**Task 1 and Task 2**

To set up an alert for Amazon RDS (Relational Database Service) instances when their capacity reaches 60%, you can still use Amazon CloudWatch in combination with Amazon SNS. Here are the steps:

### **Step 1: Create a CloudWatch Alarm for RDS**

1. **Log in to the AWS Management Console:** Go to [AWS Management Console](https://aws.amazon.com/console/).
2. **Navigate to CloudWatch:** In the AWS Management Console, type "CloudWatch" in the search bar and select CloudWatch.
3. **Create a New Alarm:**
   * In the CloudWatch dashboard, select "Alarms" from the left-hand menu.
   * Click on "Create Alarm".
4. **Select a Metric:**
   * Click on the "Select metric" button.
   * Choose "RDS" > "Per-Database Metrics".
   * Select the appropriate metric that indicates the capacity usage. Common metrics for capacity include:
     + **CPUUtilization**: Percentage of CPU utilization.
     + **FreeableMemory**: Amount of available memory.
     + **FreeStorageSpace**: Amount of available storage space.
   * Choose the specific RDS instance you want to monitor.
5. **Define the Alarm:**
   * Set the threshold type (e.g., "Static" or "Anomaly Detection").
   * For "Static", specify the threshold value (e.g., CPUUtilization > 60%).
   * Configure the period and evaluation settings as per your requirement.
6. **Configure Actions:**
   * Click on "Next" to configure actions.
   * Choose "In alarm" state.
   * Select "Send a notification to an SNS topic".

### **Step 2: Create an SNS Topic for Notifications**

1. **Create SNS Topic:**
   * If you don't have an existing SNS topic, create a new one by navigating to the SNS service in the AWS Management Console.
   * Click on "Create topic".
   * Choose the type of topic (Standard or FIFO), provide a name, and create the topic.
2. **Subscribe to the SNS Topic:**
   * Once the topic is created, click on the topic name.
   * Under the "Subscriptions" tab, click on "Create subscription".
   * Specify the protocol (e.g., "Email" or "HTTPS" for Mattermost).
   * For email, enter the email address you want to receive notifications at.
   * For Mattermost, configure a webhook URL if you want to use an HTTPS endpoint.
3. **Confirm the Subscription:**
   * For email, a confirmation email will be sent to the provided email address. Confirm the subscription by clicking on the link in the email.

### **Step 3: Link CloudWatch Alarm to SNS Topic**

1. **Link SNS Topic to Alarm:**
   * Go back to the CloudWatch alarm configuration and select the SNS topic created in the previous step.
   * Complete the alarm setup by clicking "Create alarm".

### **Configuring Mattermost for Notifications (if applicable)**

1. **Create a Webhook in Mattermost:**
   * In your Mattermost instance, navigate to the Integrations settings and create an incoming webhook.
   * Copy the webhook URL provided by Mattermost.
2. **Configure SNS Subscription:**
   * In the SNS topic, create a new subscription with the protocol set to "HTTPS" and the endpoint as the Mattermost webhook URL.

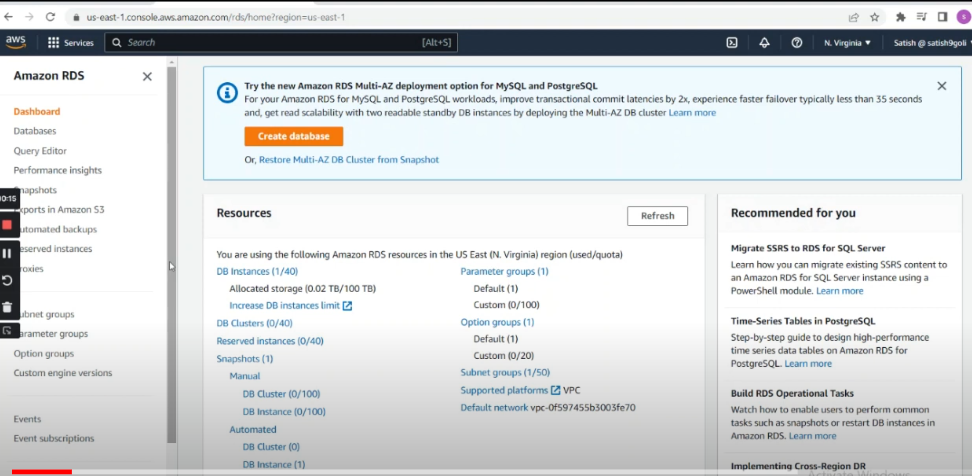
### **Summary**

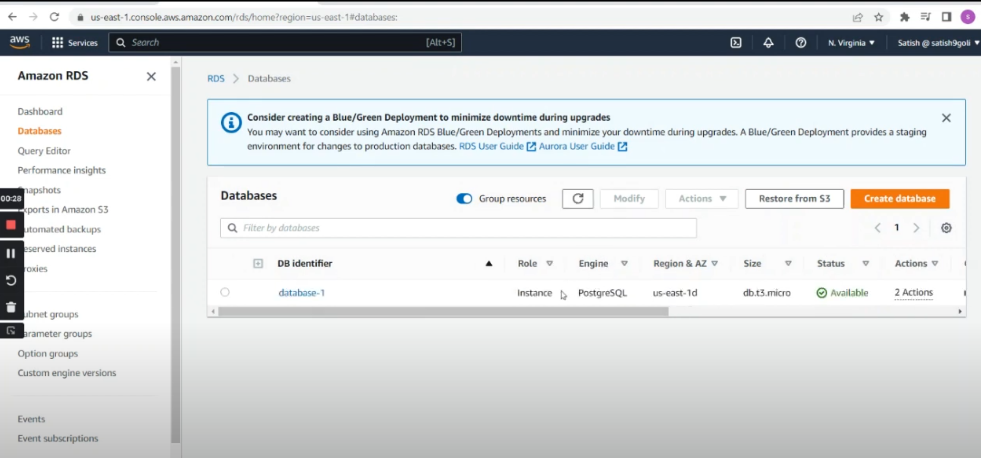
* **CloudWatch Alarm**: Monitor the RDS instance capacity (e.g., CPU utilization, Freeable Memory, Free Storage Space) and trigger an alarm when it exceeds 60%.
* **SNS Topic**: Create an SNS topic to manage notifications.
* **Email or Mattermost**: Subscribe to the SNS topic using your preferred notification method (email or Mattermost webhook).

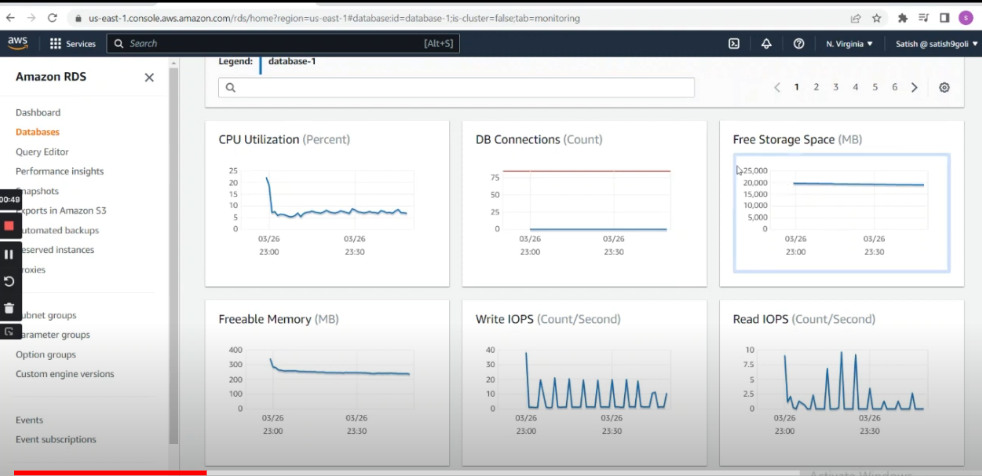
Here is a youTube link for better understanding.

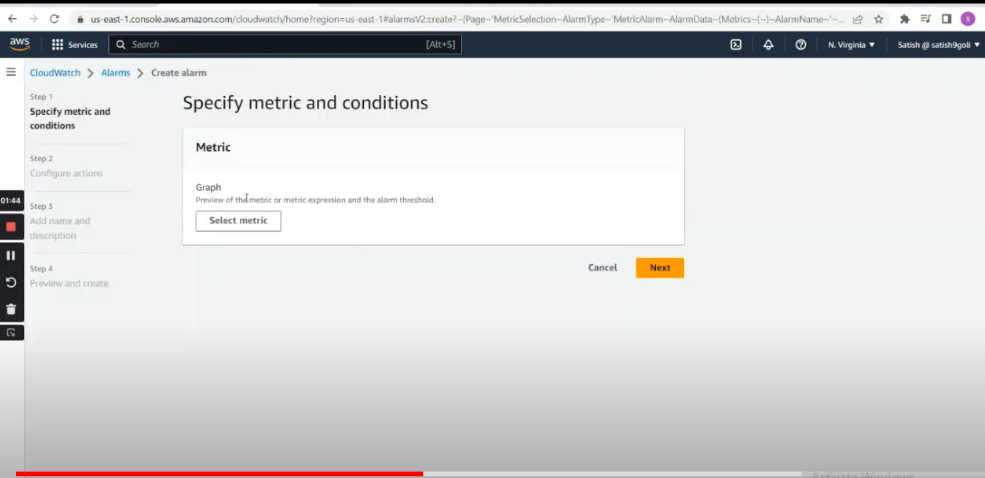
<https://www.youtube.com/watch?v=Yxl7e88cTAQ>

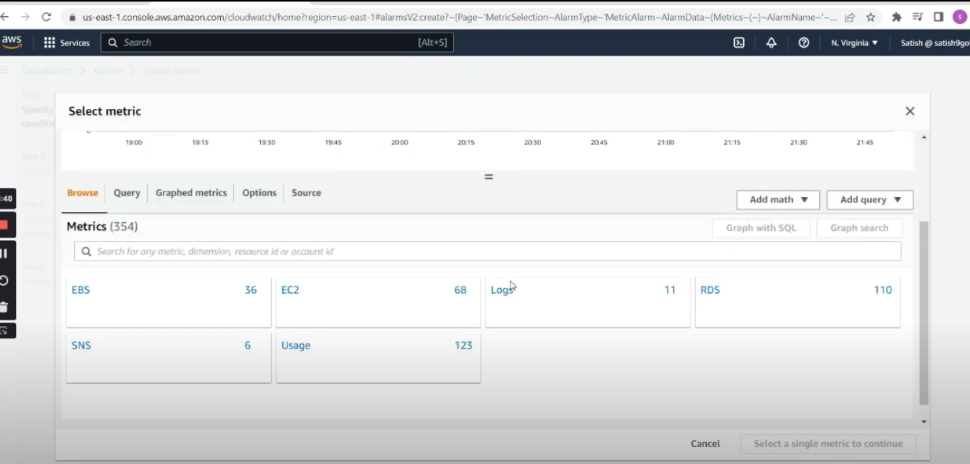
Here are some labs that I performed:

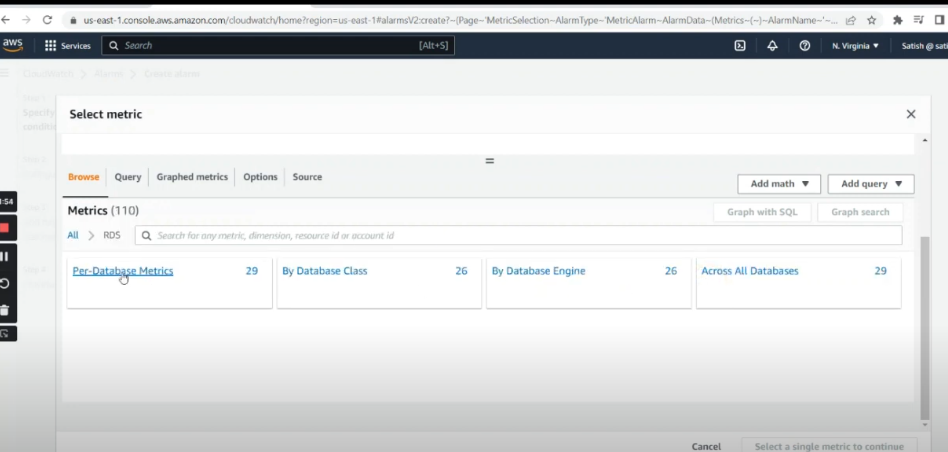


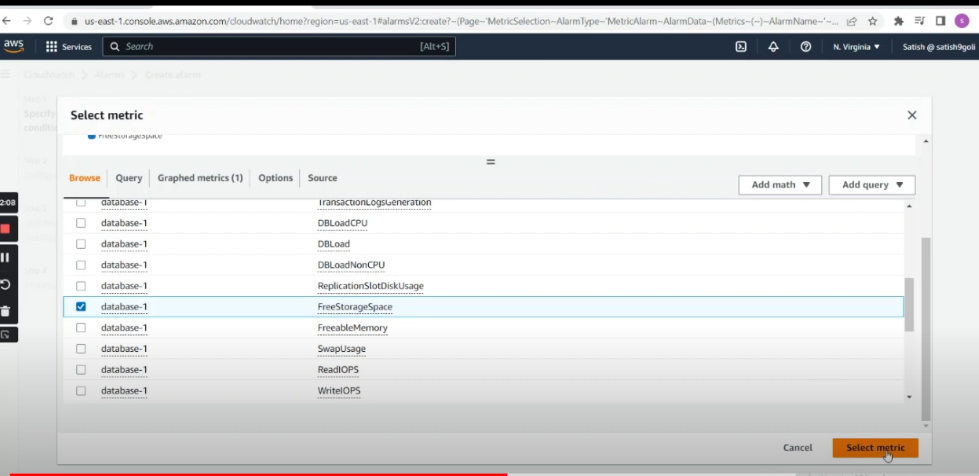


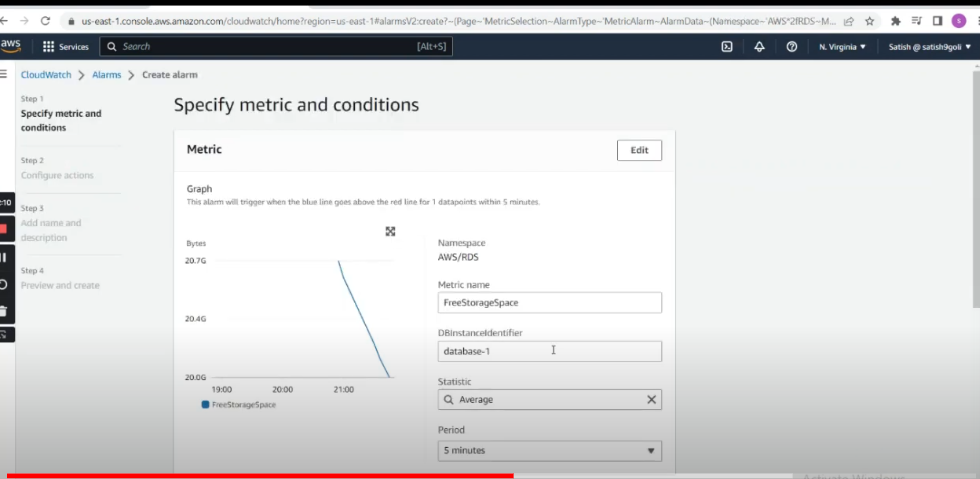


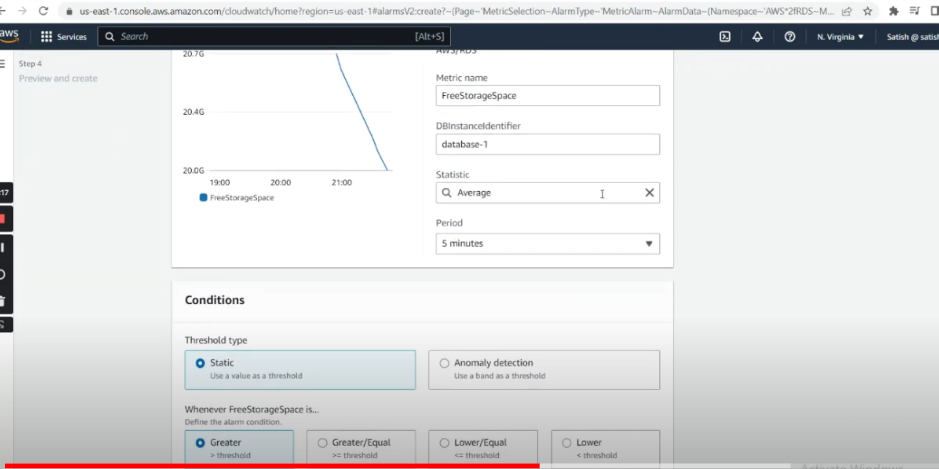


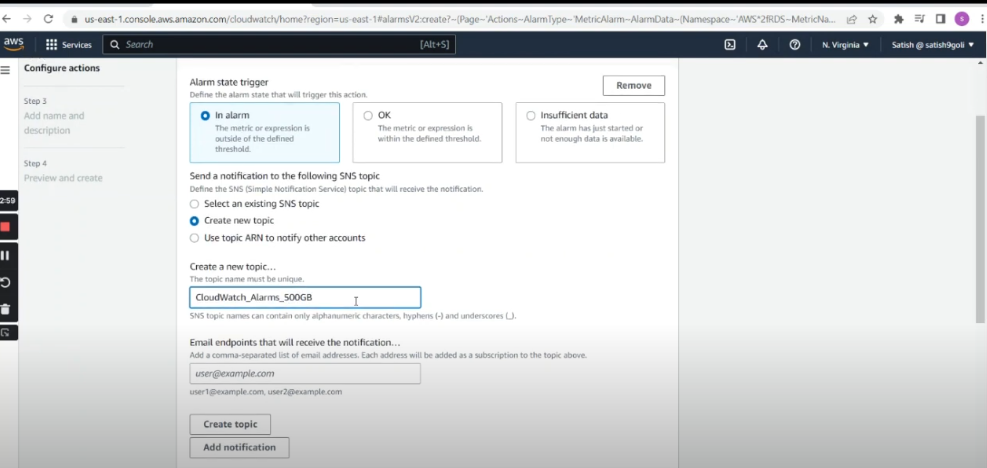


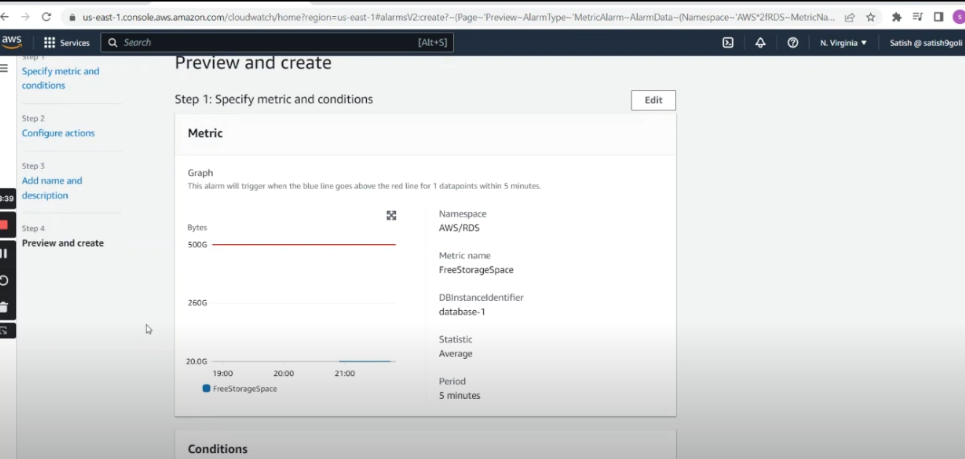


We can select CPU utilization here : 



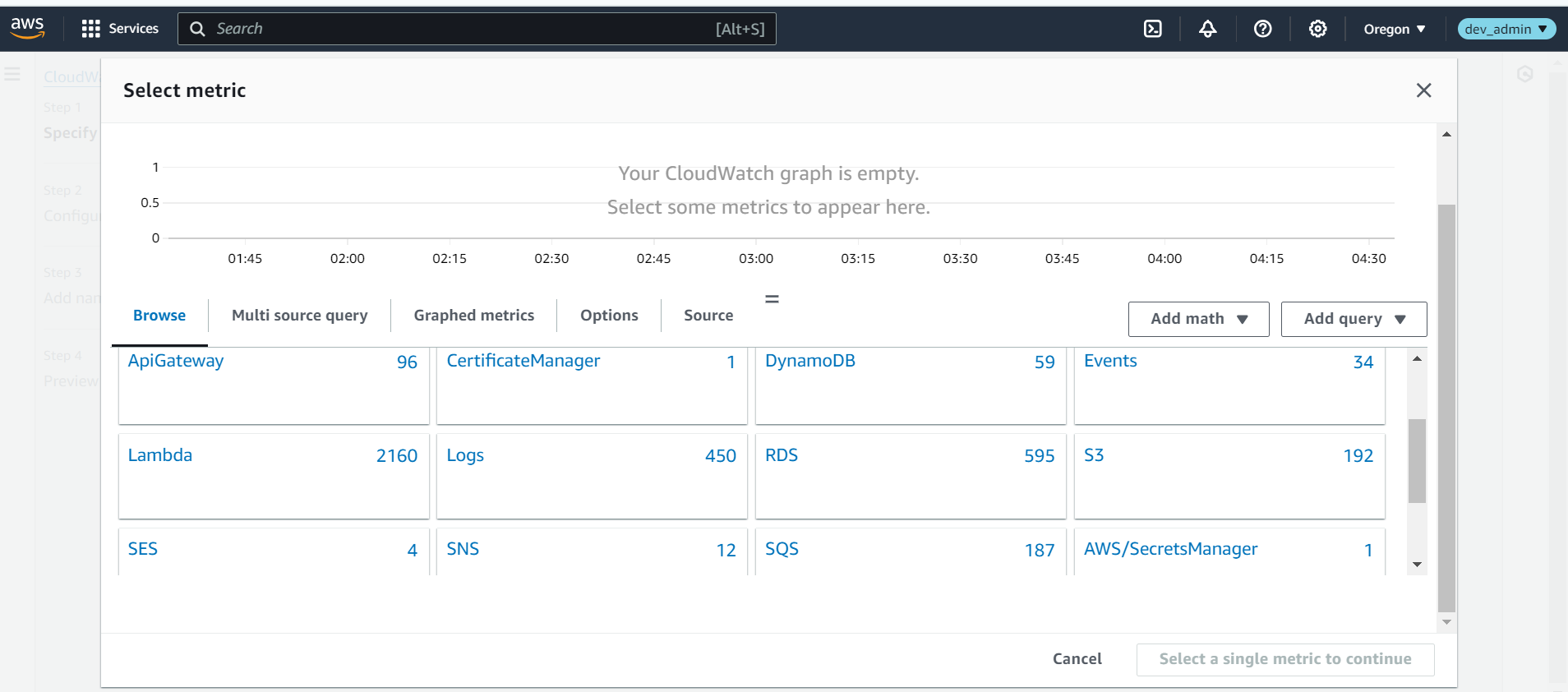




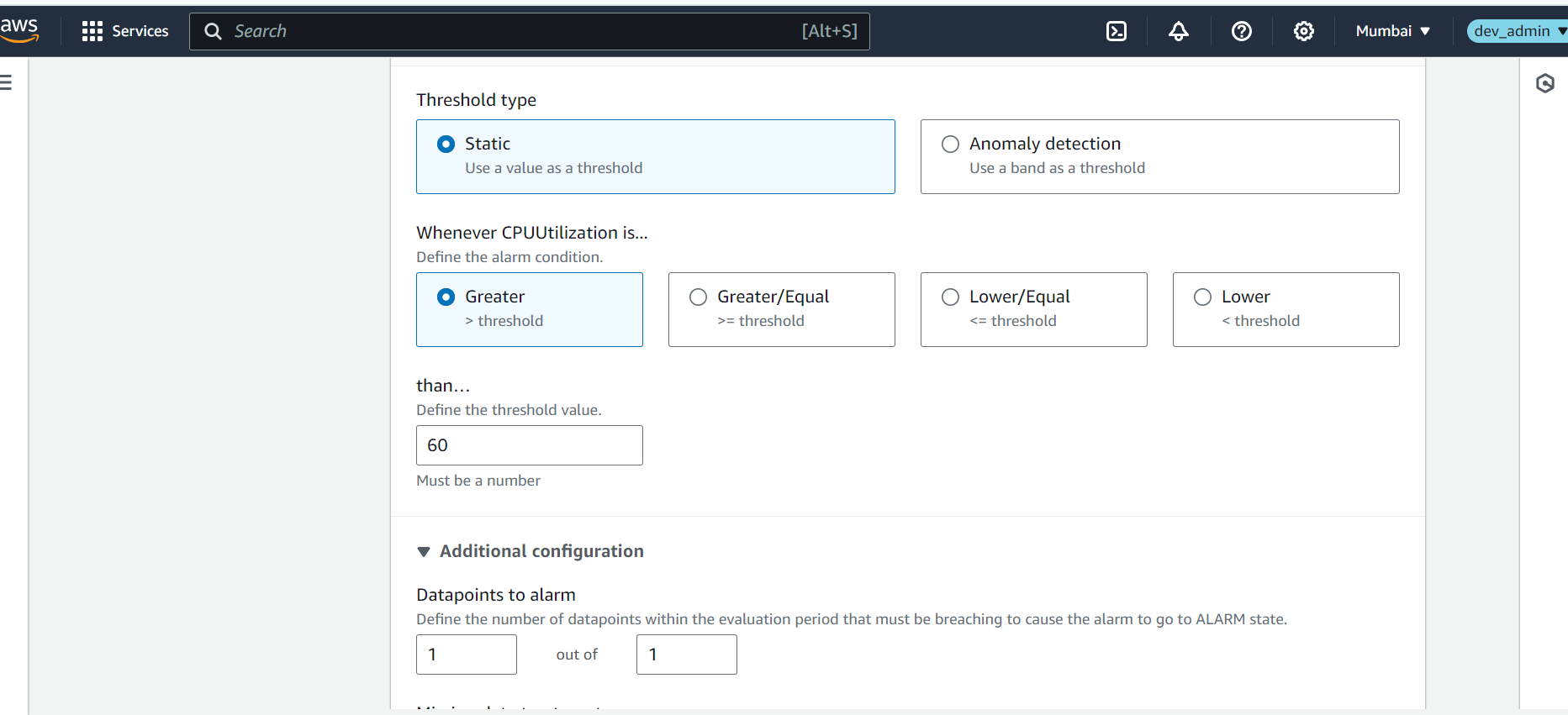


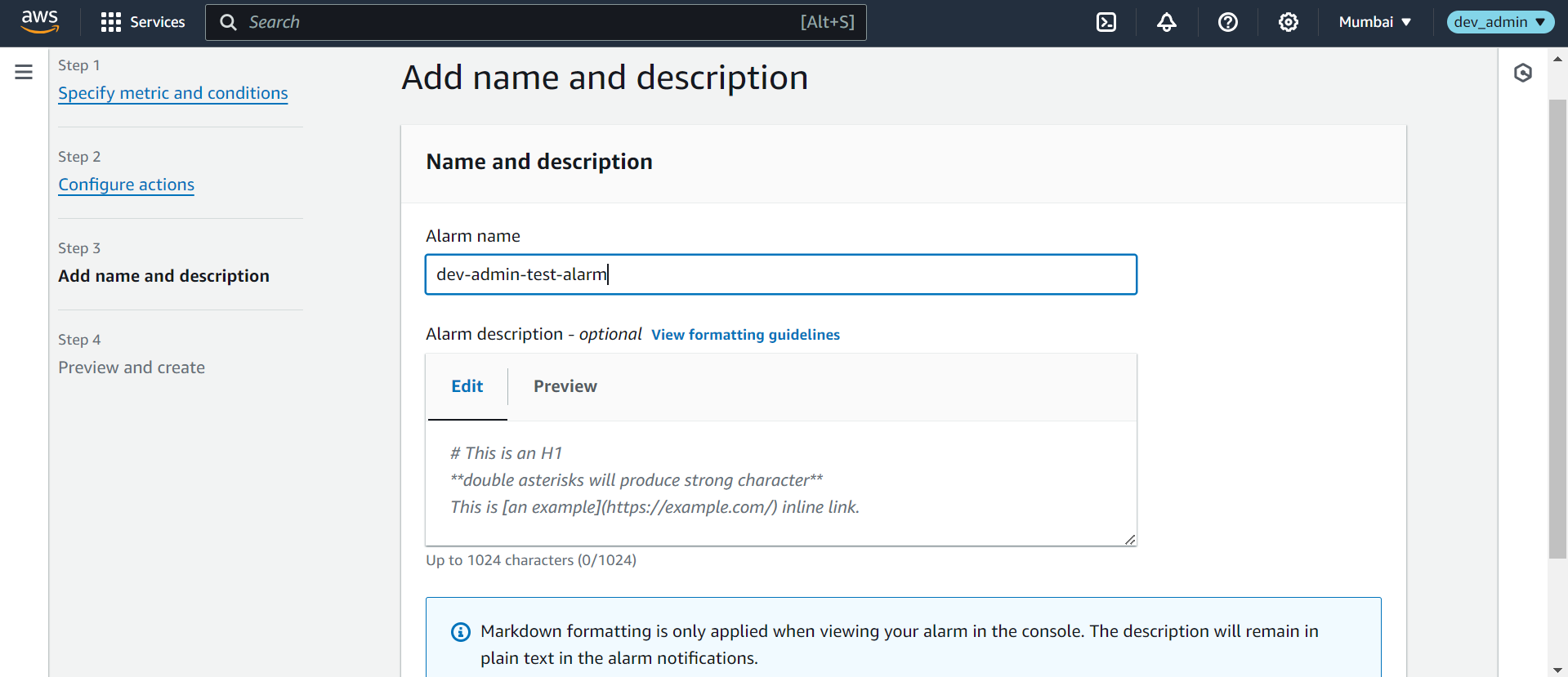
<https://www.youtube.com/watch?v=HrX4TykXT9w>

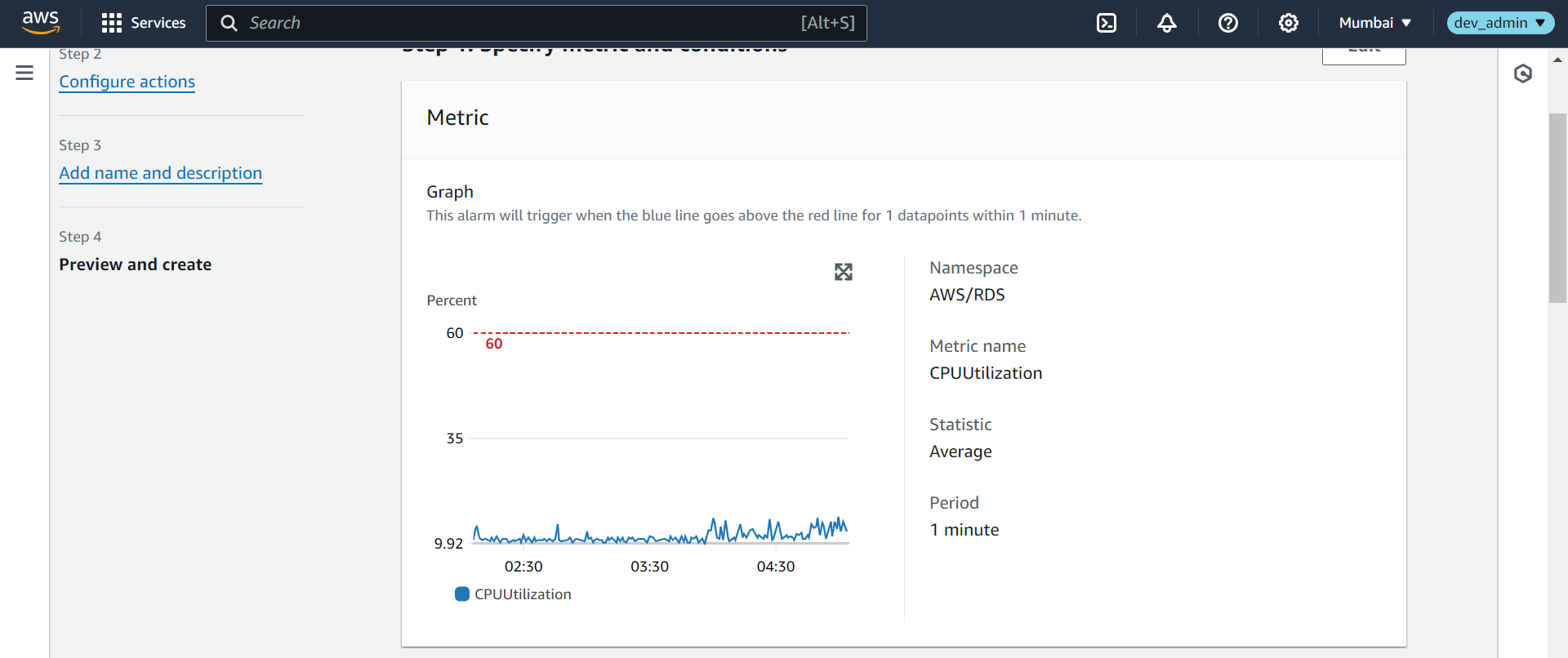
**Task 1 demo on dev-admin account**

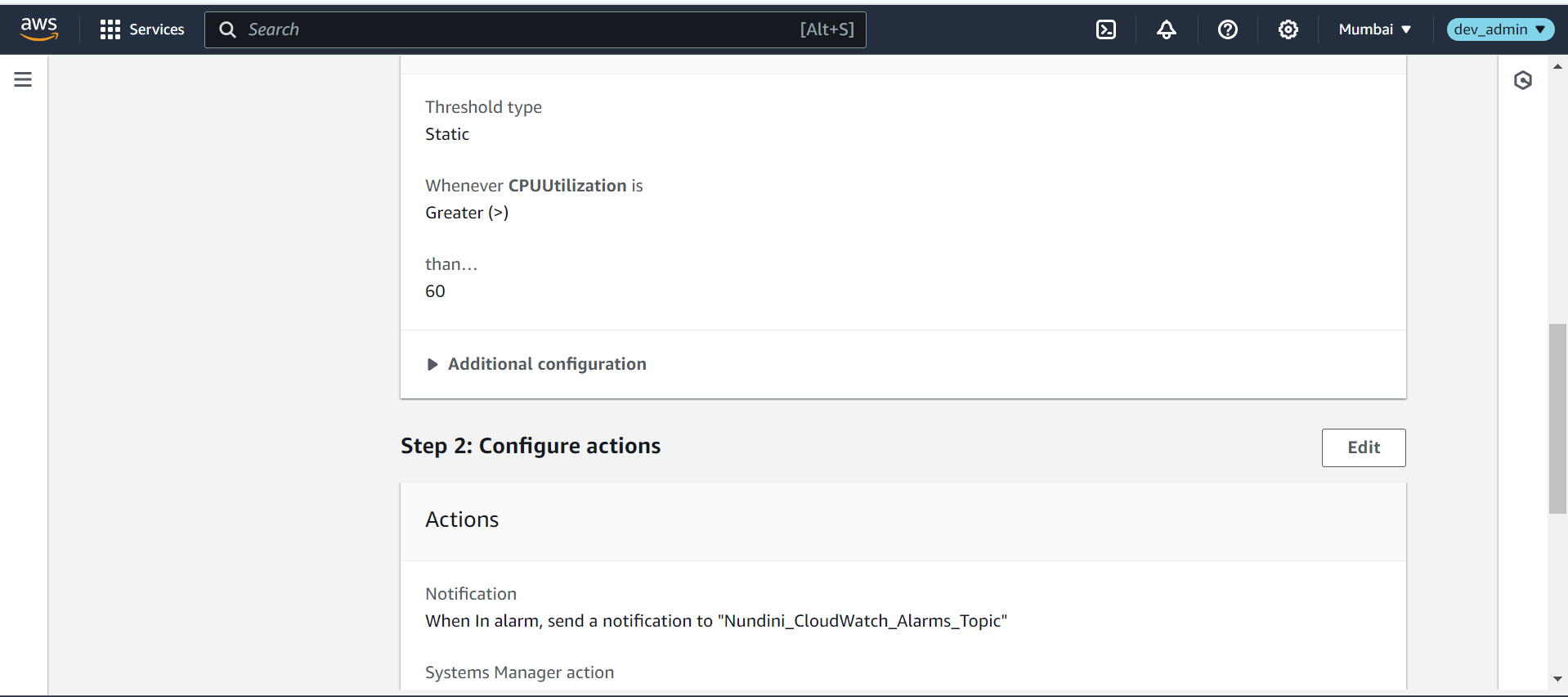


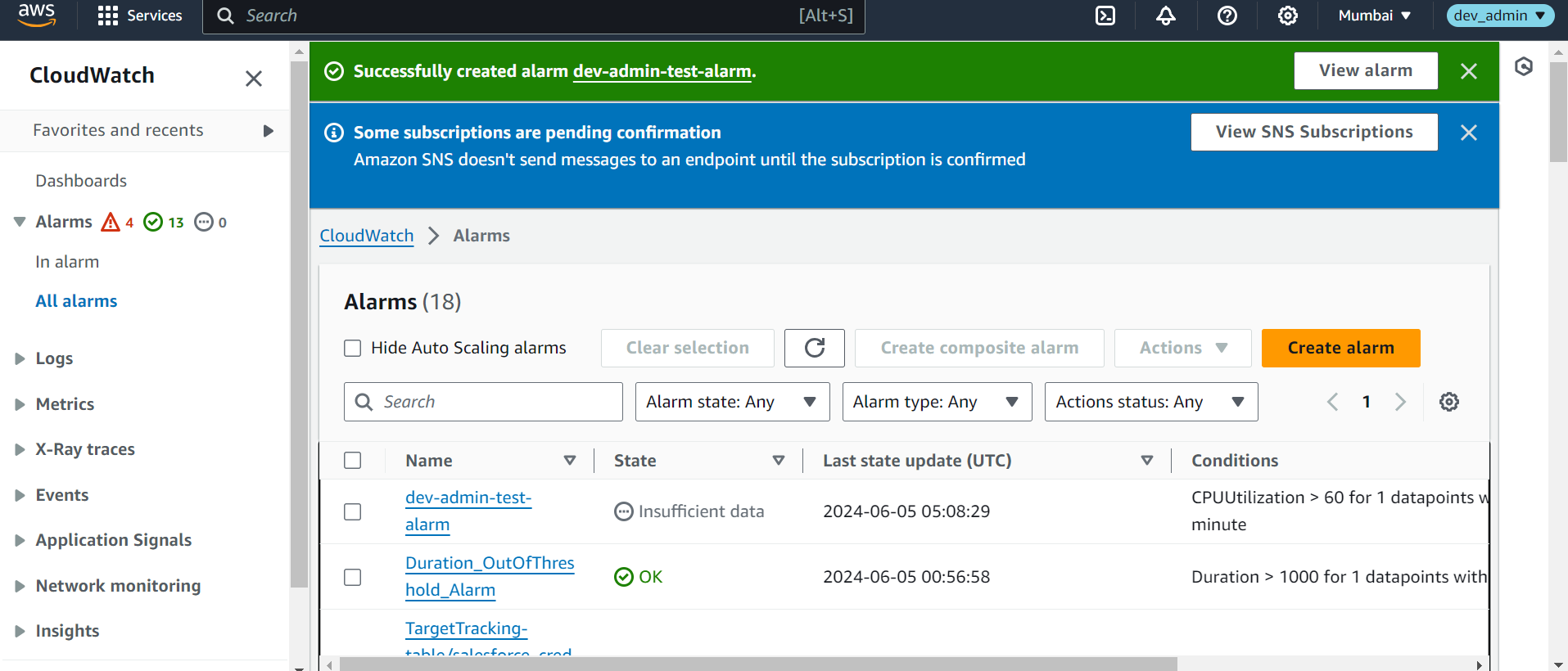


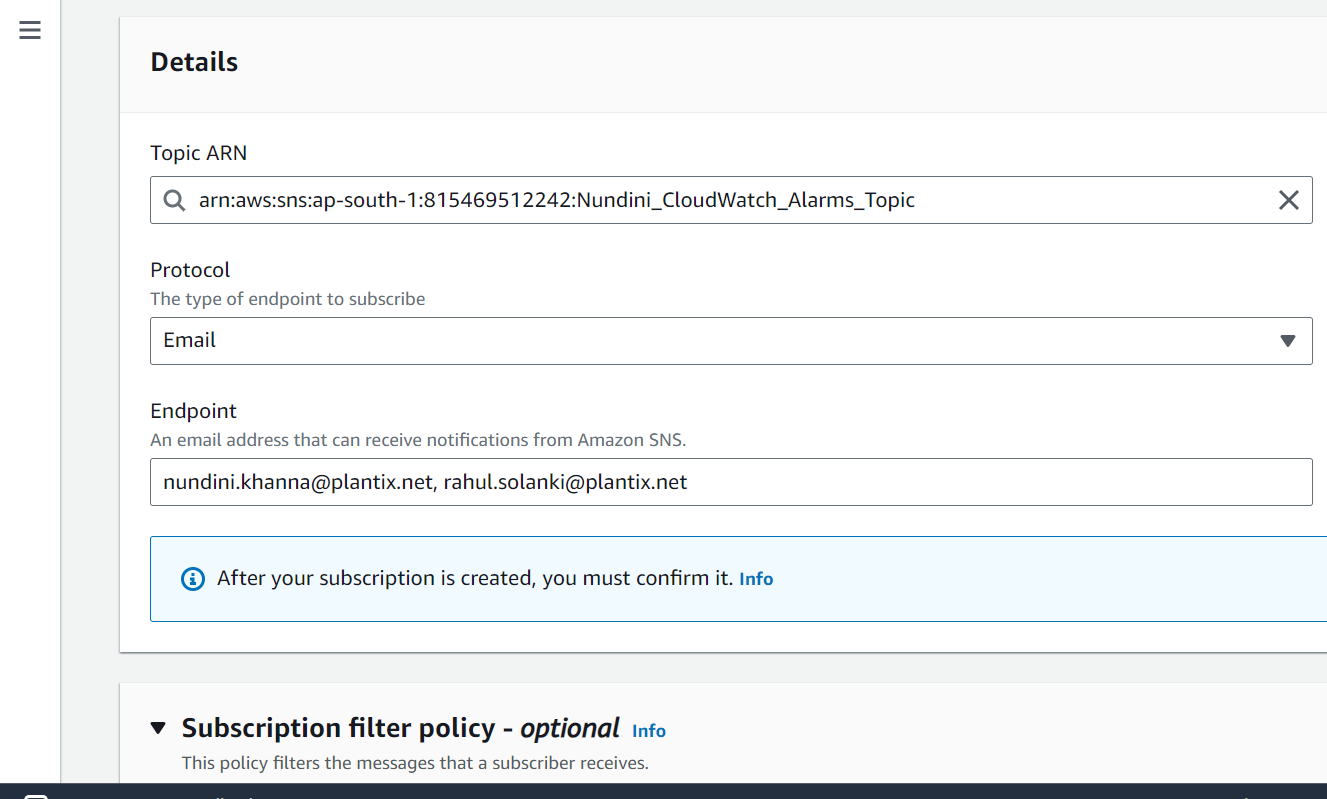


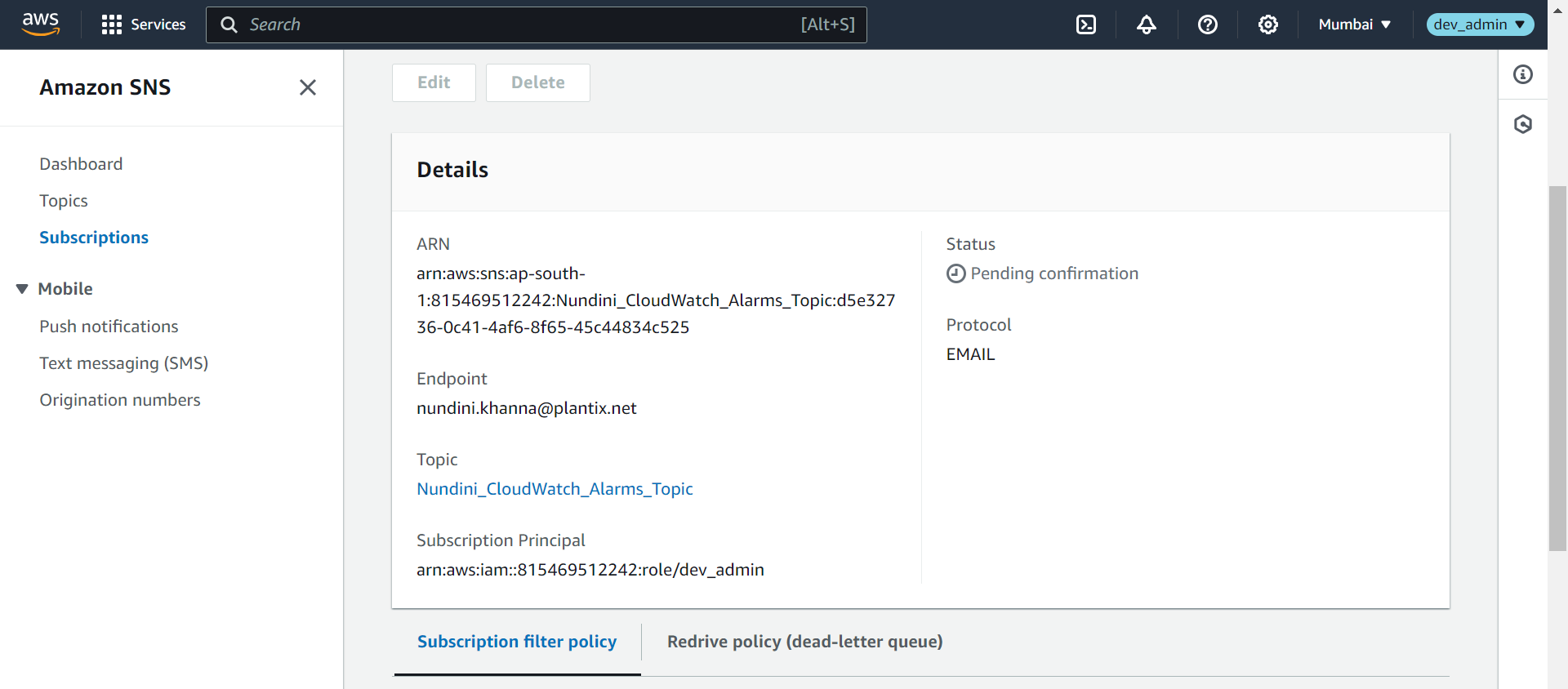


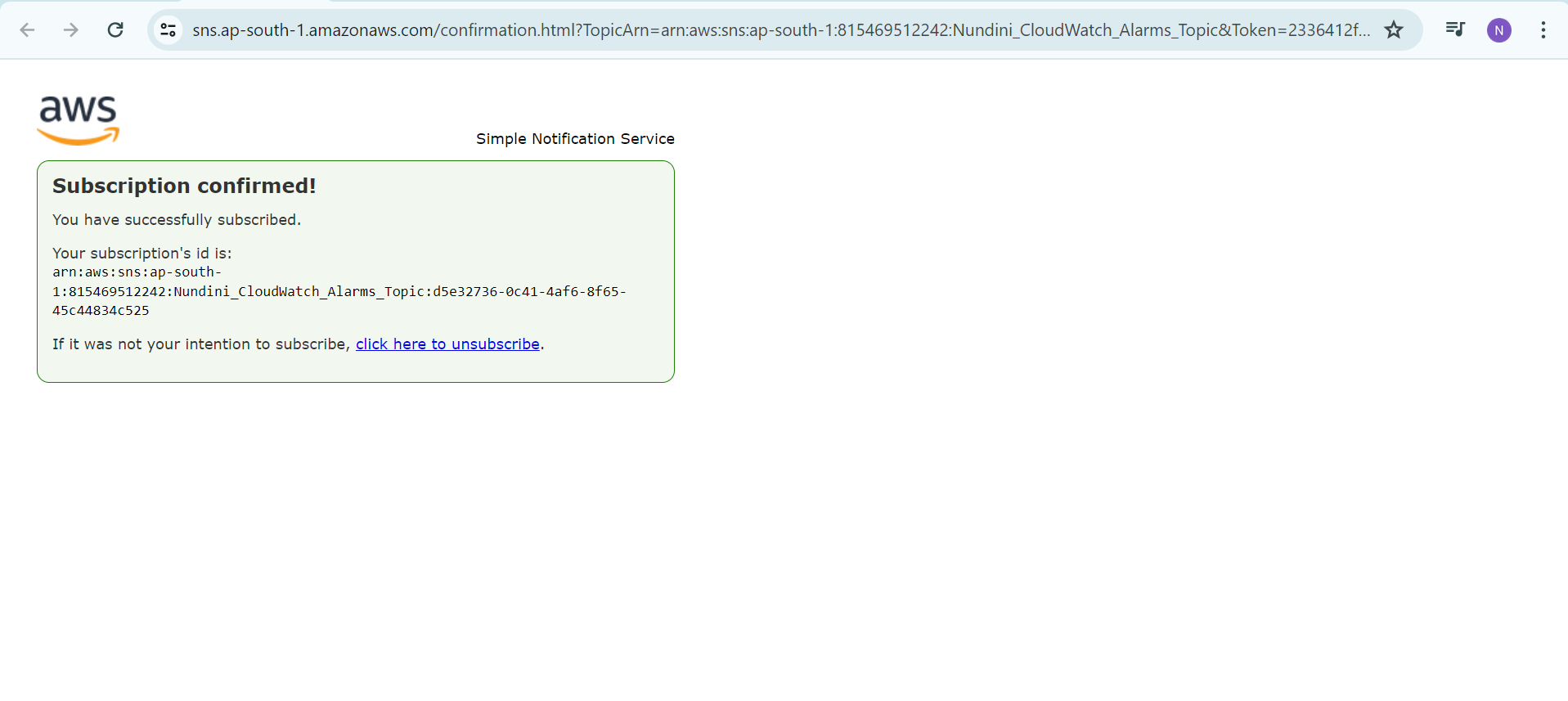


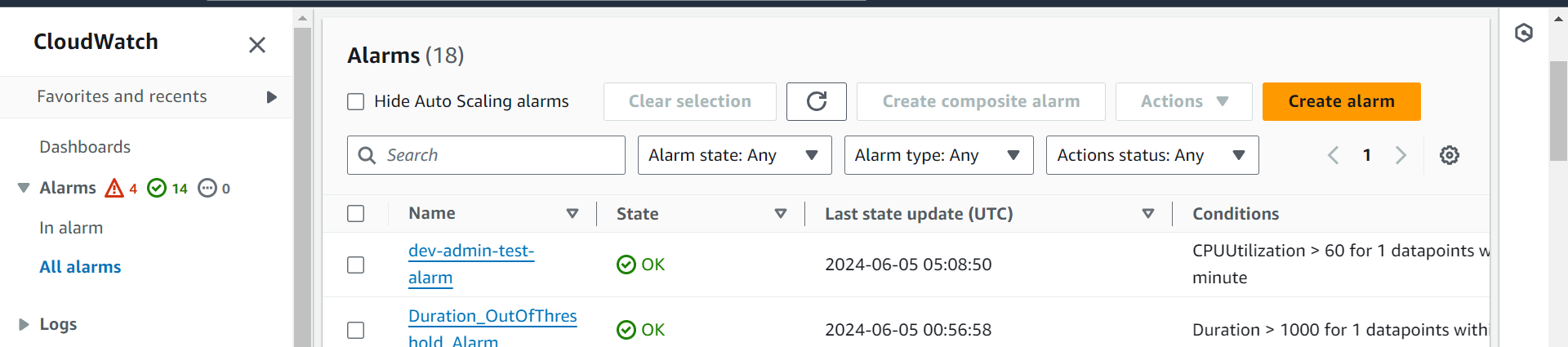












**Task 2**

Implementing auto-rotation for RDS credentials involves a few steps. We'll use AWS Secrets Manager for this purpose because it has built-in support for secrets rotation. The steps are:

1. **Create a secret in AWS Secrets Manager for your RDS credentials.**
2. **Configure rotation for the secret.**
3. **Set up a Lambda function to handle the rotation logic.**
4. **Update the credentials in your RDS instance.**
5. **Send the new credentials to a specified list of email addresses.**

### **Step 1: Create a Secret in AWS Secrets Manager**

1. Go to the AWS Secrets Manager console.
2. Choose "Store a new secret."
3. Select "RDS database credentials" and enter the required details.
4. Choose "Next" and configure the secret name and description.
5. Optionally, add tags to manage the secret.
6. Finish creating the secret.

### **Step 2: Configure Rotation for the Secret**

1. In the Secrets Manager console, select the secret you created.
2. In the "Secret details" section, choose "Edit rotation."
3. Enable automatic rotation and choose "Every 90 days" (3 months).
4. Create a new Lambda function to handle rotation or choose an existing one.

### **Step 3: Set Up a Lambda Function for Rotation**

1. Create a Lambda function with the necessary IAM permissions to access Secrets Manager and RDS.
2. Add the following code to handle the rotation process

python

Copy code

import boto3

import json

import os

import smtplib

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

secretsmanager\_client = boto3.client('secretsmanager')

rds\_client = boto3.client('rds')

ssm\_client = boto3.client('ssm')

region\_name = os.environ['AWS\_REGION']

def lambda\_handler(event, context):

arn = event['SecretId']

token = event['ClientRequestToken']

step = event['Step']

if step == 'createSecret':

create\_secret(arn, token)

elif step == 'setSecret':

set\_secret(arn, token)

elif step == 'testSecret':

test\_secret(arn, token)

elif step == 'finishSecret':

finish\_secret(arn, token)

def create\_secret(arn, token):

# Generate new credentials

new\_username = generate\_username()

new\_password = generate\_password()

# Store new credentials in Secrets Manager

secretsmanager\_client.put\_secret\_value(

SecretId=arn,

ClientRequestToken=token,

SecretString=json.dumps({'username': new\_username, 'password': new\_password}),

VersionStages=['AWSPENDING']

)

def set\_secret(arn, token):

# Retrieve the new credentials

pending\_secret = secretsmanager\_client.get\_secret\_value(SecretId=arn, VersionId=token, VersionStage='AWSPENDING')

secret\_dict = json.loads(pending\_secret['SecretString'])

# Update the RDS instance with the new credentials

update\_rds\_credentials(secret\_dict['username'], secret\_dict['password'])

def test\_secret(arn, token):

# Test the new credentials by connecting to the database

pending\_secret = secretsmanager\_client.get\_secret\_value(SecretId=arn, VersionId=token, VersionStage='AWSPENDING')

secret\_dict = json.loads(pending\_secret['SecretString'])

if not test\_rds\_credentials(secret\_dict['username'], secret\_dict['password']):

raise ValueError("Failed to authenticate with new credentials")

def finish\_secret(arn, token):

# Move the secret to the AWSCURRENT stage

secretsmanager\_client.update\_secret\_version\_stage(

SecretId=arn,

VersionStage='AWSCURRENT',

MoveToVersionId=token,

RemoveFromVersionId=secretsmanager\_client.describe\_secret(SecretId=arn)['VersionIdsToStages']['AWSCURRENT'][0]

)

# Update the credentials in SSM Parameter Store

update\_ssm\_parameter(arn)

# Send the new credentials to specified email addresses

send\_email\_notification(secret\_dict['username'], secret\_dict['password'])

def generate\_username():

return 'new\_user'

def generate\_password():

return 'new\_password'

def update\_rds\_credentials(username, password):

# Implement the logic to update the RDS credentials

pass

def test\_rds\_credentials(username, password):

# Implement the logic to test the RDS credentials

return True

def update\_ssm\_parameter(arn):

# Retrieve the current secret value

current\_secret = secretsmanager\_client.get\_secret\_value(SecretId=arn, VersionStage='AWSCURRENT')

secret\_dict = json.loads(current\_secret['SecretString'])

# Update the SSM parameter

ssm\_client.put\_parameter(

Name='/path/to/your/parameter',

Value=json.dumps(secret\_dict),

Type='SecureString',

Overwrite=True

)

def send\_email\_notification(username, password):

sender = 'your-email@example.com'

recipients = ['email1@example.com', 'email2@example.com']

subject = 'RDS Credentials Rotation'

body = f'New RDS credentials:\n\nUsername: {username}\nPassword: {password}'

msg = MIMEMultipart()

msg['From'] = sender

msg['To'] = ', '.join(recipients)

msg['Subject'] = subject

msg.attach(MIMEText(body, 'plain'))

with smtplib.SMTP('smtp.example.com', 587) as server:

server.starttls()

server.login(sender, 'your-email-password')

server.sendmail(sender, recipients, msg.as\_string())

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### 

### **Step 4: Update the Credentials in Your RDS Instance**

Implement the update\_rds\_credentials function to update the RDS instance with the new credentials. This typically involves using the RDS API to modify the DB instance's master user password.

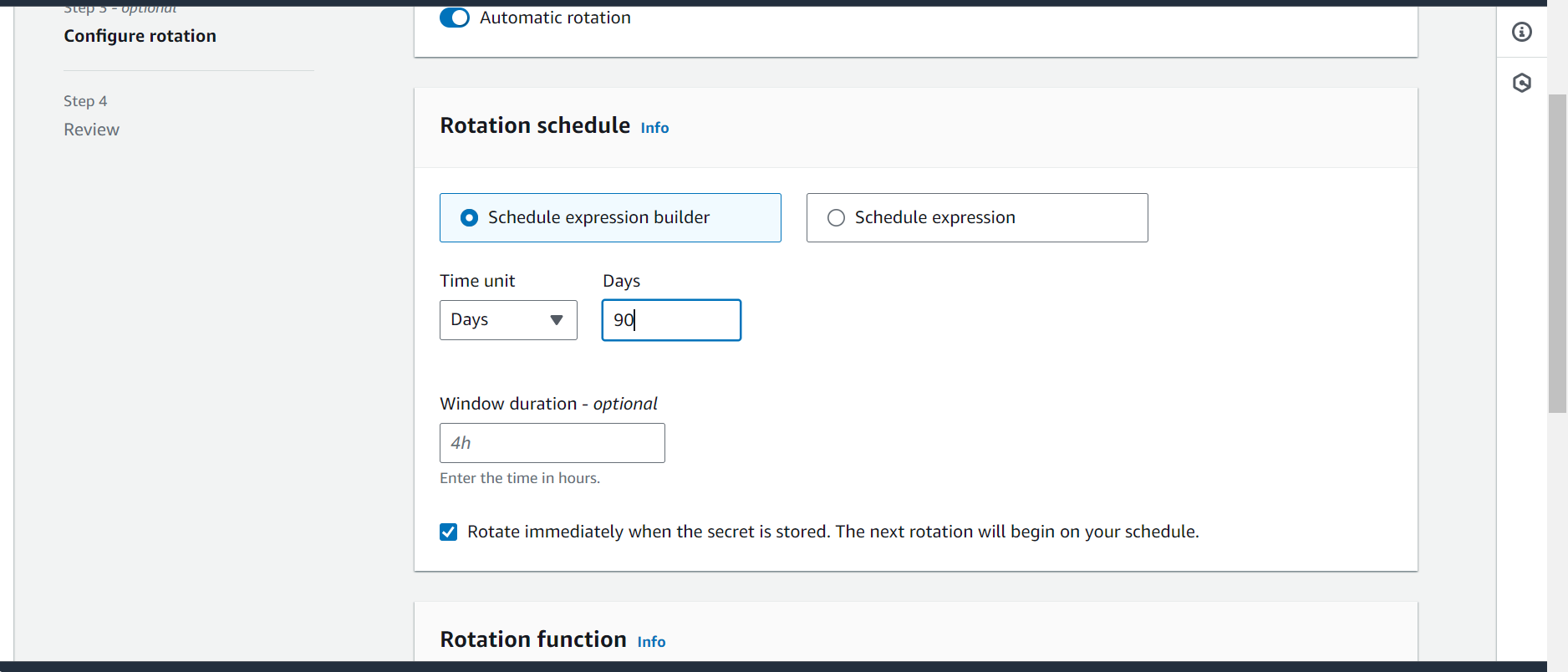
### **Step 5: Send New Credentials via Email**

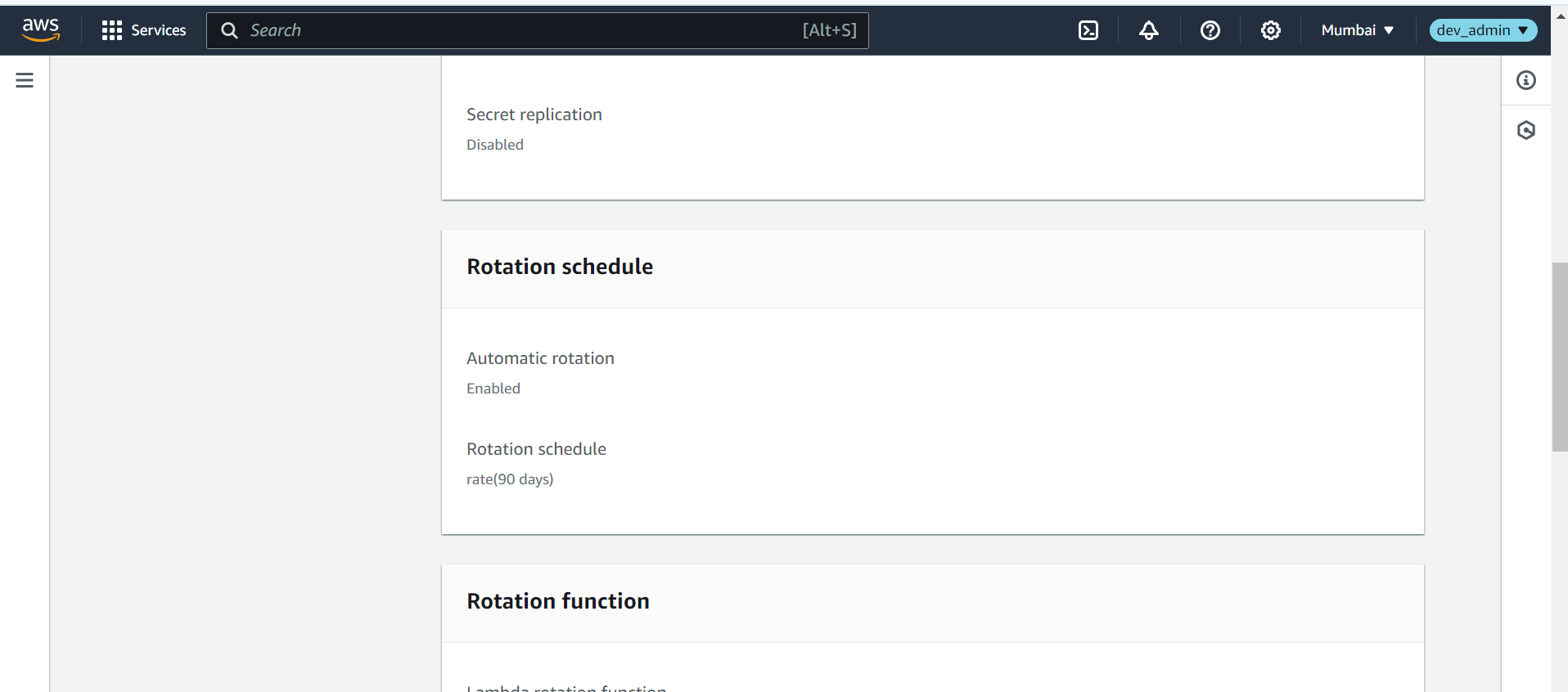
The send\_email\_notification function sends the new credentials to a specified list of email addresses. You will need to set up your SMTP server details and credentials in the function.

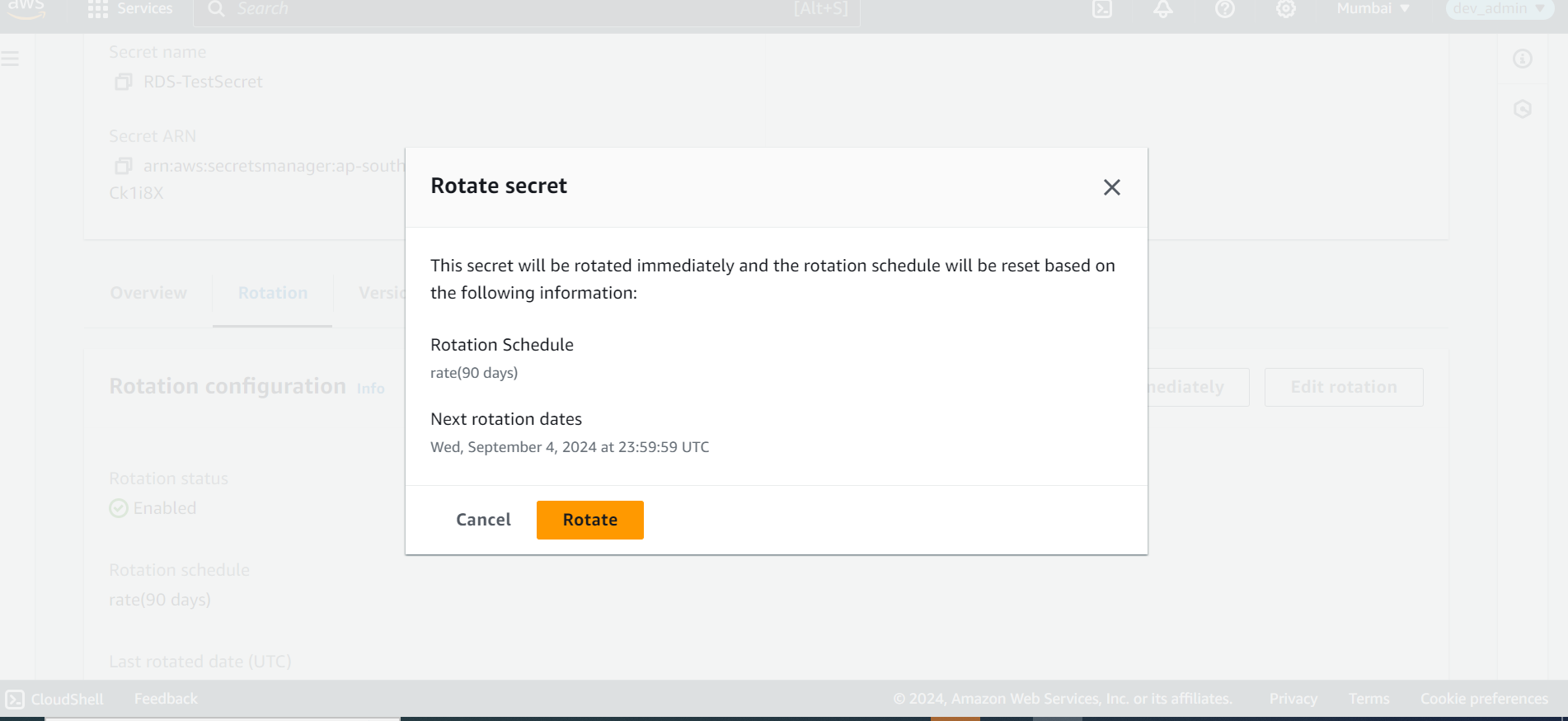
### **Note**

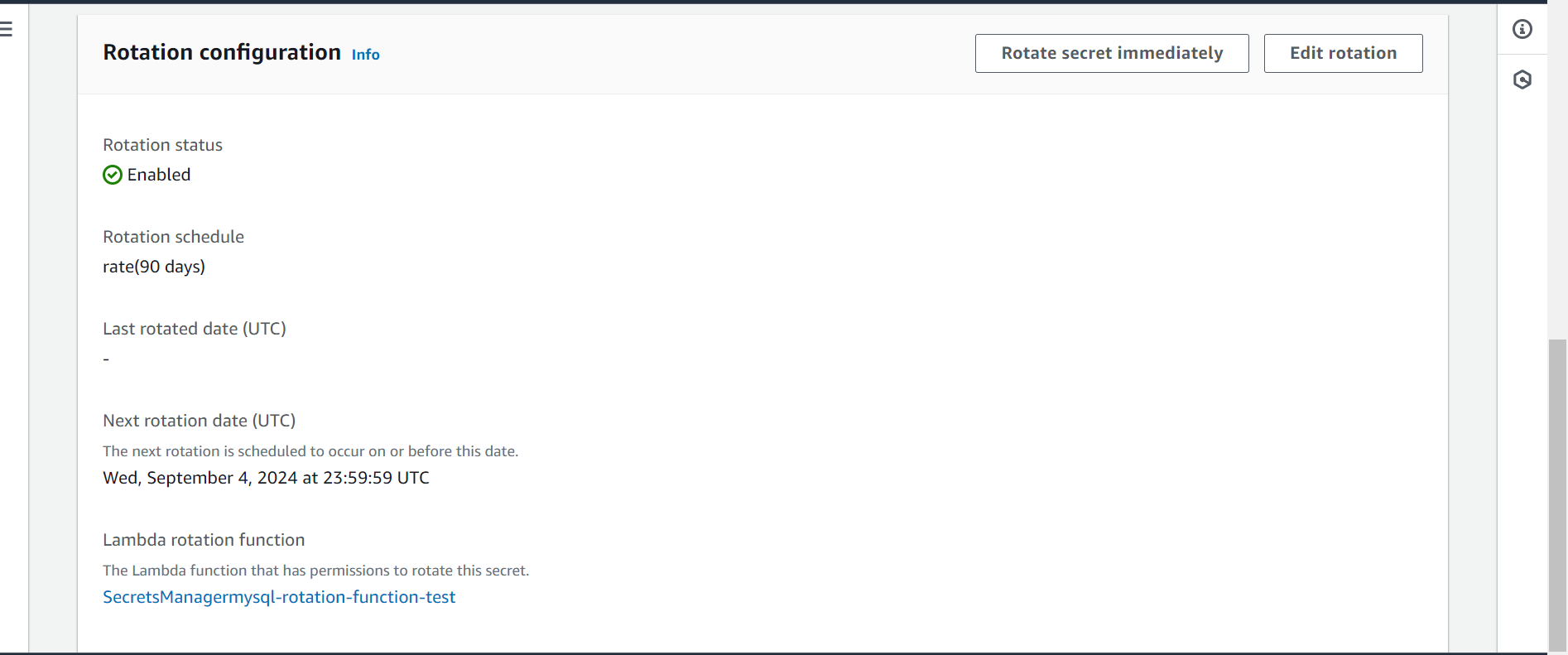
* Ensure your Lambda function has the necessary IAM permissions to access Secrets Manager, RDS, and SSM.
* Make sure to handle the security and compliance aspects, such as securing the email transmission and protecting the credentials.
* Test the rotation process thoroughly in a development environment before deploying it to production.

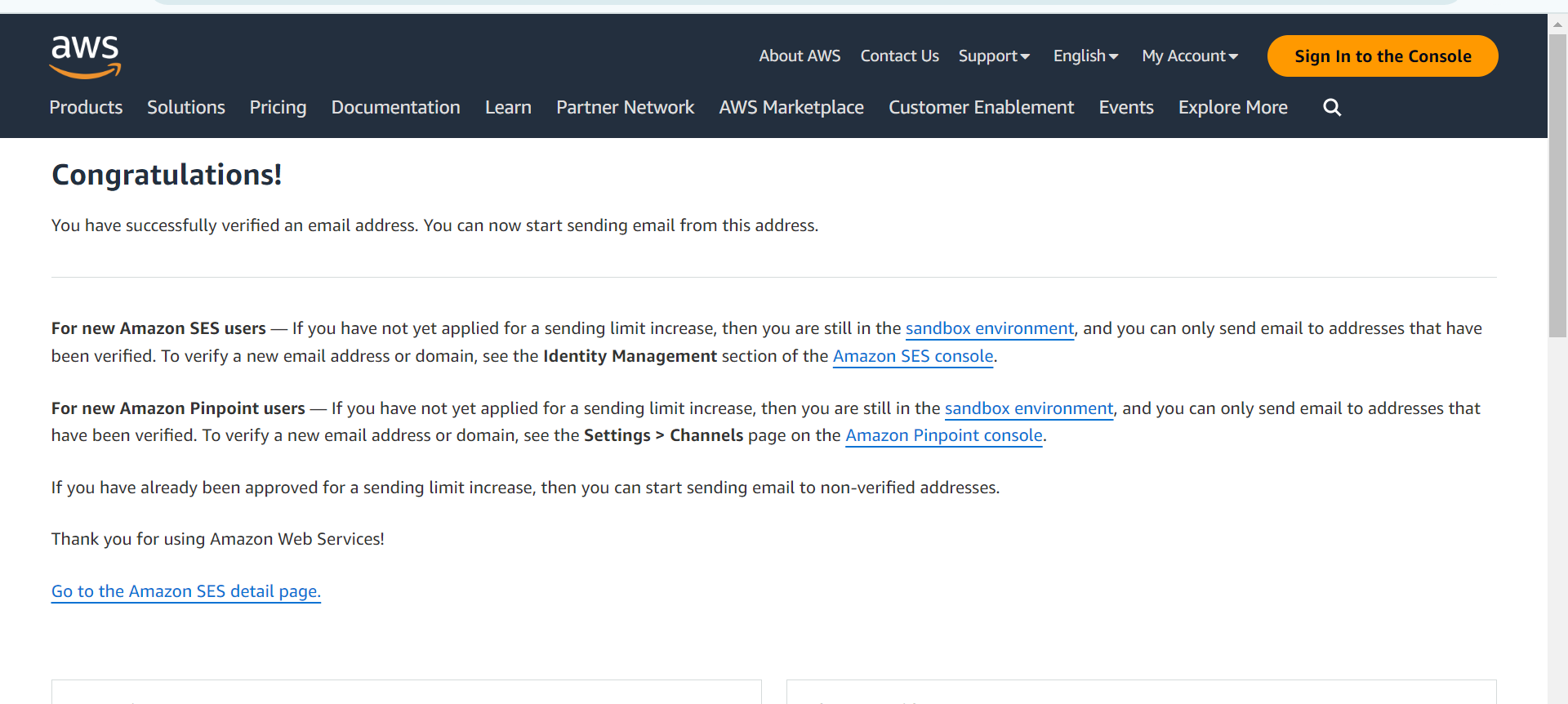
By following these steps, you can implement a robust solution for auto-rotating RDS credentials and securely sharing them with the specified recipients.

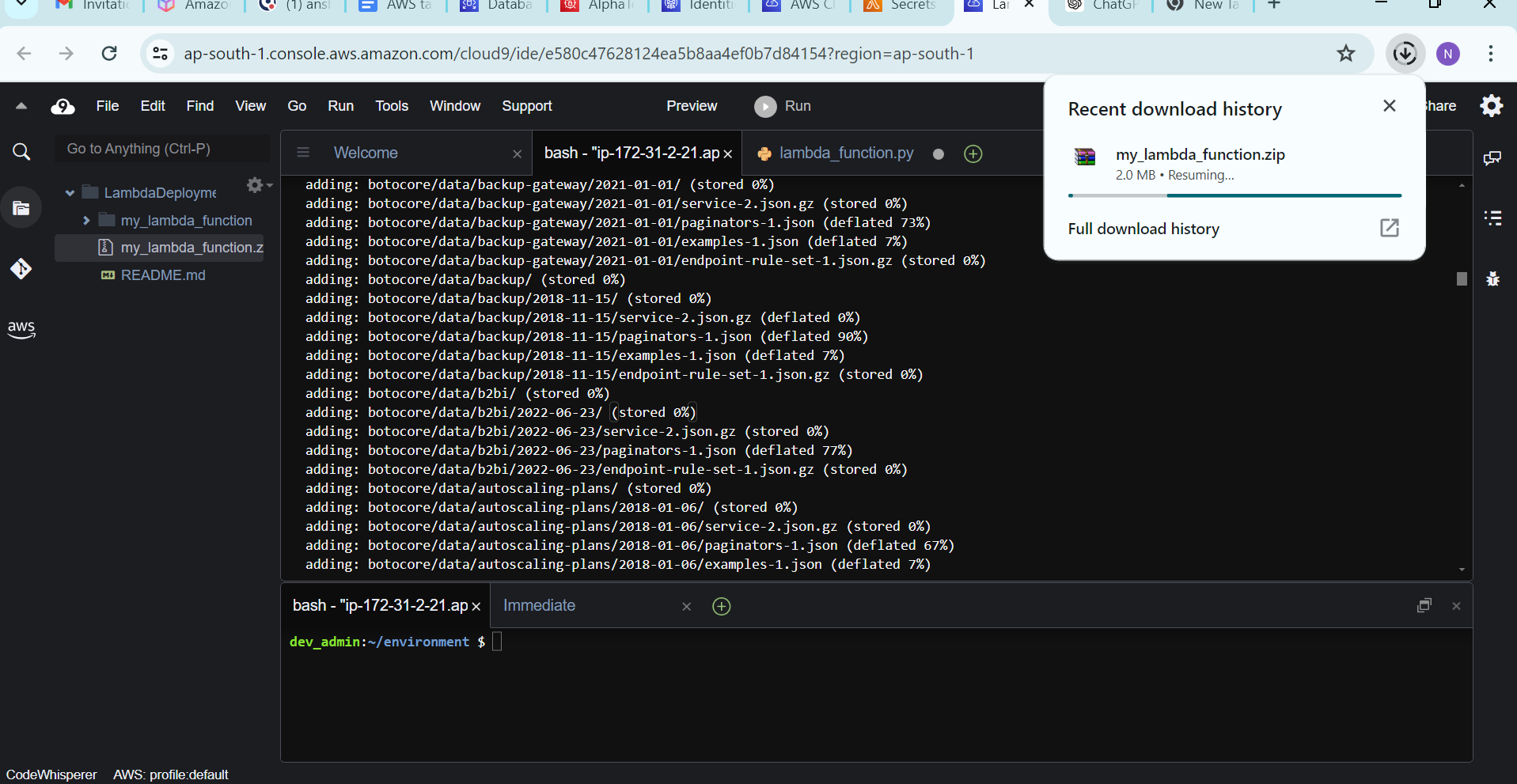


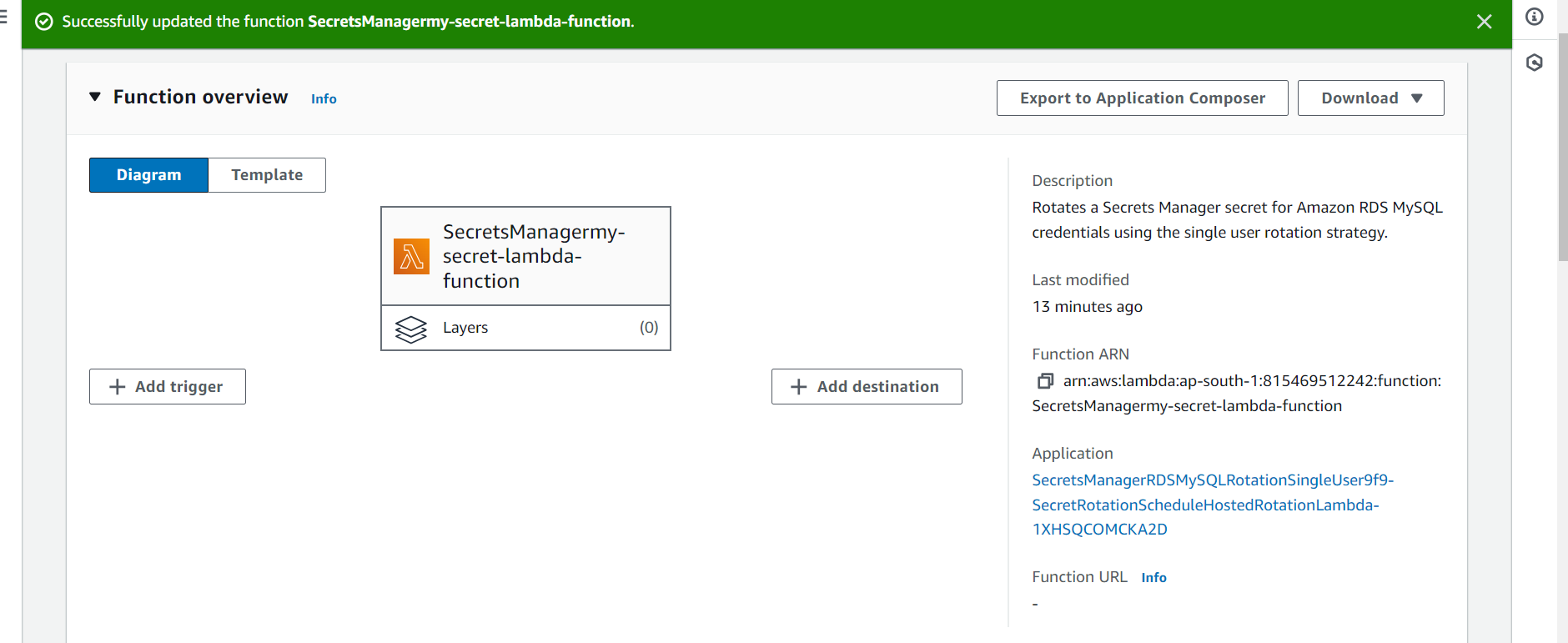


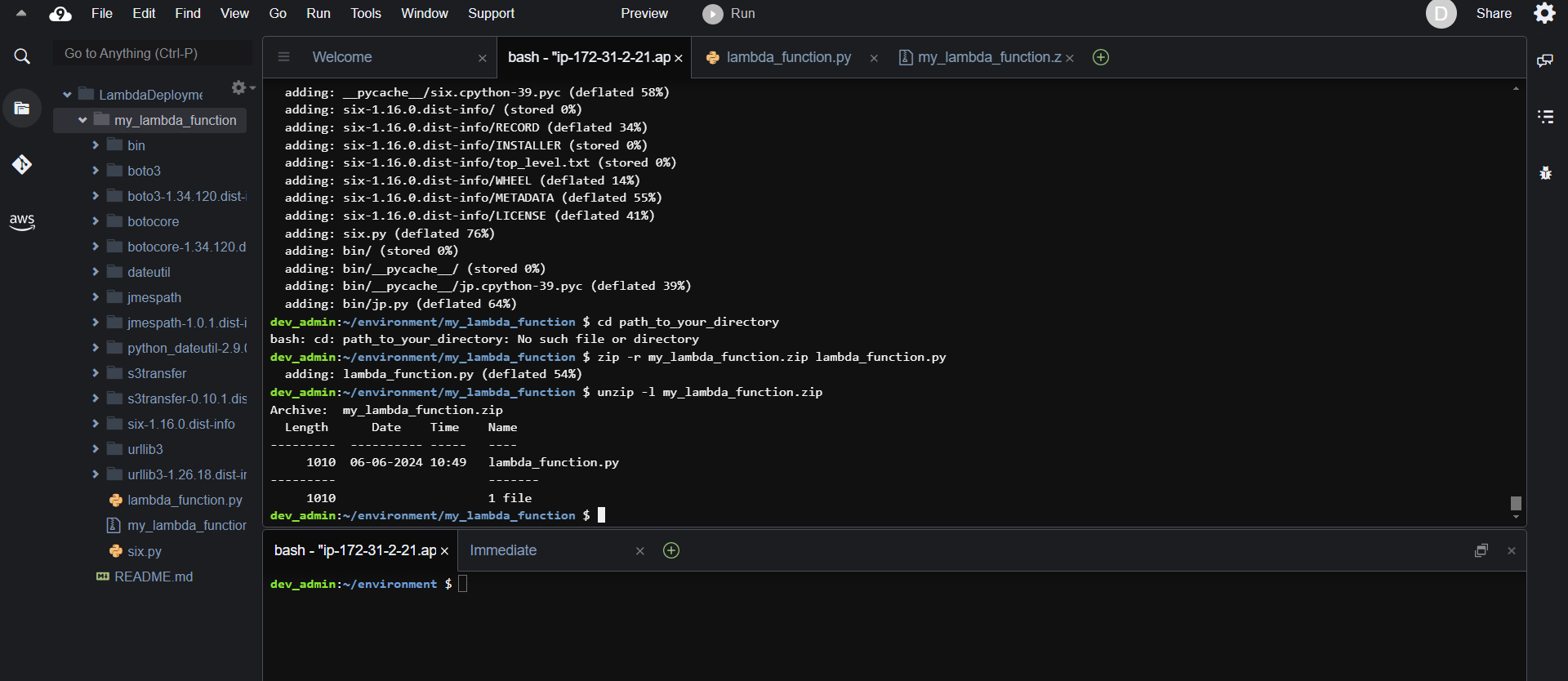


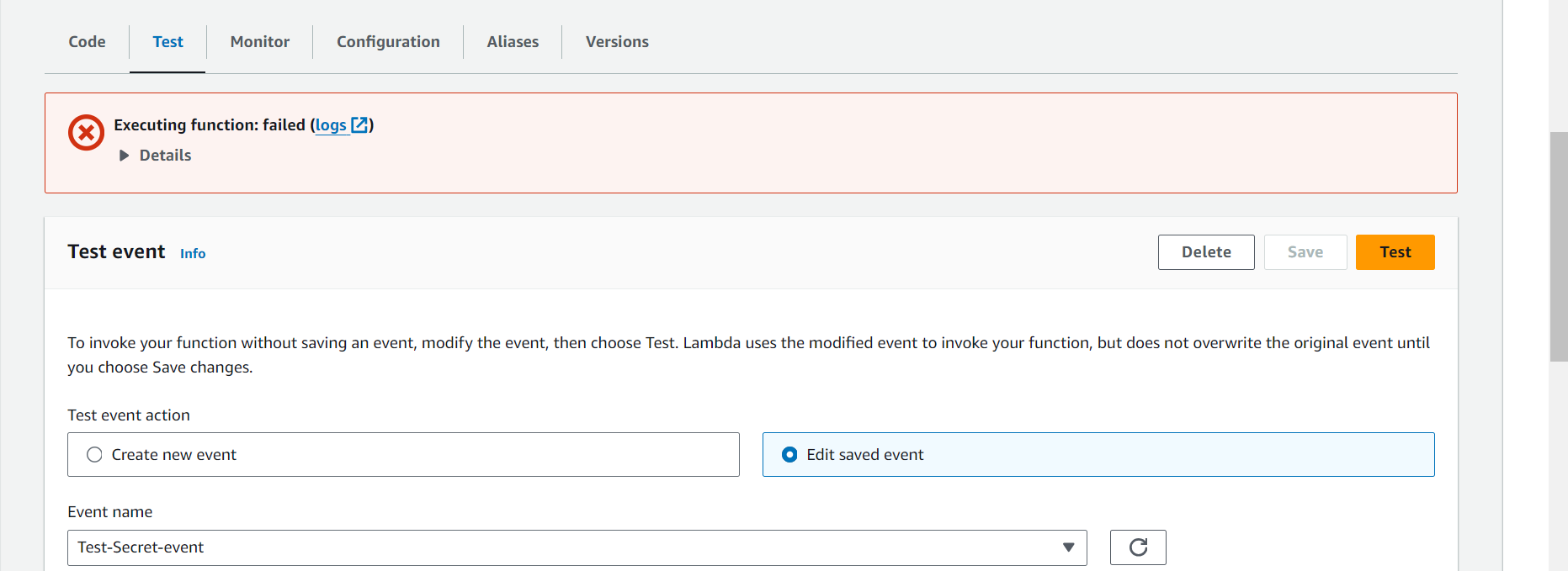


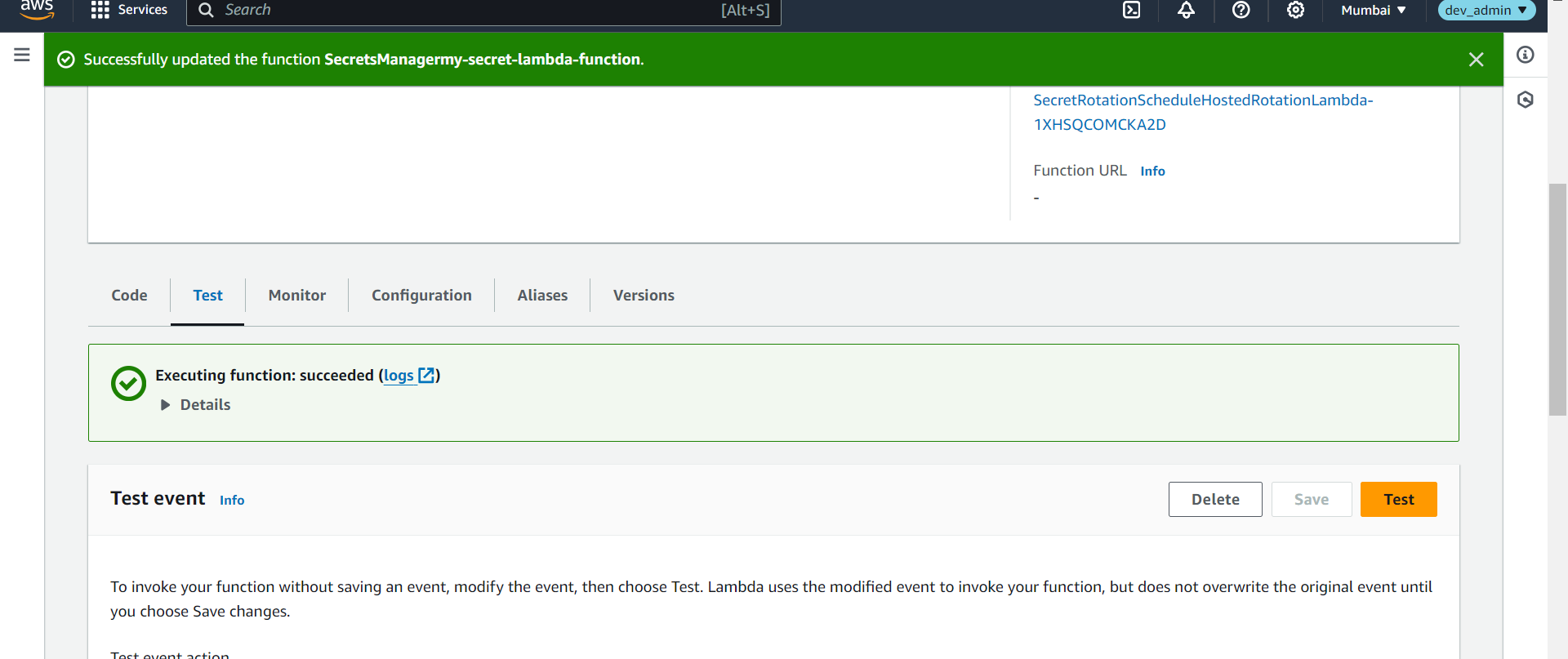


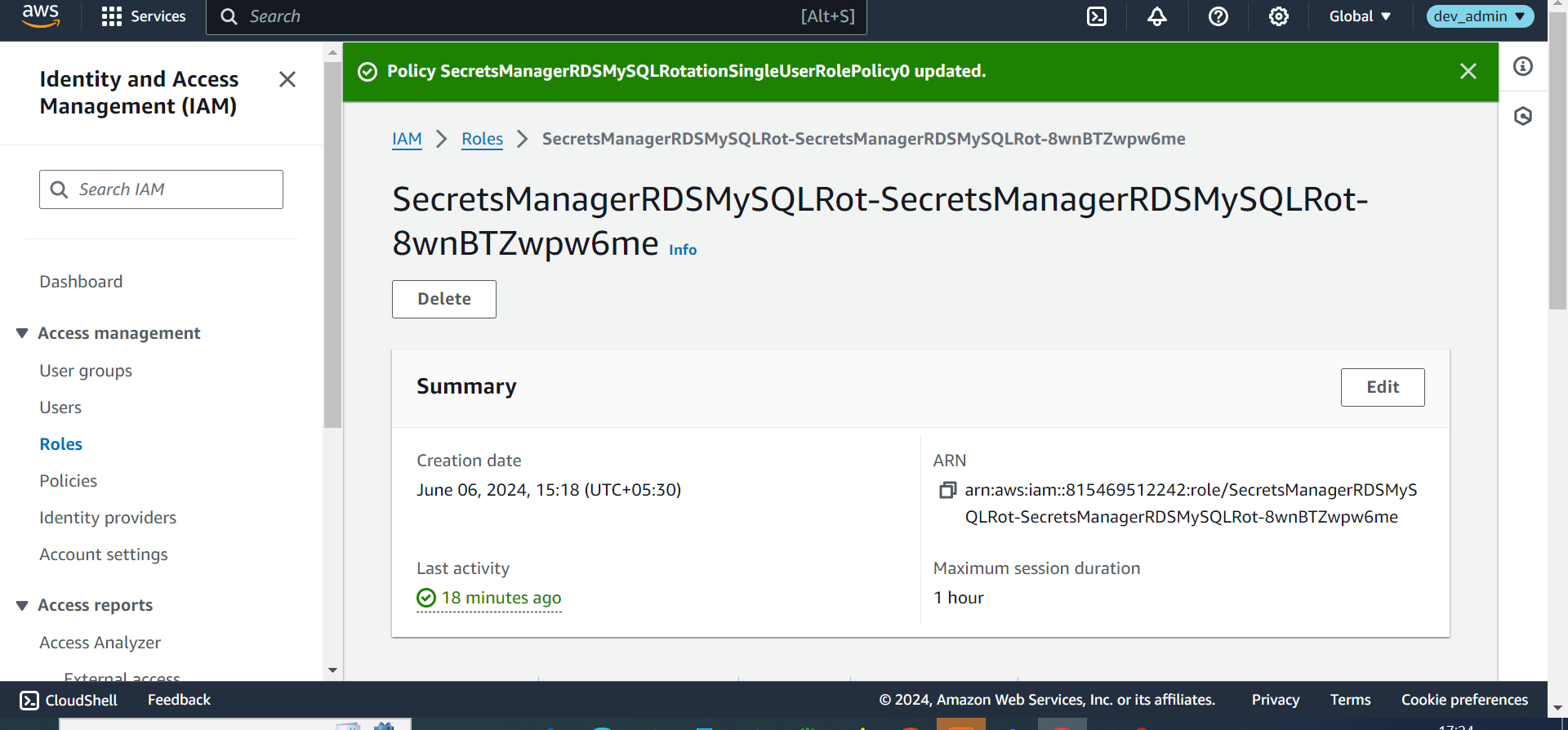


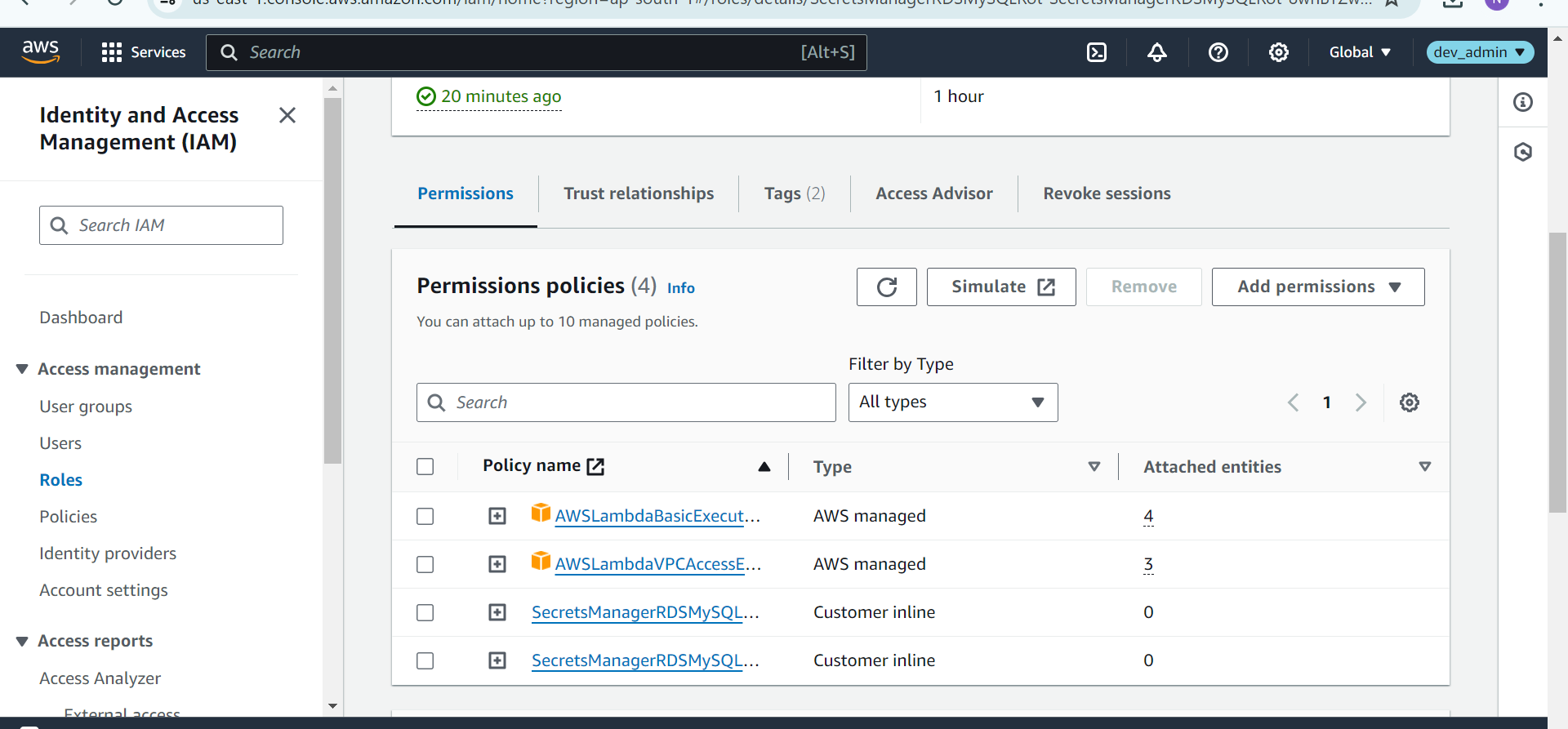


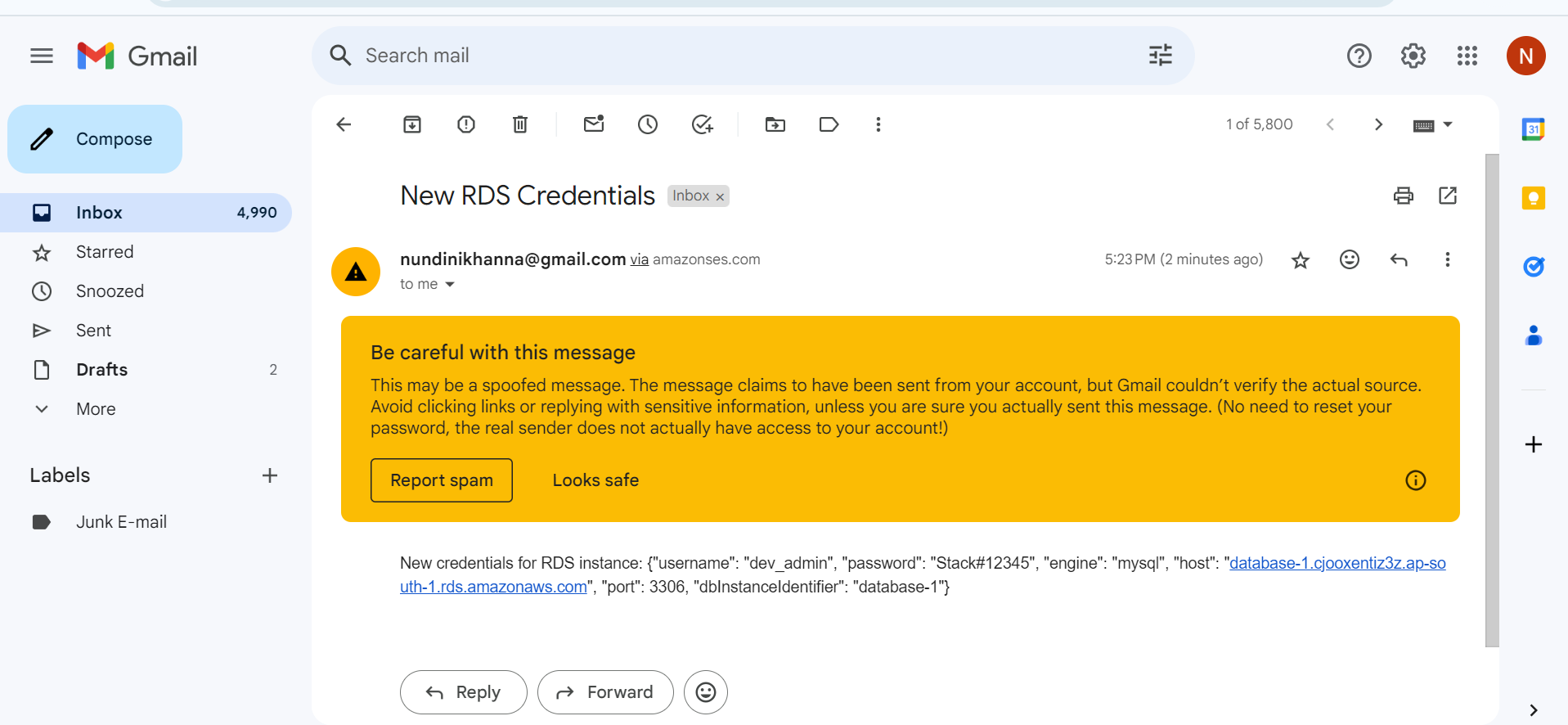












### **Some updates on Task 1**

