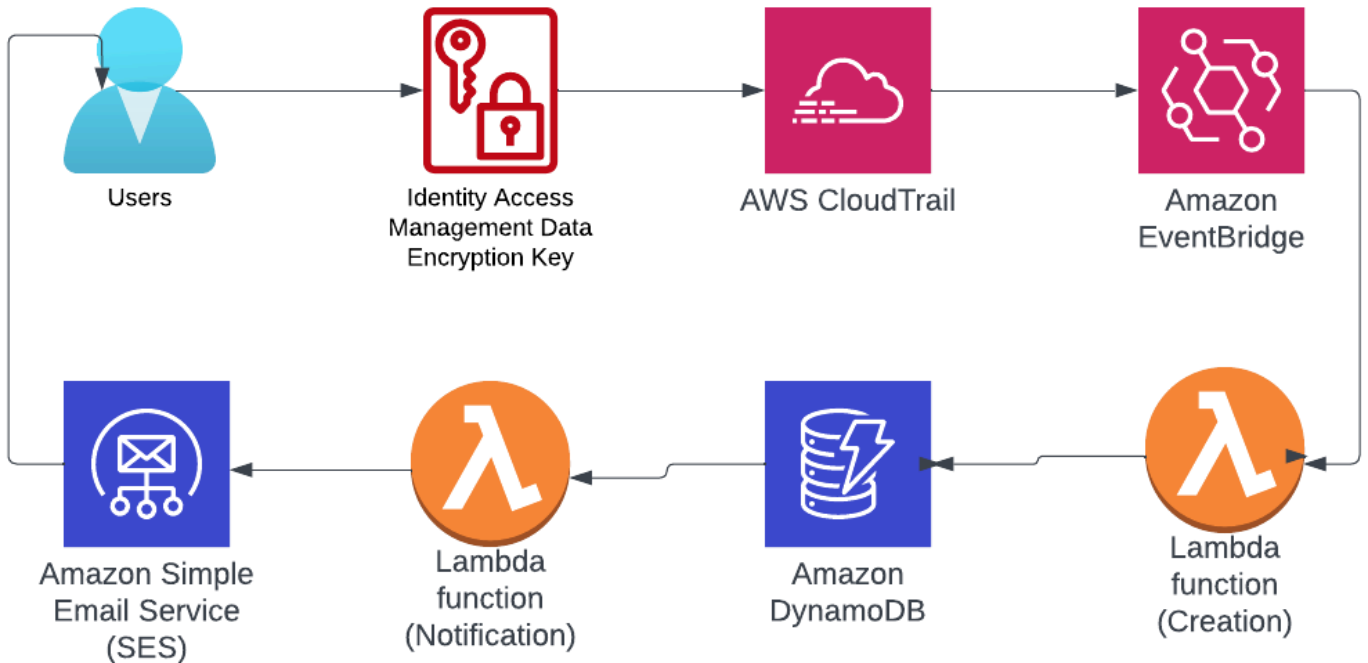
Documentation: Configuration steps of Expiry Key Rotation setup

Overview:

This documentation outlines the steps to:

1. **AWS CloudTrail:** Monitors API calls for creating new access keys and triggers the workflow.
2. **Amazon EventBridge:** Filters the CreateKey event from CloudTrail and forwards it to a Lambda function.
3. **Lambda Functions:**
 - **Creation Lambda:** Stores details about the created keys (source IP address, creation time, CloudTrail event, and access key) in DynamoDB.
 - **Notification Lambda:** Handles email notifications through Amazon SES when keys expire.
4. **Amazon DynamoDB:** Stores the access key details along with Time-to-Live (TTL) for automated expiration.
5. **Amazon Simple Email Service (SES):** Sends email notifications to the user when key expiration is near or triggered.

Architecture Setup



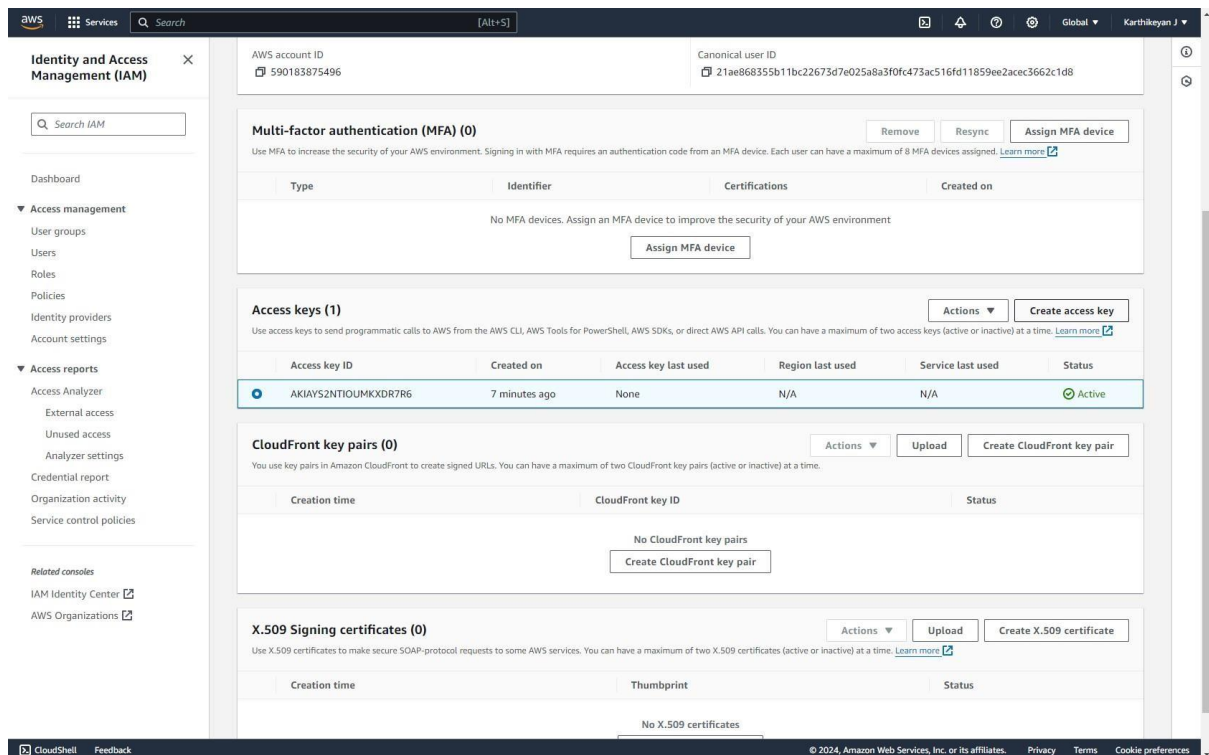
Architecture Explanation

- The architecture leverages AWS CloudTrail to monitor the creation of new IAM access keys. When a key is created, Amazon Eventbridge filters the ``CreateAccessKey`` event and triggers a Lambda function, which stores the access key details—such as the source IP address, creation time, and access key—in Amazon DynamoDB. A Time-to-Live (TTL) attribute is set in DynamoDB to automatically expire the key after 90 days. When the TTL is reached, DynamoDB Streams triggers another Lambda function, which sends an email notification to the user using Amazon SES. This automated process ensures secure key management and timely notifications for key expiration.

Step-by-Step Configuration

1. Create Access and Secret Keys

- Open the AWS Management Console and navigate to IAM (Identity and Access Management).
- Under Users, select an existing user or create a new one.
- Go to the Security credentials tab and click Create access key.
- Save the Access key ID and Secret access key securely, as this will not be available later.
- These credentials will be logged by CloudTrail, and the workflow will begin once they are created.

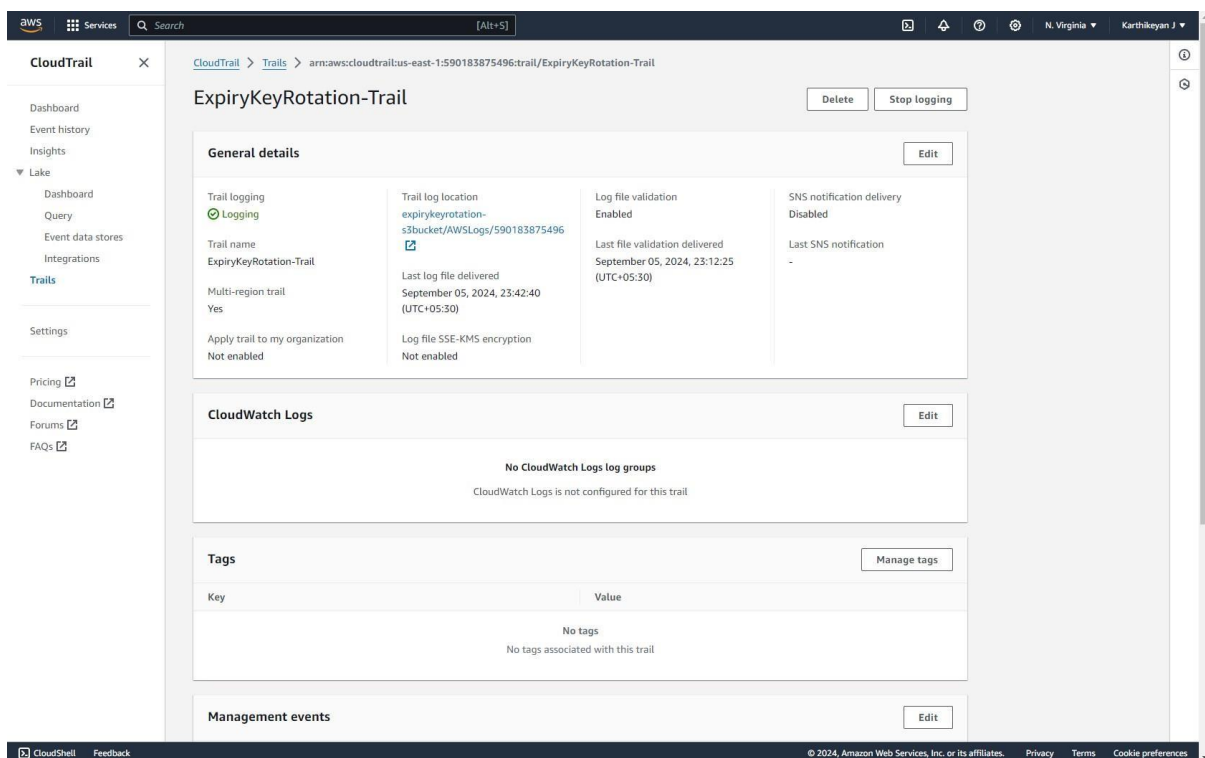


The screenshot displays the AWS IAM console interface. The left-hand navigation pane shows the 'Identity and Access Management (IAM)' section, with 'Users' selected. The main content area is titled 'Security credentials' and shows the 'Access keys' tab. At the top, the 'AWS account ID' (590183875496) and 'Canonical user ID' (21ae868355b11bc22673d7e025a8a3f0fc473ac516fd11859ee2acec3662c1d8) are displayed. Below this, the 'Multi-factor authentication (MFA) (0)' section indicates no MFA devices are assigned. The 'Access keys (1)' section shows a single active access key with ID 'AKIAJ52NTIQUKXDR7R6', created 7 minutes ago. The 'CloudFront key pairs (0)' and 'X.509 Signing certificates (0)' sections both show no items are present. The bottom of the console shows the footer with copyright information for Amazon Web Services, Inc. and links to privacy, terms, and cookie preferences.

Access key ID	Created on	Access key last used	Region last used	Service last used	Status
AKIAJ52NTIQUKXDR7R6	7 minutes ago	None	N/A	N/A	Active

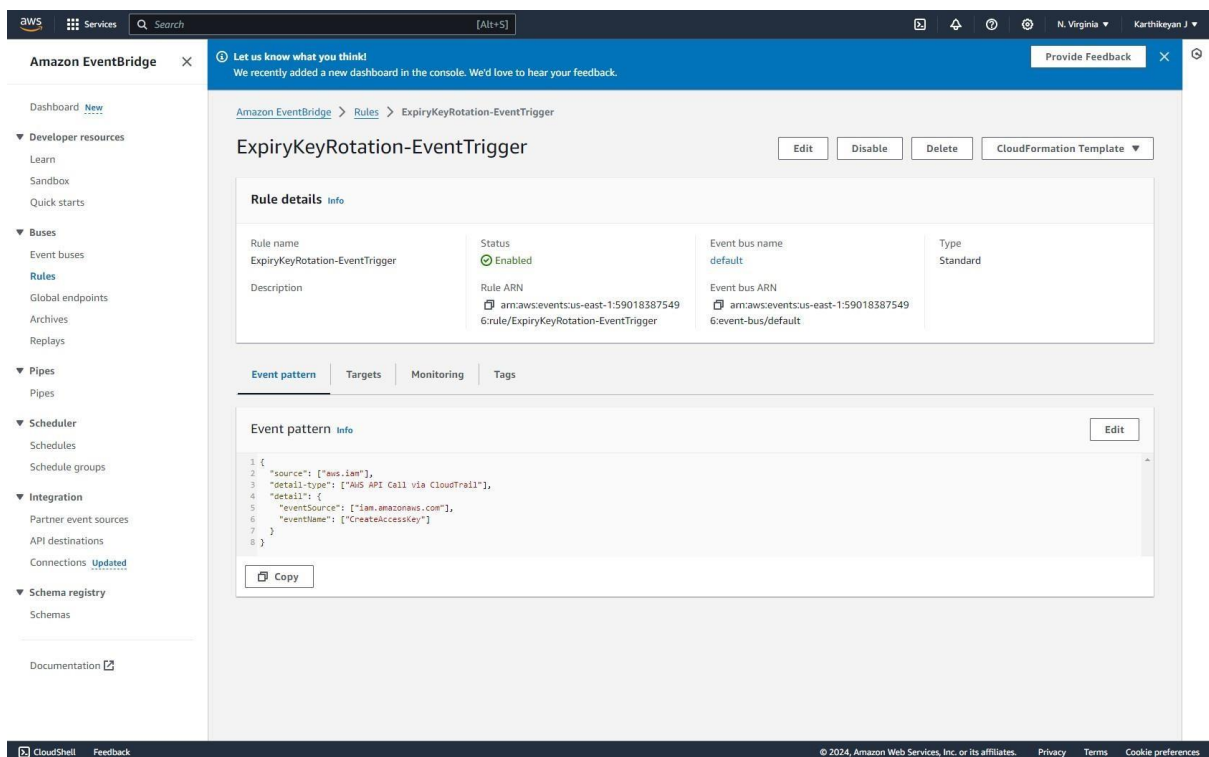
2. Enable CloudTrail for Monitoring API Calls

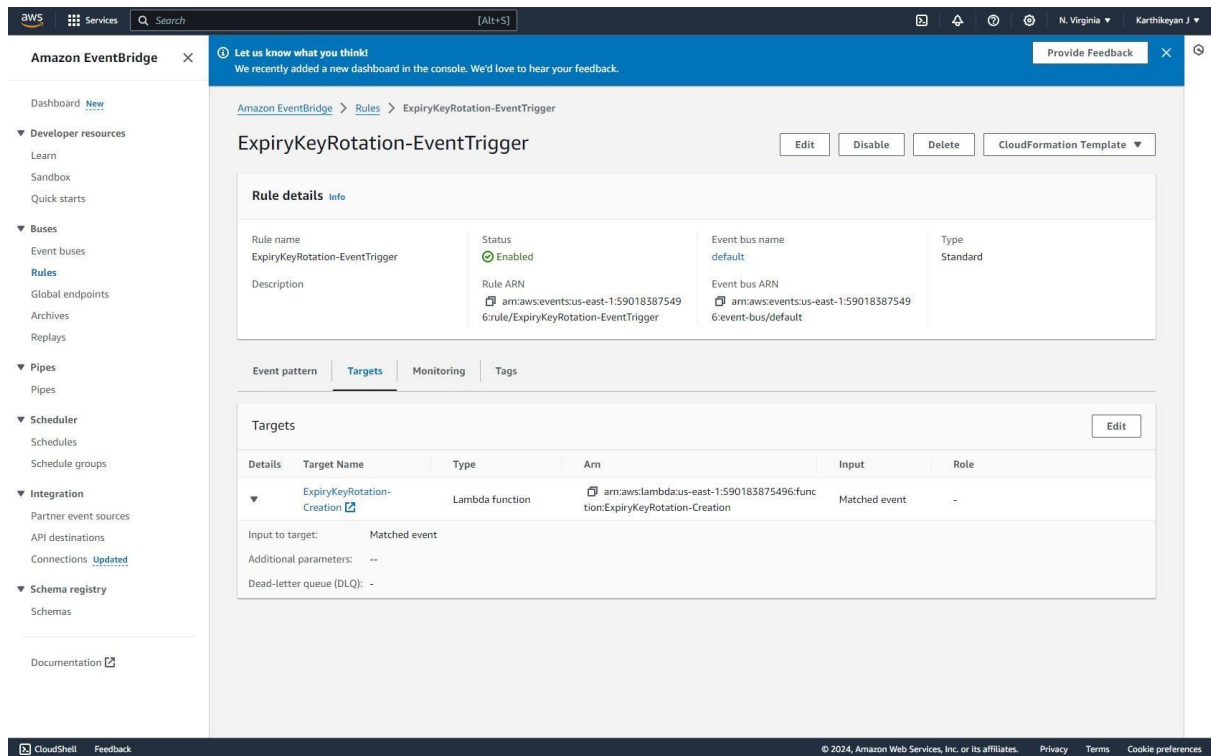
- Open the **AWS Management Console** and navigate to **CloudTrail**.
- Create a new CloudTrail or modify an existing one to capture **CreateKey** events:
 - Go to **Trails** and click **Create trail**.
 - In **Management events**, choose to log **Read/Write events** and select **Write-only** (to capture the creation of keys).
 - Enable logging for **IAM** in **Event type**.
- Set a destination (S3 or CloudWatch Logs) for storing your CloudTrail logs.



3. Set Up Amazon EventBridge for Filtering CreateKey Events

- In the **AWS Console**, navigate to **Amazon EventBridge**.
- Create a **new rule** to capture only the CreateKey events from CloudTrail:
 - Select **Event source** as **AWS services**.
 - In **Event pattern**, choose **IAM** as the service and **CreateAccessKey** as the event name.
- Set the **target** to the Lambda function you will use to process these events (the creation Lambda function).





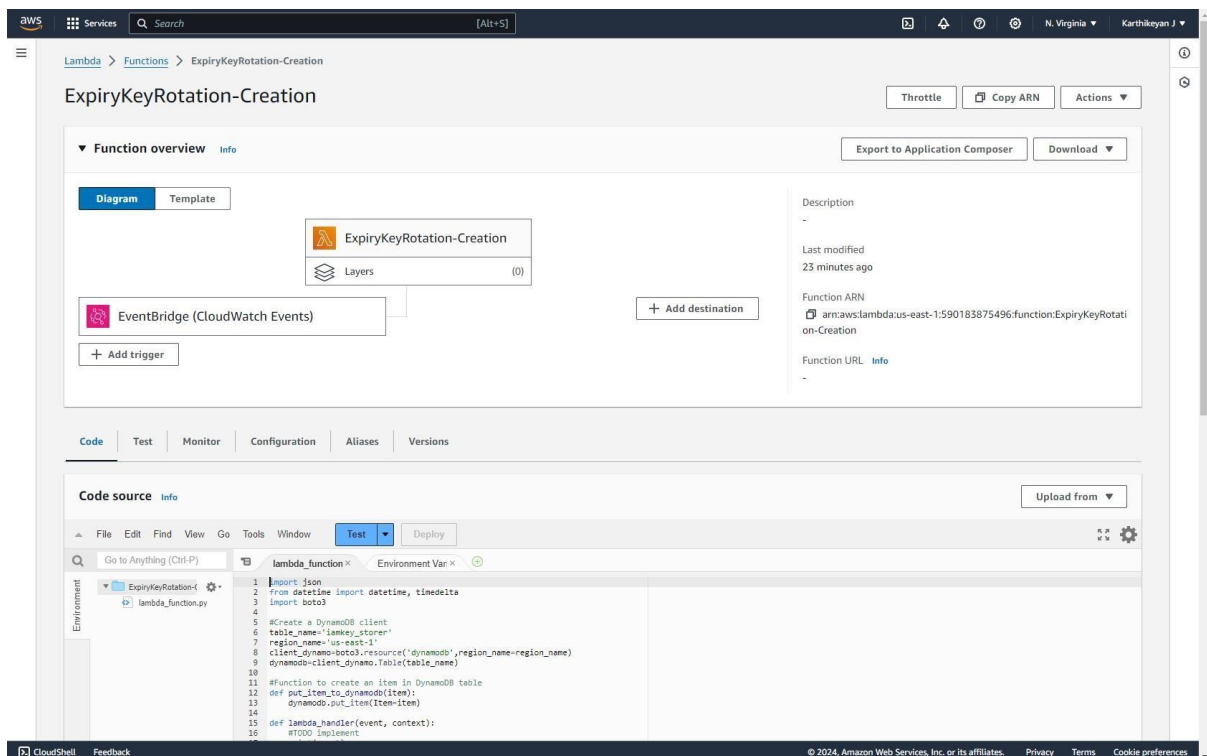
4. Create the Lambda Function to Process Key Creation Events

- In **AWS Lambda**, create a new function:
 - Select **Author from scratch** and give it a name like **ExpiryKeyRotation-Creation**.
 - Set the **runtime** to Python (**Node.js 20.x** or, depending on your code).
- Attach the necessary permissions to allow DynamoDB and CloudTrail interaction.

- Write code to:

1. Extract event data (such as sourceIPAddress, cloudtrail_key, and access_key).
2. Store this data in a DynamoDB table (with attributes for Time-to-Live (TTL)).
3. Example structure of the item stored in DynamoDB:

```
{
  "cloudtrail_key": "abc123",
  "access_key": "AKIA...",
  "created_on": "2024-09-06T12:00:00Z",
  "sourceIPAddress": "192.168.1.1",
  "ttl": "1691434000" // TTL in UNIX timestamp (90 days later)
}
```



aws Services Search [Alt+S]

+ Add trigger

on-Creation
Function URL [info](#)

Code Test Monitor Configuration Aliases Versions

Code source [info](#) Upload from

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl-P)

Environment

ExpiryKeyRotation

lambda_function.py

```
1 import json
2 from datetime import datetime, timedelta
3 import boto3
4
5 #Create a DynamoDB client
6 table_name='iamkey_storer'
7 region_name='us-east-1'
8 client_dynamo=boto3.resource('dynamodb',region_name=region_name)
9 dynamodb_client=dynamo.Table(table_name)
10
11 #Function to create an item in DynamoDB table
12 def put_item_to_dynamodb(item):
13     dynamodb.put_item(Item=item)
14
15 def lambda_handler(event, context):
16     #TODO implement
17     print(event)
18     data_to_insert={}
19     data_to_insert['access_key']=event['detail']['responseElements']['accessKey']['accessKeyId']
20     creation_time_str=event['detail']['responseElements']['accessKey']['createDate']
21     creation_time = datetime.strptime(creation_time_str, '%d %b, %Y %I:%M:%S %p')
22     data_to_insert['created_on'] = creation_time.strftime('%Y-%m-%dT%H:%M:%S')
23     data_to_insert['sourceIpAddress']=event['detail']['sourceIpAddress']
24     data_to_insert['cloudtrail_key']=event['id']
25     put_item_to_dynamodb(data_to_insert)
26
27
28     return {
29         'statusCode': 200,
30         'body': json.dumps('Hello from Lambda!')}
31 }
```

1:1 Python Spaces: 4

Code properties [info](#)

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Services Search [Alt+S]

Function overview [info](#) Export to Application Composer Download

Diagram Template

ExpiryKeyRotation-Creation

Layers (0)

EventBridge (CloudWatch Events)

+ Add trigger

+ Add destination

Description

Last modified
24 minutes ago

Function ARN
arn:aws:lambda:us-east-1:590183875496:function:ExpiryKeyRotation-Creation

Function URL [info](#)

Code Test Monitor Configuration Aliases Versions

General configuration

Triggers

Permissions

Destinations

Function URL

Environment variables

Tags

VPC

RDS databases

Monitoring and operations tools

Concurrency and recursion detection

Triggers (1) [info](#) Fix errors Edit Delete Add trigger

Find triggers

Trigger

EventBridge (CloudWatch Events): ExpiryKeyRotation-EventTrigger

arn:aws:events:us-east-1:590183875496:rule/ExpiryKeyRotation-EventTrigger

Rule state: **ENABLED**

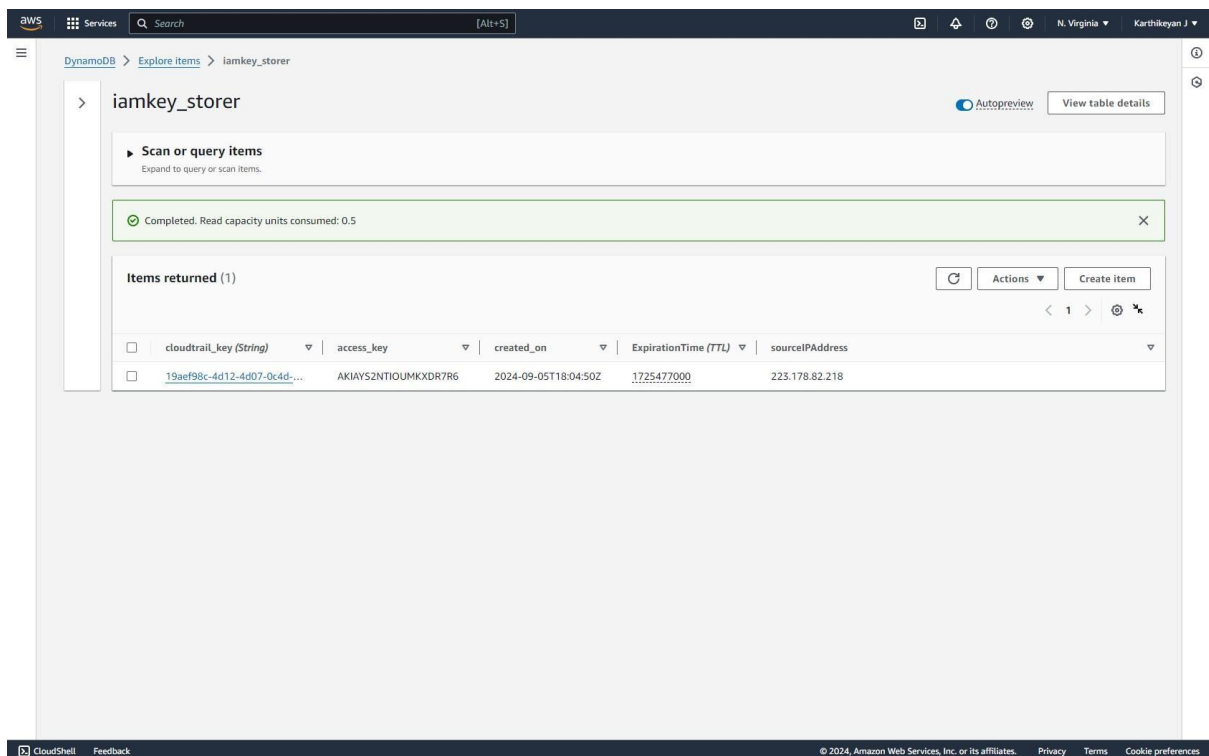
Details

CloudShell Feedback

© 2024, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

5. Set Up Time-to-Live (TTL) in DynamoDB

- Navigate to **DynamoDB** and create a table for storing key information:
 - Set **cloudtrail_key** as the partition key.
- In the table settings, enable **Time-to-Live (TTL)** and configure it to use a dedicated attribute (e.g., ttl).
- This TTL attribute should be a Unix timestamp representing the expiration time (90 days from creation).

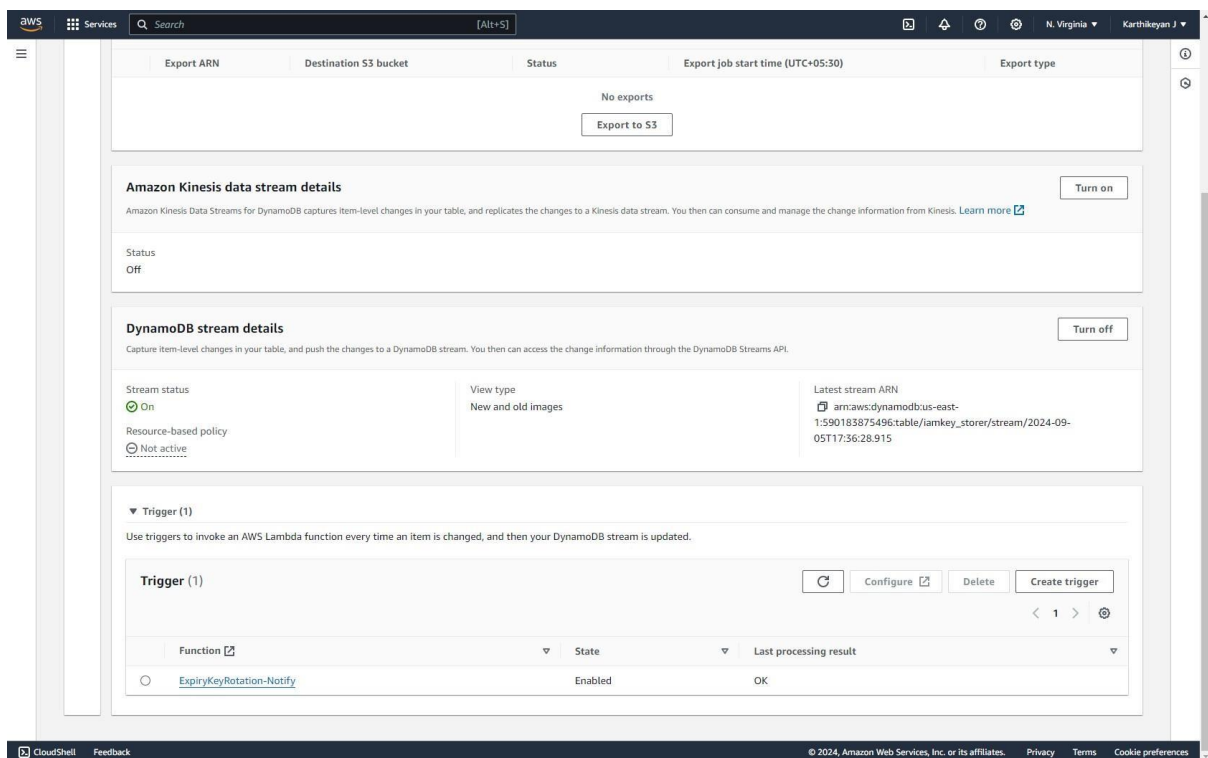


The screenshot displays the AWS Management Console interface for a DynamoDB table named 'iamkey_storer'. The breadcrumb navigation shows 'DynamoDB > Explore items > iamkey_storer'. A green status bar indicates a successful scan: 'Completed. Read capacity units consumed: 0.5'. Below this, a section titled 'Items returned (1)' shows a single item in a table format. The table has columns for 'cloudtrail_key (String)', 'access_key', 'created_on', 'ExpirationTime (TTL)', and 'sourceIPAddress'. The item's values are: '19aef98c-4d12-4d07-0c4d-...', 'AKIAYS2NTIQUUMKXDR7R6', '2024-09-05T18:04:50Z', '1725477000', and '223.178.82.218'. The console footer includes 'CloudShell', 'Feedback', and copyright information for Amazon Web Services, Inc. or its affiliates.

cloudtrail_key (String)	access_key	created_on	ExpirationTime (TTL)	sourceIPAddress
19aef98c-4d12-4d07-0c4d-...	AKIAYS2NTIQUUMKXDR7R6	2024-09-05T18:04:50Z	1725477000	223.178.82.218

6. Enable DynamoDB Streams for Monitoring Changes

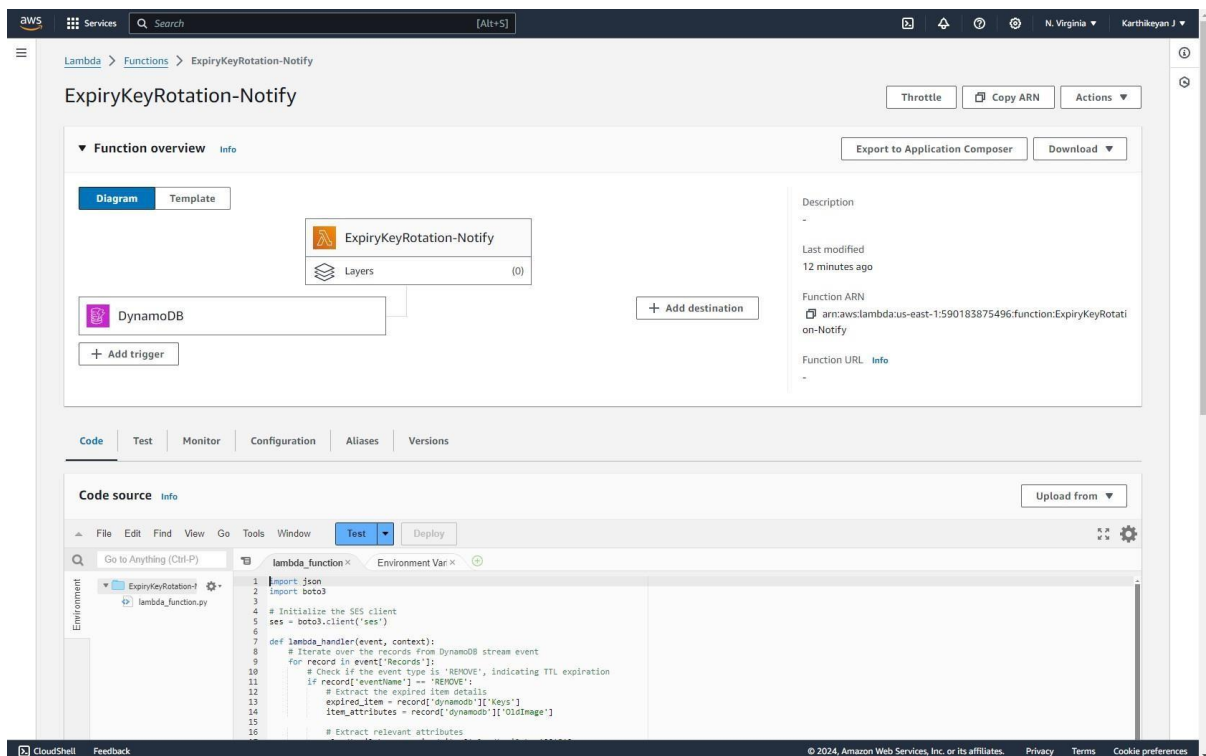
- In the **DynamoDB Console**, enable **DynamoDB Streams** for the table:
 - This will capture any changes to the table, including TTL-triggered deletions.
- Configure a stream ARN and set it as a trigger for a Lambda function that sends an email notification upon deletion (for expired keys).



8. Create the Lambda Function for Email Notification

- Create another Lambda function (ExpiryKeyRotation-Notify):
 - This function will be triggered by DynamoDB Streams when a TTL event occurs (key expiration).
 - Use AWS SDK to send an email via **Amazon SES**:
 - ▢ Configure the SES client to specify the sender and recipient email addresses.
 - ▢ SES must be verified and have appropriate permissions for sending emails.
 - ▢ Example email body:

“Your IAM access Key with ID <access_key> has expired. Please rotate your credential or remove unused keys.”



9. Set Up Amazon SES for Email Notifications

- In **Amazon SES**, verify the sender and recipient email addresses.
- Configure the necessary permissions and policies to allow SES to send emails.
- Link the **Notification Lambda function** to SES for sending the email upon key expiration.

The screenshot shows the Amazon SES console interface. The left sidebar contains navigation links for various services and configurations. The main content area is titled 'Identities' and shows a list of verified identities. Below the identities list, there is a 'Recommendations' section which currently shows 'No recommendations found'.

Identities (2/2)

Identity	Identity type	Identity status
<input checked="" type="checkbox"/> j.karthikeyandev@gmail.com	Email address	Verified
<input checked="" type="checkbox"/> 2116018@saec.ac.in	Email address	Verified

Recommendations (0)

To see high-impact recommendations, select up to 10 identities in the list above and choose Check for recommendations. If you implement a provided solution, refresh the table by selecting Check for recommendations to ensure it's no longer listed. If you enable Virtual Deliverability Manager, this table will refresh automatically for all your identities.

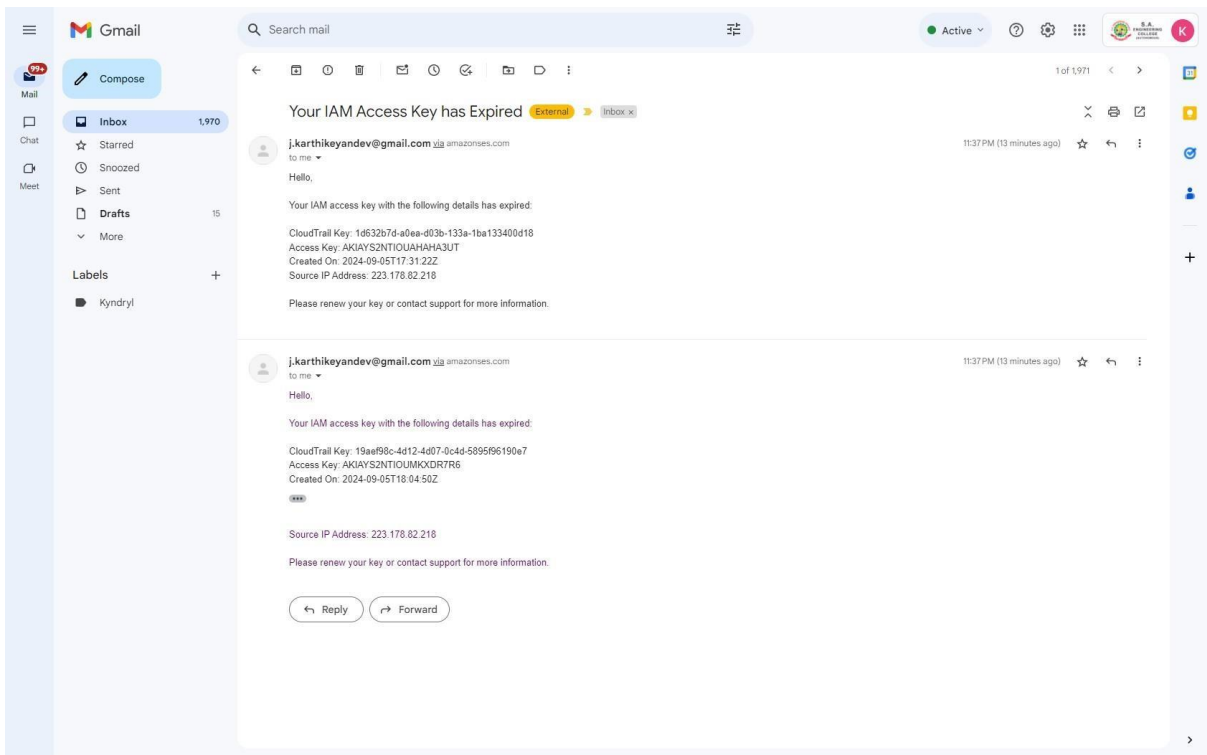
Impact	Identity name	Age	Recommendation/Description	Last checked	Resolve issue
No recommendations found					

Select up to 10 verified domain identities and click Check for recommendations on the Identities table above.

[Enable Virtual Deliverability Manager to generate recommendations automatically](#)

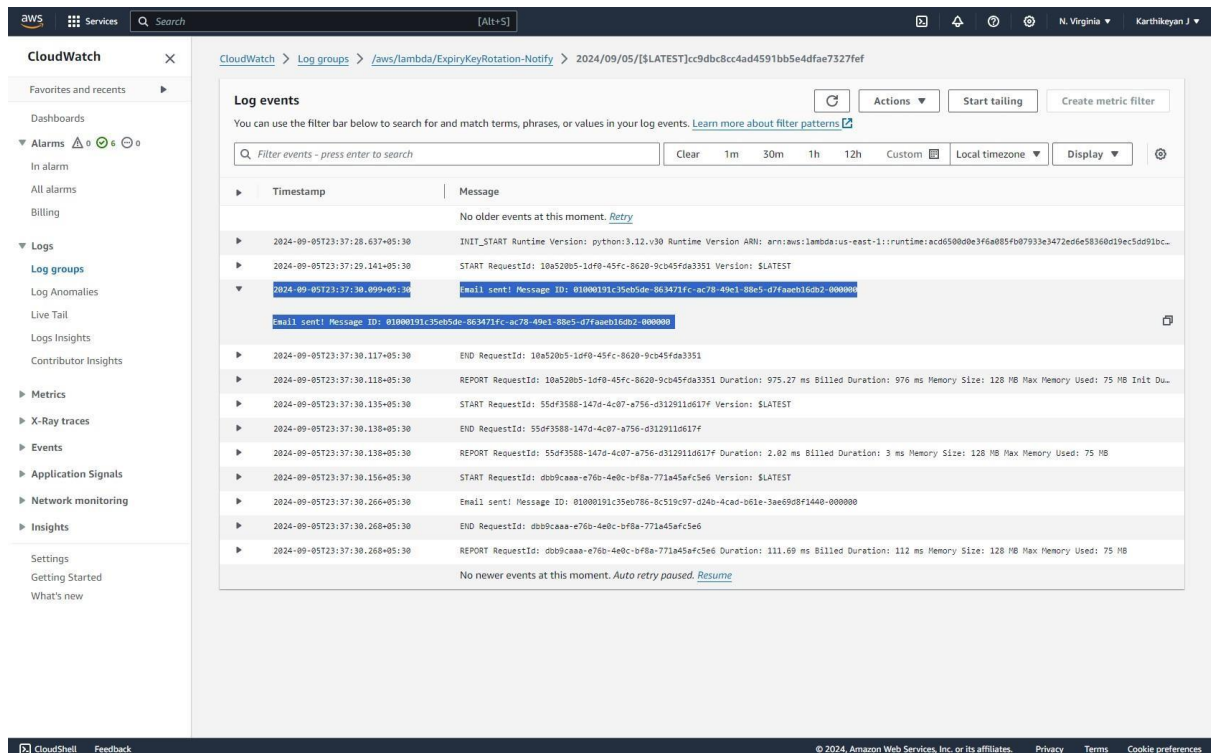
10. Test the Setup

- Test the flow by creating a new access key using AWS CLI or Console.
- Verify that the CloudTrail logs the event, EventBridge triggers the Lambda function, and the key is stored in DynamoDB with the TTL attribute.
- After key expiration (or by manually updating the TTL for testing), confirm that SES sends the expiration email.



11. Security and Permissions

- Ensure all necessary IAM roles and policies are in place for the Lambda functions to interact with CloudTrail, DynamoDB, and SES.
- Use **AWS Key Management Service (KMS)** if you require encryption for sensitive data like access keys stored in DynamoDB.



Conclusion

This setup ensures that IAM access keys are automatically rotated every 90 days. When a key is created, it is logged via CloudTrail and processed through EventBridge to a Lambda function that stores the details in DynamoDB. A Time-to-Live (TTL) attribute is added to automatically expire the keys after 90 days, triggering another Lambda function that sends a notification via SES when the key expires.

1.ExpiryKeyRotation-Creation (Lambda Function Code):

```
import json

from datetime import datetime, timedelta

import boto3

#Create a DynamoDB client

table_name='iamkey_storer'

region_name='us-east-1'

client_dynamo=boto3.resource('dynamodb',region_name=region_name)

dynamodb=client_dynamo.Table(table_name)


#Function to create an item in DynamoDB table

def put_item_to_dynamodb(item):

    dynamodb.put_item(Item=item)


def lambda_handler(event, context):

    #TODO implement

    print(event)

    data_to_insert={}

    data_to_insert['access_key']=event['detail']['responseElements']['accessKey']['accessKeyId']

    creation_time_str=event['detail']['responseElements']['accessKey']['createDate']

    creation_time = datetime.strptime(creation_time_str, '%b %d, %Y %l:%M:%S %p')

    data_to_insert['created_on'] = creation_time.strftime('%Y-%m-%dT%H:%M:%SZ')

    data_to_insert['sourceIPAddress']=event['detail']['sourceIPAddress']

    data_to_insert['cloudtrail_key']=event['id']


    put_item_to_dynamodb(data_to_insert)


    return {

        'statusCode': 200,

        'body': json.dumps('Hello from Lambda!')

    }
```


2.ExpiryKeyRotation-Notify (Lambda Function Code):

```
import json

import boto3

# Initialize the SES client

ses = boto3.client('ses')


def lambda_handler(event, context):

    # Iterate over the records from DynamoDB stream event

    for record in event['Records']:

        # Check if the event type is 'REMOVE', indicating TTL expiration

        if record['eventName'] == 'REMOVE':

            # Extract the expired item details

            expired_item = record['dynamodb']['Keys']

            item_attributes = record['dynamodb']['OldImage']

            # Extract relevant attributes

            cloudtrail_key = expired_item['cloudtrail_key']['S']

            source_ipaddress = item_attributes['source_ipaddress']['S']

            current_time = item_attributes['current_time']['S']

            # Create the email content

            subject = "Your IAM Access Key has Expired"

            body_text = (

                f"Hello,\n\nYour IAM access key with the following details has expired:\n\n"

                f"CloudTrail Key: {cloudtrail_key}\n"
```

```

f"Source IP Address: {source_ipaddress}\n"

f"Expiration Time: {current_time}\n\n"

f"Please renew your key or contact support for more information."

)

# Define SES email parameters

response = ses.send_email(

    Source='your-verified-email@example.com', # Replace with your verified sender email

    Destination={

        'ToAddresses': ['j.karthikeyandev@gmail.com'], # Replace with recipient email

    },

    Message={

        'Subject': {

            'Data': subject,

            'Charset': 'UTF-8'

        },

        'Body': {

            'Text': {

                'Data': body_text,

                'Charset': 'UTF-8'

            }

        }

    }

)

```

```
# Log the email response

print(f"Email sent! Message ID: {response['MessageId']}")


return {

    'statusCode': 200,

    'body': json.dumps('Lambda function executed successfully!')

}
```